Abstract of “Learning from ambiguous examples” by Stuart J. D. Andrews, Ph.D., Brown University, August 2006.

The main drawback of the supervised learning approach to solving pattern classification problems is that the initial instance-label pairs are often expensive to collect due to required human effort or comprehensive testing. In many applications however, it is evidently more practical and sometimes essential to collect training examples that are ambiguous due to polymorphism or missing labels. Since these ambiguous examples have a small number of interpretations as instance-label pairs, they are still informative. It is therefore of great interest to practitioners and machine learning researchers alike to develop principled methods that can utilize such examples. This thesis demonstrates that the burden of collecting a large number of examples may be supplanted with algorithms that learn from readily available inputs that are ambiguous. This thesis presents a statistical learning theoretic framework for learning from ambiguous examples that is based on a novel formalization of disambiguation consistency. Intuitively, a valid interpretation and concept hypothesis must be mutually reinforcing. Using this principle, disambiguation and learning are jointly formulated as a non-convex maximum-margin problem. The first presented algorithmic approach for solving the disambiguation and learning problem uses a 2-stage mixed-integer local search technique that leverages state-of-the-art support vector machine software. The subsequent two algorithms use novel specializations of methods from disjunctive programming, a branch of combinatorial optimization. The first and third algorithms are of practical importance because they are efficient and scale to large data sets. Empirical results on benchmark data sets from the multi-instance and transductive learning domains are provided. These results demonstrate that, by accounting for ambiguity explicitly, classifier accuracy does not degrade and can improve significantly over techniques that ignore ambiguity. The results also suggest that our approach is best suited for tasks in which one and only one interpretation of each ambiguous example is valid, which is a reasonable assumption when the ambiguity is due to missing labels in the training data.

Learning from ambiguous examples

by
Stuart J. D. Andrews
B. S., University of Toronto, 1997
M. Sc., University of Toronto, 2000
M. Sc., Brown University, 2005

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
August 2006

This dissertation by Stuart J. D. Andrews is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date ________________ ____
Thomas Hofmann, Director

Recommended to the Graduate Council

Date ________________ ____
Michael Black, Reader

Date ________________ ____
Pascal van Hentenryck, Reader

Approved by the Graduate Council

Date ________________ ____
Sheila Bond
Dean of the Graduate School and Research

Vita

Vita goes here.

Preface

This thesis documents my doctorate research on multiple-instance learning.

Acknowledgements

I would like to thank my family and friends for their continued support during these past years.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Learning from ambiguous examples

This thesis is concerned with pattern classification; loosely speaking, the study of *categorizing things*. Things will be referred to as *instances*, and their categories will be given *labels*. Supervised learning refers to a process in which examples are used to infer a rule for guessing the category of things (predicting instance labels). A basic requirement of traditional supervised learning algorithms is that the inputs consist of instance-label pairs that are *examples* of the concept to be learned.

The main drawback of the supervised learning approach to solving pattern classification problems is that the training data are often expensive to collect. This is because the training data must include categorizations for a large number of items of interest. These must be provided by a human supervisor or verified through comprehensive testing. Moreover, confounding the task is the general experience that, the more specific a categorization is, the more difficult it is for a human to provide it. In this situation, it is difficult and sometimes impossible to collect the data required to train the classifier.

In many applications however, it is evidently more practical and sometimes essential to collect training examples that are ambiguous due to polymorphism or missing labels. Some examples are given below.

1. In text classification [4], a first step toward classifying a document is the classification of individual paragraphs. When a document is decomposed into paragraphs, it is said to have a polymorphic representation. However, when a human is labeling documents for supervised learning, it is more typical, intuitive and convenient to provide binary or keyword labels at the document level, than it is to annotate the topics of individual paragraphs within documents. The assumption is that such labels are derived from individual paragraphs. Such labeled documents are ambiguous since the association between the label and the particular paragraph from which the label was derived is lost. In other words, there are several ways to interpret the label as a paragraph-level label. The distinct interpretations arise by associating the label with each paragraph in turn.

2. In content-based image categorization [4, 12, 16, 60], a first step toward classifying an image is the

categorization of individual image segments. Polymorphic representations of images are typically obtained by running a keypoint detector over the image or by segmenting the image into regions. On the other hand, when a human is labeling images for supervised learning, it is more typical, intuitive and convenient to provide binary or keyword labels at the image-level than it is to provide geometric annotations for segments or keypoints within images. The assumption is that such labels are derived from individual segments or keypoints. Such labeled images are ambiguous since the association between the label and the particular segment or keypoint from which the label was derived is lost. In other words, there are several ways to interpret the given label as a segment-level or keypoint-level label. The distinct interpretations arise by associating the label with each segment or keypoint in turn.

3. In data mining applications [50, 56], ambiguous examples of the relationships between data items are readily available, or can be inferred directly from data base entries. For example, a polymorphic representation consisting of the purchases of an credit card cardholder can be associated with a profile of the cardholder. In this case, the assumption is that the profiles are derived from individual purchases made by the cardholder.

4. In drug-design applications [27], it is sometimes convenient to utilize a polymorphic representation for molecules. This is because a molecule is dynamic, assuming many different molecular conformations (shapes) that are either reactive and un-reactive, but cannot be distinguished during a chemical reaction. The assumption, proposed by organic chemistry, is that the molecular reactivity is derived from individual conformations. A polymorphic representation of the molecule is obtained by sampling conformations. Distinct interpretations arise by associating the reactivity with each conformation.

5. In stock portfolio prediction [47] and protein family modeling [66], it is natural to decompose the input sequences to detect an event or motif at an arbitrary position. Here, the polymorphism is analogous to that considered for content-based image categorization. Similarly, the assumption is that the labels provided at the sequence-level by humans are derived from individual events or motifs within the sequence. Distinct interpretations arise by associating the label with each event or motif in turn.

In the previous examples, the source of ambiguity is the polymorphic representation of the input. In the next and last example, ambiguity comes from not knowing the labels for a subset of the inputs.

6. Finally, in practically any pattern classification application, from optical character recognition to diabetes prediction [14, 17, 39], it is trivial to collect unlabeled inputs. Unlabeled inputs are ambiguous because their label is not specified. There are several ways to interpret such unlabeled inputs. Unlabeled examples clearly have labels, only their true identity is not known. The distinct interpretations arise by associating each possible label with the input.

In these examples, the inputs that are readily available are called *ambiguous examples*, since they have multiple interpretations as unambiguous examples (instance-label pairs). The fact that these inputs are ambiguous does not mean that the inputs are vague. Ambiguity is modeled explicitly in a manner that reflects explicit identifiable structures or meaningful invariants of an input.

While there are many applications where ambiguous examples are relatively easy to collect in comparison to fully labeled examples, this is not always the case. Typically there is a trade-off between the quantity and specificity of training data that can be obtained with fixed resources such as patience, time or energy. One can collect either: 1) a small set of unambiguous examples, 2) a large set of ambiguous examples, or 3) some combination of unambiguous and ambiguous examples in between. The relative costs involved in choosing the number and type of examples are application-specific. In some cases, there are constraints inherent in the application that dictate that examples are ambiguous. In other applications, when potential sources of ambiguity are known and there are resources to deal with them, it may be possible to collect fully labeled unambiguous examples. The basic fact stands that for several applications, it is evidently more practical to collect ambiguous examples. It is therefore of great interest to practitioners and machine learning researchers alike to develop principled methods that can utilize ambiguous examples.

The work in this thesis is based on a simple observation, which is the following: Despite the fact that ambiguous examples have multiple interpretations, they are still informative. This is especially true when considering any number of ambiguous examples at once, since the valid interpretations of all ambiguous examples must be mutually consistent. Unfortunately, supervised learning is not directly applicable. Moreover, the task of selecting interpretations is fundamentally distinct from feature selection, because interpretations must be chosen on an example by example basis whereas features are chosen in parallel. The key question then is: Is it possible to recover the implicit categories underlying a collection of ambiguous examples? This thesis answers in the affirmative. The burden of collecting a large number of instance-label pairs may be supplanted with algorithms that learn from readily available ambiguous examples. As in the supervised case, the goal is to infer a rule for predicting individual instance labels. This rule will be called a *classifier*.

This thesis is concerned with learning from examples that are ambiguous, as observed in several semi-supervised learning problems. Specifically the problems are multiple-instance learning, transductive inference and semi-supervised inductive inference. Multiple-instance learning addresses the problem of learning from inputs that are ambiguous due to polymorphism. Examples of polymorphic inputs are:

1. multiple instances of the same modality
2. multiple segments from a part-whole decomposition
3. multiple samples from a stochastic process

There are two common terms for such inputs: *multi-instances*, and *bags*. The first term *multi-instances* captures the direct meaning of an input consisting of many things of the same type. The second term *bag* evokes the image of a container holding the individual instances in a jumbled and meaningless order. The multi-instance and label comprise an *ambiguous example*. The multi-instance label models a 1-of-N relationship. In other words, the assumption is that one of the instances within a multi-instance is responsible for the label. While each instance within a multi-instance belongs to a category, these instance-level individual labels are not included directly in the input. However, for true/false (T/F) labels, the multi-instance label is a logical disjunction of the instance-level labels. Therefore, as outlined in the examples given in this section, it is natural to consider a labeled multi-instance as having several *interpretations* as an example (instance-label pair) of the concept to be learned.

Transductive learning and semi-supervised inductive inference deal with the problem of learning from a small collection of examples (instance-label pairs) that is augmented by a large collection of unlabeled inputs. The unlabeled inputs are ambiguous. Each unlabeled input belongs to a category although its label is not provided in the input. Assuming T/F labels, an unlabeled input has two interpretations as an example of the concept to be learned and only one is valid. Transductive inference focuses only on making predictions for the finite set of inputs that are unlabeled, while the goal of semi-supervised inductive inference is to infer a rule for predicting arbitrary instance labels.

Interestingly, under certain weak assumptions, transductive learning and semi-supervised inductive inference can both be reduced to multi-instance learning. The reduction follows by converting the unlabeled and ambiguous inputs from these problems, into polymorphic inputs (multi-instances) with specific labels. This reduction to multi-instance learning aligns all sources of ambiguity under one framework.

A systematic treatment of ambiguity has the promise of extending the utility of traditional pattern classification techniques. In order to derive useful information from ambiguous examples requires inferences involving the interpretations of ambiguous examples. The process of selecting an interpretation is called *disambiguation*. The key intuition is that *the true interpretation is self-reinforcing*. To formalize this idea, this thesis proposes a risk minimization principle for learning from ambiguous examples. Several algorithms are developed within the framework for learning from ambiguous examples, and subsequently analyzed. Learning from ambiguous examples is a principled way to make specific and plausible inferences about ambiguous inputs in a pattern classification system. The significance of this framework is evident from the number and diversity of annotation-starved domains to which it has been applied.

Contributions of this thesis include:

1. Two data sets

2. A structural risk minimization / maximum-margin framework for learning from ambiguous examples

   This framework encompasses multi-instance learning, semi-supervised learning and transductive inference

3. A pictorial framework for conveying principles of risk minimization and convex programming

4. One algorithm for learning from ambiguous examples using a 2-norm formulation of the maximum-margin problem

   (a) *Multi-instance support vector machine* (MI-SVM): this algorithm is characterized by integer disambiguation variables and a greedy local search heuristic.

5. Three algorithms for learning from ambiguous examples using a 1-norm formulation of the maximum-margin problem

   (a) *Disjunctive programming boosting* (DPBoost): this algorithm is derived using geometric concepts from disjunctive programming and convex programming resulting in a very large linear program.

   (b) *1-norm multi-instance support vector machine* (1-norm MI-SVM): similar to the first algorithm but using the 1-norm formulation.

   (c) *Lift and project boosting* (LNPBoost): this algorithm is characterized by an efficient cutting-plane approximation strategy and theoretical performance guarantees.

6. Empirical evaluations on multi-instance learning and transductive learning domains.

Chapter 2

Examples

Some concrete applications of learning from ambiguous examples are discussed below.

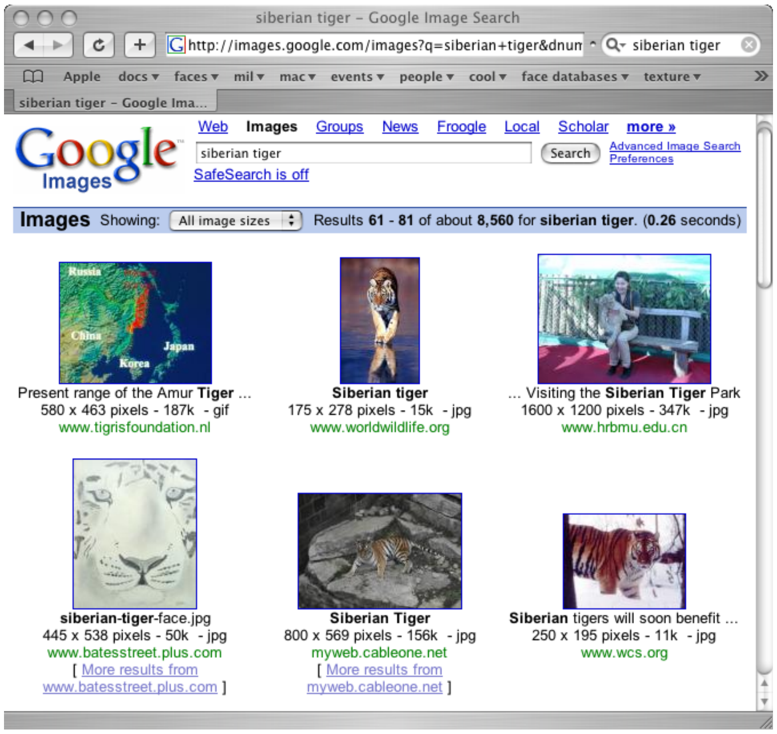
2.1 Automatic image annotation

Content-based image retrieval based on subject matter is a valuable tool for artists as a means to stimulate their creativity. Some art schools provide students with archives of photographs and other artwork that has been collected from current periodicals and indexed by subject matter. As a result of the careful indexing of this collection, a student who wishes to develop an artistic theme based on tigers or elephants can easily retrieve relevant source material and thus inspiration.

Building image archives is generally done by scavenging resources such as the world wide web, unless one has the funding and patience required to send professional photographers around the world. Two popular methods for building image archives are described. The first technique is to bootstrap a text-based search engine. The second is to leaf through image-laden periodicals. In the first technique, assuming one wants to collect images of the Siberian tiger species, one proceeds by: 1) searching by text query, and 2) separating *positives* from *negatives* by hand, as depicted in Figure 2.1. This process can be repeated for as many categories as desired. The second popular technique used to build image archives requires, at one art school, several full-time librarians and many periodical subscriptions. The employees cut images from the periodicals and file them into one of a wide range of categories based on their own judgements of dominant subject matter.

Both approaches require a significant amount of human labeling effort; especially when considering that search engines can return a large number of query results, and periodicals are practically limitless. Then there is the issue of updating these collections over time. In order to relieve the humans required for annotation, the idea of automatic image annotation is to design a classifier to identify the images of a category using a learning algorithm. Since there are potentially many subject categories, the advantage of using a learning algorithm is great.

Automatic classification of images based on their subject matter is a difficult problem. In computer vision, the problem is called object or scene recognition. In data mining, it is called image retrieval. The recognition


siberian tiger – Google Image Search
http://images.google.com/images?q=siberian+tiger&dnum
siberian tiger
Apple docs faces mil mac events people cool face databases texture
siberian tiger – Google Ima...
Web Images Groups News Froogle Local Scholar more »
Google Images
siberian tiger
Search
Advanced Image Search
Preferences
SafeSearch is off
Images Showing: All image sizes
Russia
China
Japan
Korea

Results **61** - **81** of about **8,560** for **siberian tiger**. (**0.26** seconds)

Present range of the Amur **Tiger** ...
580 x 463 pixels - 187k - gif
www.tigrisfoundation.nl

Siberian tiger
175 x 278 pixels - 15k - jpg
www.worldwildlife.org

... Visiting the **Siberian Tiger** Park
1600 x 1200 pixels - 347k - jpg
www.hrbmu.edu.cn

siberian-tiger-face.jpg
445 x 538 pixels - 50k - jpg
www.batesstreet.plus.com
[More results from www.batesstreet.plus.com]

Siberian Tiger
800 x 569 pixels - 156k - jpg
myweb.cableone.net
[More results from myweb.cableone.net]

Siberian tigers will soon benefit ...
250 x 195 pixels - 11k - jpg
www.wcs.org

Figure 2.1: Text-based image retrieval. Query results that do not contain a Siberian tiger must be detected by hand.

of images containing tigers, for example, is nontrivial due to the variations of tiger appearance and pose, as well as the variations in external viewing conditions that affect these images. A further confounding problem is that several objects beside a tiger may appear in the image. The image may also contain grass, trees, water, or in some cases maybe, a zebra. Arguably, the only way to "recognize" the tiger is to focus on the image segment occupied by the tiger. In other words, recognition depends on some form of part-whole decomposition of the image.

As described in Chapter 1, a supervised learning algorithm requires examples (instance-label pairs) of the concept to be learned. Thus, if the goal is to recognize image segments, then the training data should consist of image segments and image segment labels. On the other hand, the keyword indexing scheme is the prevailing method for indexing image databases. Keywords are relatively easy to collect for images in a training set. Moreover, they may be inferred from a caption or other text associated with the images. However, keywords are image-level annotations, and do not identify the image segment occupied by the object. In contrast, one might rely on specific geometric annotations within the image, such as keypoints and bounding boxes [36, 51, 69]. These approaches are more restrictive, and involve tedious and time-consuming manual labeling. Moreover, in practice, this approach is typically restricted to the use of only one type of geometric annotation which cannot be changed after labels have been collected. Furthermore, to collect the geometric annotations, specialized software must be developed and human annotators must be trained to use it. These reasons suggest that there is a benefit to the use of image-level keyword annotations instead of within-image geometric annotations. With image-level annotations, a learning algorithm can experiment with several part-whole decompositions of images, even after labels have been collected.

With the ability to learn from ambiguous examples, the underlying classifier for image segments may be learned from a training set consisting only of images and keywords. Then, given the learned classifier for the segments, a new image can be classified as containing the subject matter if and only if there exists a positively classified segment (1-of-N semantics).

In order to apply learning from ambiguous examples to this domain, a dataset with polymorphic examples was constructed. The original labeled images were color images from the Corel data set. These were subsequently preprocessed and segmented with the Blobworld system [20]. Blobworld generates an image segmentation by applying a clustering algorithm to the collection of pixels that comprise the image. This yields a polymorphic representation of the image. In this representation, an image consists of a set of segments (or blobs), each characterized by color, texture and shape descriptors. Figure 2.2 demonstrates the role of segmentation in the construction of ambiguous examples. Observe in Figure 2.3 that the average pixel color is a suitable predictor for the tiger when the segmentation is correct.

Three different categories (“elephant”, “fox”, “tiger”) were used to generate ambiguous data sets. In each case, the resulting data sets have 100 positive and 100 negative example images. The latter images have been randomly drawn from a pool of photos of other animals. Due to the limited accuracy of the image segmentation, the relative small number of region descriptors and the small training set size, this ends up being quite a hard classification problem. These data sets are appropriate for a comparative performance analysis of algorithms, have been made publicly available on the Internet[1], and have thus become standard test cases in the multi-instance community.

2.2 Drug activity prediction

In the process of designing drugs, candidate molecules that are believed to participate in a desired positive reaction are tested in expensive laboratory tests. For this reason, it is only possible to test a small number of carefully selected candidates. However, the natural conformational polymorphism of molecular structure complicates the nomination of candidate molecules.

Nomination of candidate molecules is difficult is for the following reason. First, a molecule is composed of a collection of atoms and connective bonds which may assume, at any given moment in time, one of several dominant 3-dimensional arrangements which are known as molecular conformations. In addition, if there is a reaction when a molecule comes in contact with a fixed reagent, by the theory of organic chemistry the reactivity is hypothesized to depend on just one molecular conformation [27]. While the likely conformations of a molecule are easy to enumerate, it is impossible to measure which of these participates in a reaction if one occurs. Therefore, even knowing that one molecule induces a desired positive reaction, since there are several conformations to consider, it is difficult to predict which other molecules will do the same.

With the ability to learn from ambiguous examples, a classifier for the underlying reactive molecular conformations may be learned from a training set consisting only of molecules and their reactivities. Then,

[1]http://www.cs.columbia.edu/~andrews/mil/datasets.html

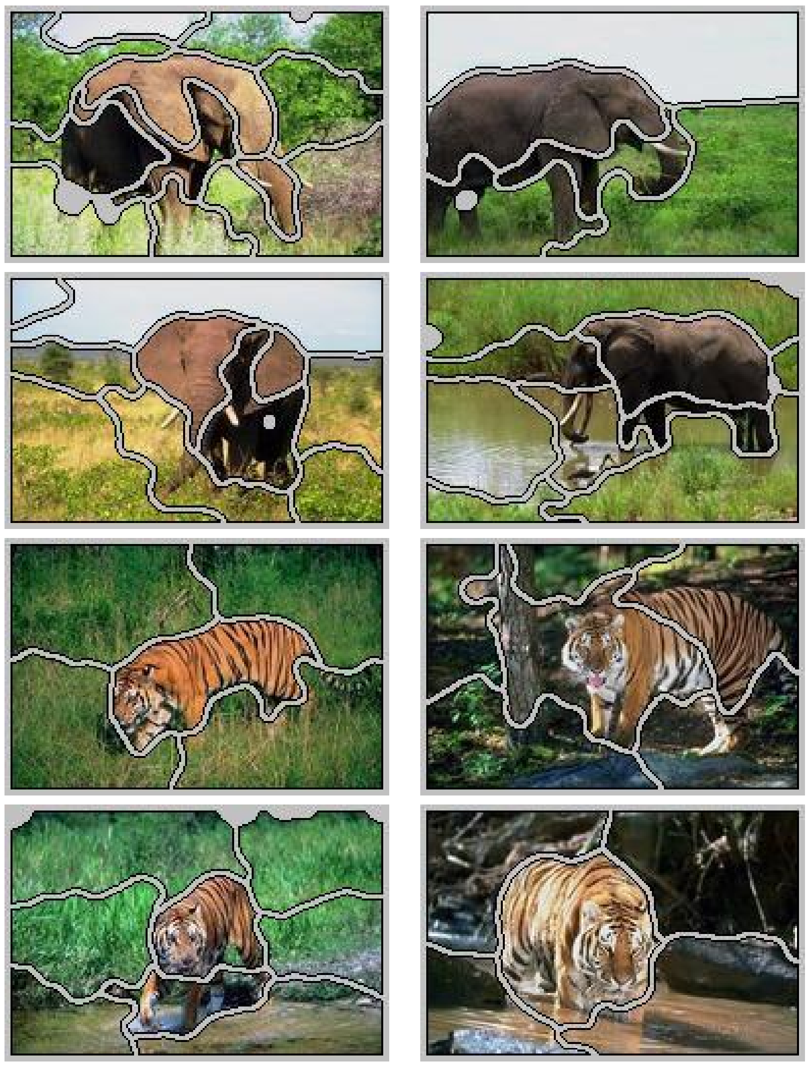


Figure 2.2: Segmentations of elephant and tiger images.

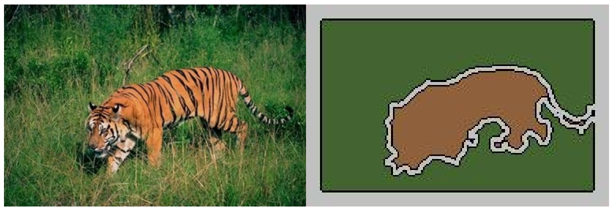


Figure 2.3: The role of segmentation in recognition. The average color of pixels in each segment determines the displayed color.

given the learned classifier for reactive molecular conformations, a new molecule will be classified as reactive if and only if it has a positively classified conformation (1-of-N semantics).

The authors of [27] created ambiguous data sets, called MUSK1 and MUSK2, with the above characteristics. The MUSK data sets are *the* benchmark data sets used in virtually all research papers on this subject. They can be downloaded from the Internet[2]. First the dominant conformations of a molecule had to be sampled. This was performed by simulating molecular dynamics, and selecting multiple low-energy conformations that occur most frequently. This yields a polymorphic representation of a molecule. Each conformation is represented by a 166-dimensional feature vector derived from surface properties. Two data sets were generated in this manner. MUSK1 contains on average approximately 6 conformation per molecule, while MUSK2 has on average more than 60 conformations in each bag. Complete details are found in the landmark paper [27]. Figure 2.4 demonstrates the role of conformations in the construction of ambiguous examples.

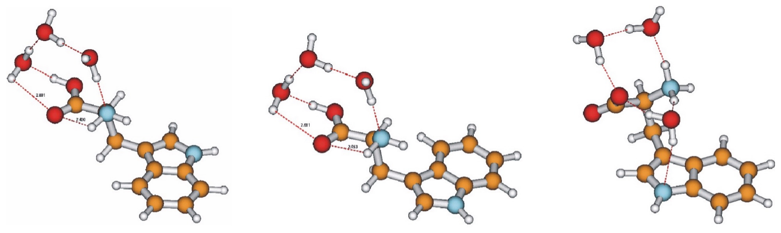


Figure 2.4: Polymorphism in molecular structure. A molecule can assume, at any given time, one of several different dominant conformations.

2.3 Text categorization

Text categorization concerns the assignment of a document to one or more topic categories. This problem is of paramount importance given the proliferation of document archives and repositories that are being made publicly available. Topic-based categorization of document collections provides a structure to navigate and access individual documents.

Categorization is easy when a document has pre-defined keywords, as is the case in carefully indexed collections such as those found in some libraries, businesses, governments and other large institutions. Due to the natural organization and presentation of textual information, the categories to which a document belongs can usually be inferred from the topics of individual paragraphs. In this thesis, this property is assumed to hold in general. On the other hand, when a document is categorized by a human, the specific paragraphs are typically not specified. Instead, the topic annotation is assigned to the document as a whole. One approach to learning from an annotated collection is to infer which paragraphs correspond to the topic annotations.

With the ability to learn from ambiguous examples, a classifier for the underlying paragraphs may be learned from a training set consisting only of documents and keyword annotations. Then, given the learned

[2]http://www.ics.uci.edu/~mlearn/MLRepository.html

classifier for paragraphs, a new document will be classified as belonging to the topic category if and only if it has a positively classified paragraph (1-of-N semantics).

In order to experiment with text categorization, several ambiguous data sets were generated starting from the publicly available TREC9 data set, which is also known as OHSUMED[3]. The original data consists of several years of selected MEDLINE articles. A subset of articles from 1987, which consists of approximately 54,000 documents, was selected for the TREC9 filtering competition. This subset was therefore an appropriate and convenient choice to use as source data. MEDLINE documents are annotated with MeSH terms (Medical Subject Headings). The total number of MeSH terms in TREC9 was 4904. The MeSH terms were separated into subsets: preliminary, train and test for the filtering competition.

From the preliminary portion, the first seven MeSH annotations that individually appeared on at least 200 documents were used to define binary concepts across the collection. For each of the seven annotations, 200 documents with the annotation and 200 without the annotation were drawn at random. Documents were split into passages using overlapping windows of maximal 50 words each, because paragraph information was not always preserved in the TREC9 source files. This yields a polymorphic representation for documents. In this representation, a document consists of a set of term frequency feature vectors that summarize the words in each passage. Figure 2.5 depicts the construction of ambiguous examples from text documents.

Compared to the other data sets the representation is extremely sparse and high-dimensional, which makes this data an interesting additional benchmark. These have also been made publicly available on the Internet[4], and have become a challenging test case for the community.

2.4 Optical character recognition

Finally, a data set for semi-supervised inductive inference was generated. The well-known database of USPS handwritten digits[5] was used, following the experiment design of [17]. A subsample containing 100 of each of the digits “0”-”3” was randomly drawn from the entire collection. Five samples from each digit category were labeled. “0” and “1” samples were labeled $+1$, while “2” and “3” samples were labeled -1. The remaining 380 digits were used to create multi-instance examples, as is described in Chapter 3, Section 3.2.1.

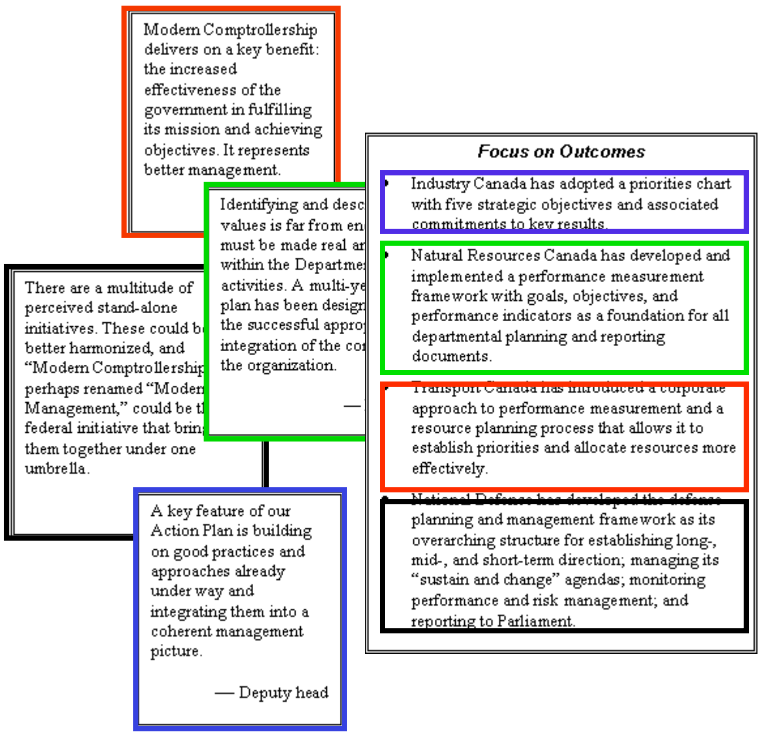
2.5 Summary

A comparison of several key characteristics of these datasets is shown in Table 2.1.

[3]http://trec.nist.gov/data/t9_filtering

[4]http://www.cs.columbia.edu/~andrews/mil/datasets.html

[5]http://www.gaussianprocess.org/gpml/data/



Modern Comptrollership delivers on a key benefit: the increased effectiveness of the government in fulfilling its mission and achieving objectives. It represents better management.

Identifying and desc
values is far from en
must be made real ar
within the Departme
activities. A multi-ye
plan has been design
the successful appro
integration of the co
the organization.

—

There are a multitude of perceived stand-alone initiatives. These could b
better harmonized, and
“Modern Comptrollership
perhaps renamed “Moder
Management,” could be t
federal initiative that brin
them together under one umbrella.

A key feature of our Action Plan is building on good practices and approaches already under way and integrating them into a coherent management picture.

— Deputy head

Focus on Outcomes

- Industry Canada has adopted a priorities chart with five strategic objectives and associated commitments to key results.
- Natural Resources Canada has developed and implemented a performance measurement framework with goals, objectives, and performance indicators as a foundation for all departmental planning and reporting documents.
- Transport Canada has introduced a corporate approach to performance measurement and a resource planning process that allows it to establish priorities and allocate resources more effectively.
- National Defence has developed the defence planning and management framework as its overarching structure for establishing long-, mid-, and short-term direction; managing its “sustain and change” agendas; monitoring performance and risk management; and reporting to Parliament.

Figure 2.5: Polymorphism in text categorization. Consecutive windows of 50 words are used to form a polymorphic representation of a text document. In the construction of the ambiguous TREC9 data sets, overlapping windows were used. Color is used to indicate different topics.

	MI examples		MI size			Features	
	ambig.	unambig.	max.	med.	total	dim.	%fill
MUSK1	47	45	40	4	207	166	100
MUSK2	39	63	1044	12	1017	166	100
COREL	100	100	13	6.3	1310	230	50.4
TREC9	200	200	19.5	8	3332	$6.6e^4$	0.03
USPS	380	20	2	2	780	256	100

Table 2.1: Data set statistics. The “unambig” column reports the number of multi-instances that are considered to be unambiguous. For multi-instance learning, these are the negatively labeled multi-instances. For semi-supervised learning and transductive inference, these are the labeled examples. All statistics for the three Corel data sets (Tiger, Elephant and Fox) as well as the seven TREC9 data sets have been averaged since the data sets are so similar.

Chapter 3

A learning framework

3.1 Background

Pattern classification addresses many of the questions that arise when studying the trends in a large collections of information, in particular, questions of association. Associations are mappings between *inputs* and *outputs*, and a rule for computing the mapping is called a *classifier*. Much attention in pattern classification research concerns the construction of classifiers in an automatic fashion in the context of supervised learning. To see how supervised methods can be generalized to ambiguous examples, some basic concepts and notation are required.

3.1.1 Inputs and outputs

First, it is assumed that inputs are encoded digitally in some fashion. Let $\mathcal{I} \in \mathbb{I}$ denote a raw encoding of the input. This encoding is captured using domain specific sensors such as a digital camera, a digital video camera, a mass spectrometer, etc.

A *preprocessing* stage converts the input $\mathcal{I}$ into a *feature vector* which is denoted by $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^d$. The space of feature vectors $\mathbb{X}$ is called the *input space* or *feature space*. The task of selecting meaningful features to represent objects is called *feature engineering*. The *similarity* of two objects is measured by comparing their feature vectors. When the input features span a vector space with a Euclidean metric, the inner product from this space may be used as a measure of similarity. It is also possible to use non-metric representations, and to define a measure of similarity by other means.

The output of the association is an integer indicating the category to which the object belongs. The set of possible labels is called the *output space* and is denoted $\mathbb{Y}$. Let $y \in \mathbb{Y}$ denote the label. Binary classification, where $\mathbb{Y}$ consists of two labels, is most prevalent. Since there are ways to reduce multi-class classification problems to the binary case, it is arguably of primary importance.

Definition 1. *An* example *of a concept is denoted* $(\mathbf{x}, y)$.

Figure 3.1 shows a collection of examples.

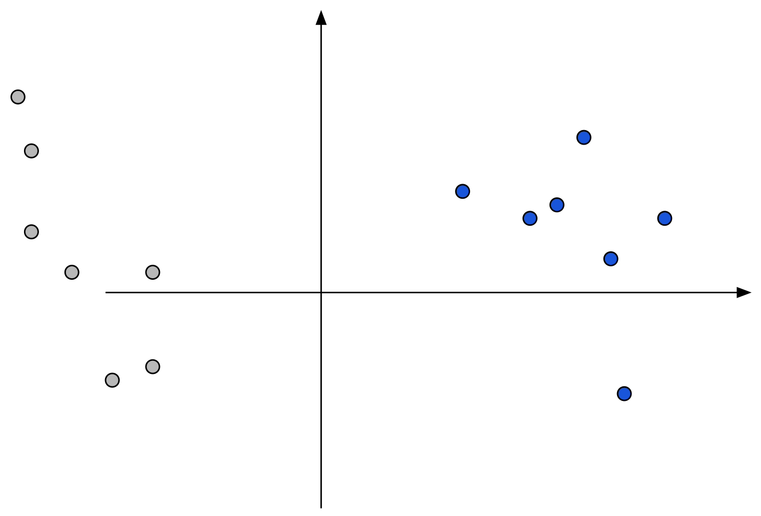


Figure 3.1: Examples $(\mathbf{x}, y)$ of a concept to be learned. The axes correspond to dimensions of the feature vectors $\mathbf{x}$. Color indicates the value of the binary label y for the inputs, where blue and grey indicate $y = 1$ and $y = -1$ respectively.

3.1.2 Supervised learning

Supervised learning refers to the general pattern classification problem of deriving a classifier $\hat{y} : \mathbb{X} \to \mathbb{Y}$ from a finite set of examples of input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}, \ i = 1, \ldots, m$. This is also known as *learning by example* and *inductive inference*. It is assumed that a subset of features is sufficient to discriminate classes to some reasonable degree.

Formally, examples are assumed to be independently and identically distributed according to a stationary but unknown probability distribution $(\mathbf{x}, y) \overset{i.i.d.}{\sim} p(\mathbb{X}, \mathbb{Y})$. This reflects the stochastic nature of the world. The goal then is to produce a classifier that will *generalize*; in other words predict the correct label $\hat{y}(\mathbf{x})$ for previously unseen inputs $\mathbf{x}$ in accordance with $p(\mathbb{X}, \mathbb{Y})$.

3.1.3 Risk minimization

Supervised learning algorithms are frequently derived from the principle of risk minimization [68]. For simplicity, this overview assumes that classifications are binary and misclassification costs are equal. The general idea is to select, from a large collection of candidates, a classifier that is expected to make few errors. For convenience of notation *in this section*, binary labels belonging to $\mathbb{Y} = \{+1, -1\}$ are assumed, with $+1$ corresponding to a true (T) logical truth value.

This thesis focuses on a family of threshold classifiers of the form

$$\hat{y}(\mathbf{x}) = \operatorname{sgn} f(\mathbf{x}) \tag{3.1}$$

where $f : \mathbb{X} \to \mathbb{R}$ is a real-valued *discriminant function*. Furthermore, it is typical to restrict the set of admissible discriminant functions $f \in \mathcal{F}$ to be *linear*

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{3.2}$$

parameterized by a weight vector $\mathbf{w}$ with $\|\mathbf{w}\| = 1$. The parameters $(\mathbf{w}, b)$ describe a hyperplane through the point $-b\mathbf{w}/\|\mathbf{w}\|^2 = -b\mathbf{w}$ that is perpendicular to $\mathbf{w}$.

A classification error occurs whenever the predicted label $\hat{y}(\mathbf{x}) = \operatorname{sgn} f(\mathbf{x})$ for an input $\mathbf{x}$ does not match the true label $y(\mathbf{x})$. To quantify the number of errors made by a classifier, the *zero-one loss* is defined as follows.

Definition 2. *The* zero-one loss $\mathcal{L} : \mathbb{Y} \times \mathbb{R} \to \mathbb{R}$ *is defined as*

$$\mathcal{L}(y, f(\mathbf{x})) = \begin{cases} 1 & yf(\mathbf{x}) < 0 \\ 0 & \textit{otherwise} \end{cases} . \tag{3.3}$$

Using this notation, the expected number of errors made by a classifier $\hat{y}$ is quantified by the *risk*.

Definition 3. *The* risk *associated with the classifier is defined as*

$$\mathcal{R}_p(f) = \int \mathcal{L}(y, f(\mathbf{x}))\, dp \tag{3.4}$$

where the integral is evaluated over the assumed but unknown joint probability distribution $p(\mathbb{X}, \mathbb{Y})$.

Figure 3.2 shows two examples of classifiers that obtain minimal (zero) risk on the given examples.

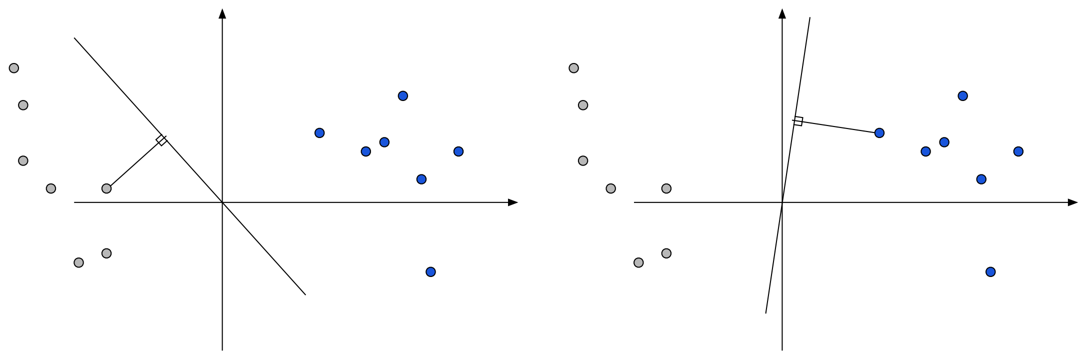


Figure 3.2: Low risk separating hyperplanes. Shown are two linear classifiers that obtain the minimal (zero) risk. Lines perpendicular to the hyperplanes indicate examples with the minimal functional margin.

The risk can be approximated using the training examples by a quantity called the *empirical risk*

Definition 4. *The* empirical risk *associated with the classifier is defined as*

$$\mathcal{R}_{\mathcal{D}}(f) = \sum_{i=1}^{m} \mathcal{L}(y_i, f(\mathbf{x}_i)) . \tag{3.5}$$

Unfortunately, minimizing the empirical risk alone does not guarantee that the resulting classifier will generalize. The problem is that the resulting classifier may perform well only on the training data, and not in general. To obtain a classifier that generalizes with high probability, it is necessary to also minimize its *capacity*. In order to define the capacity, the concept of margins is introduced.

Definition 5. *The* functional margin *of an* **unambiguous** training example *is defined by*

$$\gamma\left(\mathbf{x}, y\right) = \gamma_f\left(\mathbf{x}, y\right) = yf\left(\mathbf{x}\right) \ . \tag{3.6}$$

The subscript f is usually omitted to simplify notation. However, the margin is always understood to be defined in terms of the discriminant function $f\left(\mathbf{x}\right)$.

The functional margins of two selected points are displayed in Figure 3.2. A positive margin value for an example $\left(\mathbf{x}, y\right)$ indicates that the loss for the example is zero; i.e. it is correctly classified by the threshold classifier. For linear classifiers, the functional margin is proportional to the distance from the point $\mathbf{x}$ to the hyperplane described by $\left(\mathbf{w}, b\right)$. This distance is referred to as the geometric margin.

Definition 6. *For linear classifiers, the* geometric margin *of an* **unambiguous** training example *measured with respect to the p-norm $\left\|\cdot\right\|_p$ is the following value*

$$\frac{\left|\left\langle \mathbf{w}, \mathbf{x}\right\rangle + b\right|}{\left\|\mathbf{w}\right\|_q} \ . \tag{3.7}$$

Definition 7. *The* overall functional margin *for the* training set *is defined by*

$$\gamma\left(\mathcal{D}\right) = \gamma_f\left(\mathcal{D}\right) = \min_{1 \le i \le m} \gamma_f\left(\mathbf{x}_i, y_i\right) \ . \tag{3.8}$$

For a given margin γ, the capacity or Vapnik-Chervonenkis (VC) dimension h of the set of discriminant functions $\mathcal{H}\left(\gamma\right) = \left\{f \in \mathcal{F} | \gamma_f\left(\mathcal{D}\right) = \gamma\right\}$ that obtain a margin $\gamma_f\left(\mathcal{D}\right) = \gamma$ is bounded

$$h \le \min\left(\frac{R^2}{\gamma^2}, d\right) + 1 \tag{3.9}$$

where R is the radius of a ball that contains all instances $\mathbf{x}$ in the training set $\mathcal{D}$. The bound on the VC dimension decreases as the margin increases.

Proposition 1. *[Vapnik][68] With probability $1 - \delta,\ \delta > 0$ that the true risk is bounded by the empirical risk plus the width of a confidence interval around this estimate*

$$\mathcal{R}_p\left(f\right) \le \mathcal{R}_{\mathcal{D}}\left(f\right) + \ \Phi\left(\frac{m}{h}, \delta\right) \ , \tag{3.10}$$

where the width of the confidence interval $\Phi\left(s, \delta\right)$ is an increasing function of its first argument.

In this way, the width of the confidence interval $\Phi\left(s, \delta\right)$, and hence the bound on the risk $\mathcal{R}_p\left(f\right)$, decrease as the margin γ increases. Details (uniform convergence of means to their expectations, entropy and growth functions, Chernoff bounds, law of large numbers, ...) are not covered. Other bounds apply. See [68].

The empirical risk is often zero, especially when the training sets is small. For supervised learning, therefore, the confidence interval Φ of Equation (3.10) is most significant. The key observation is that the width of the confidence interval can be controlled through the overall margin γ.

Definition 8. Structural Risk Minimization (SRM) *The optimal discriminant function $f \in \mathcal{F}$ is the one that maximizes the minimum functional margin of examples over the training set $\gamma_f(\mathcal{D}) = \min_{1 \le i \le m} \gamma_f(\mathbf{x}_i, y_i)$.*

Figure 3.3 shows the maximum-margin separating hyperplane for the given examples.

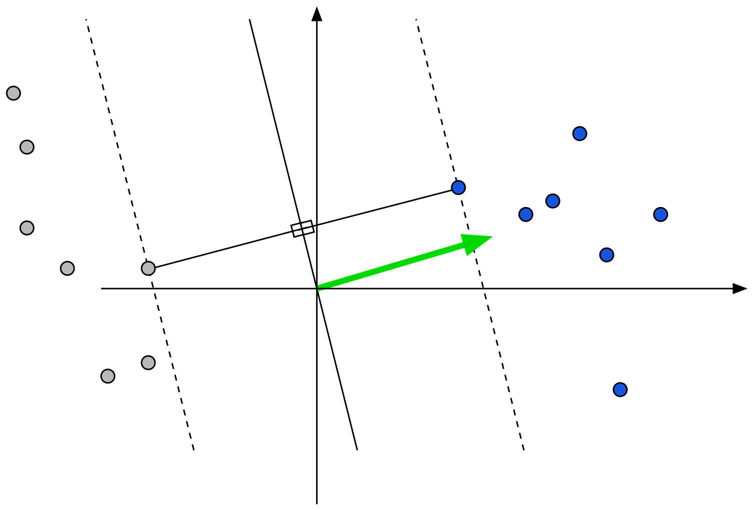


Figure 3.3: Maximum margin separating hyperplane. The dashed lines in this figure are used to emphasize the margin.

3.1.4 Maximum-margin separating hyperplanes

The maximum-margin separating hyperplane is the solution of an optimization problem defined in terms of a training data set. The input is a set of examples $\mathcal{D} = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, m\}$, where $\mathbf{x} \in \mathbb{X}$ and the labels are binary $y \in \mathbb{Y} = \{-1, 1\}$. The maximum-margin separating hyperplane is a linear discriminant function $f \in \mathcal{F}$ that robustly separates two categories in feature space $\mathbb{X} \subseteq \mathbb{R}^d$. It is the hyperplane that minimizes the overall margin. By the risk minimization principle, this results in a classifier having a bounded generalization error.

Definition 9. *The* maximum-margin separating hyperplane *is defined by the following optimization problem.*

$$\textit{Max-Margin} \qquad \max_{\mathbf{w}, b} \gamma(\mathcal{D}) \tag{3.11}$$

$$\textit{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \ge \gamma(\mathcal{D}), \forall i, \tag{3.12}$$

$$\|\mathbf{w}\| \le 1\,. \tag{3.13}$$

The weight vector is restricted to unit length, since $f \in \mathcal{F}$, ensuring that the objective function is bounded. The norm $\|\cdot\|$ used to bound $\mathbf{w}$ has a significant impact on the problem, and the algorithm used to solve it. The hyperplane defined by parameters $(\mathbf{w}, b)$ *separates* the data set $\mathcal{D}$ if the margin quantities $\gamma(\mathbf{x}_i, y_i)$ are all positive; in other words $\gamma(\mathcal{D}) > 0$. A $\mathcal{D}$ is called *linearly separable* when there exists a hyperplane that *separates* $\mathcal{D}$.

3.2 Dealing with ambiguity

3.2.1 Polymorphic input representations

Ambiguous signals occur frequently in pattern classification applications. As discussed in Chapter 1 and 2, in several pattern classification applications it is most convenient to work with inputs that are ambiguous due to polymorphism. In some cases, the raw inputs are polymorphic themselves, consisting of multiple digital encodings from one or more sensors of the same modality. Otherwise, polymorphism is the result of a preprocessing stage specifically designed to decompose or sample from the raw input. The first step to dealing with these inputs is a preprocessing step that converts the raw input $\mathcal{I}$ into a suitable form.

Definition 10. *A* multi-instance *is* a set *of feature vectors*

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \in 2^{\mathbb{X}} \,. \tag{3.14}$$

The number of members K may vary across inputs. The notation $2^{\mathbb{X}}$ denotes the power set of $\mathbb{X}$; and $\mathbf{X} \in 2^{\mathbb{X}}$ implies that $\mathbf{x} \in \mathbb{X}$ for all the multi-instance members $\forall \mathbf{x} \in \mathbf{X}$. The task of selecting a meaningful polymorphic representation is called *multi-instance engineering*. This step parallels the feature engineering described in Section 3.1.1. Unlike some related work [32, 34], this framework does not define a measure of similarity between multi-instances. Instead, only the similarities among the members of a multi-instance are used.

Individual instances $\mathbf{x}_1, \ldots, \mathbf{x}_K$ have labels $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_K)$ where $y(\mathbf{x}_k) \in \mathbb{Y}$, however these are unobserved. These labels model a first-order predicate of the form $y(\mathbf{x})$ =“is $\mathbf{x}$ an instance of *category*”. In this discussion, a binary output space $\mathbb{Y} = \{T, F\}$ is implied.

Definition 11. *A* multi-instance label Y *models a first-order predicate of the form* $Y(\mathbf{X})$ *=“does* $\mathbf{X}$ *contain an instance of* category*”.*

Assuming T/F labels, a multi-instance label can be expressed as a disjunction

$$Y(\mathbf{X}) = \bigvee_{\mathbf{x} \in \mathbf{X}} y(\mathbf{x}) \tag{3.15}$$

over the member instance labels.

Definition 12. *The rule for computing the multi-instance label given the labels of the members is called the* label semantics*. The fundamental label semantics, contained in Equation (3.15), are called* disjunctive label semantics*, or* “1-of-N” label semantics*. The label semantics can be formalized by defining a first-order predicate that is used to compute a multi-instance label* Y *from the labels of members* $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_K)$*.*

Definition 13. *An* ambiguous example *is a pair consisting of a multi-instance* $\mathbf{X} \in 2^{\mathbb{X}}$ *and a label* Y *and is denoted* $(\mathbf{X}, Y)$*. It is understood that the label is related to the member instance labels according to an assumed label semantics.*

There is an *asymmetry* in the multi-instance labels due to the disjunction: if the training label $Y = T$ then *at least one* individual $y(\mathbf{x})$ must be T, while if the training label $Y = F$ then *all* individual labels

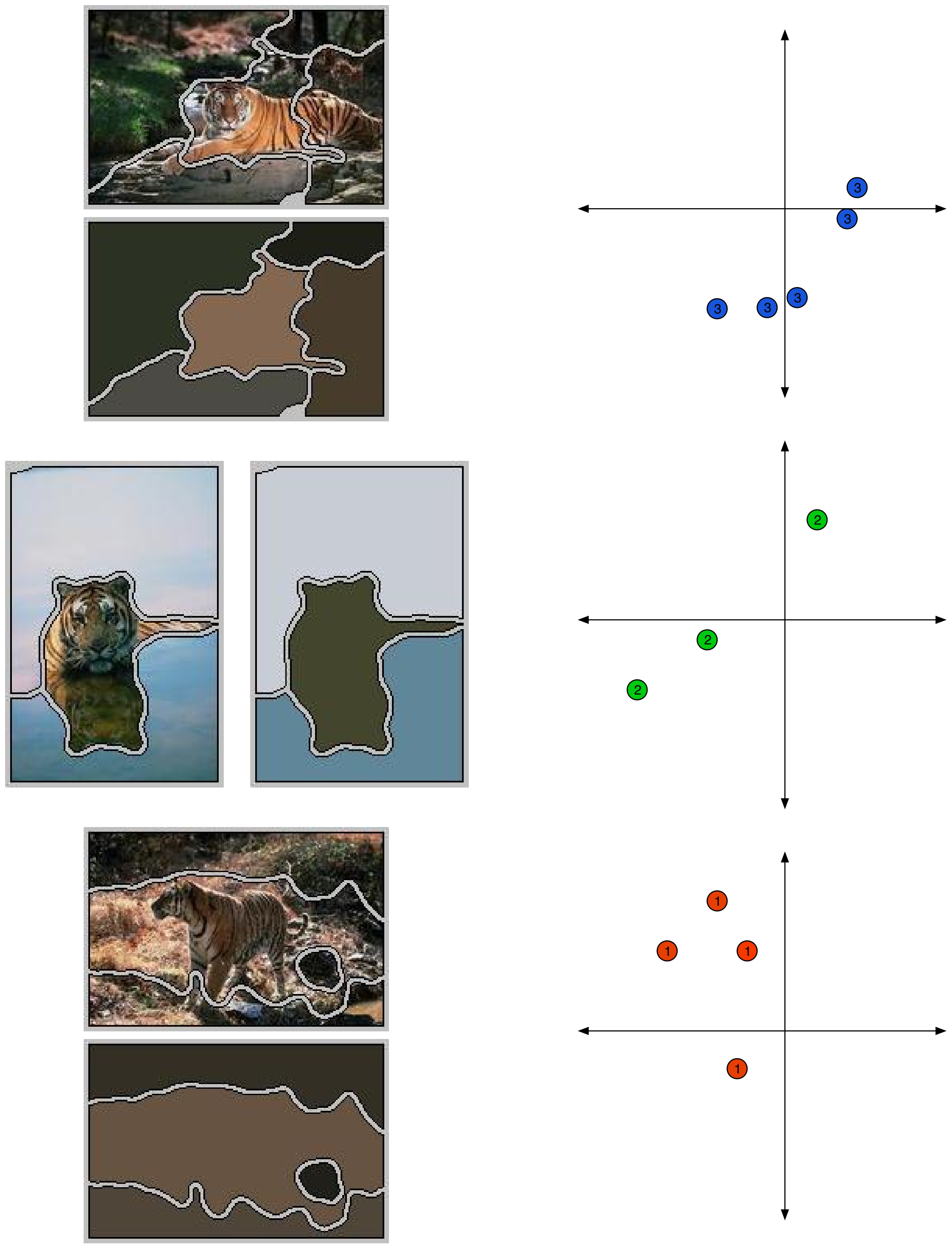


Figure 3.4: Polymorphism of image segmentation. Each image on the left is converted by way of segmentation, into the multi-instance on the right. Two examples of poorly segmented tiger images have been included to emphasize the challenges of content-based image classification, and how learning is still possible. Feature vectors for segments are depicted in red, blue and green, according to image; henceforth, these will be referred to as the red, blue and green image. The constraint among segments of a "tiger" image is that at least one must contain the tiger. In other words, when $Y = T$ then the must be an $\mathbf{x} \in \mathbf{X}$ with $y(\mathbf{x}) = T$.

must be F. Viewed another way, the ambiguous example $(\mathbf{X}, T)$ has *several interpretations* $\{(\mathbf{x}, T) \,|\mathbf{x} \in \mathbf{X}\}$ as input-output pairs, and at least one must be valid, while the example $(\mathbf{X}, F)$ has *several interpretations* $\{(\mathbf{x}, F) \,|\mathbf{x} \in \mathbf{X}\}$ as input-output pairs and all of them are valid. Due to this fact, multi-instances that are labeled F are sometimes considered to be unambiguous.

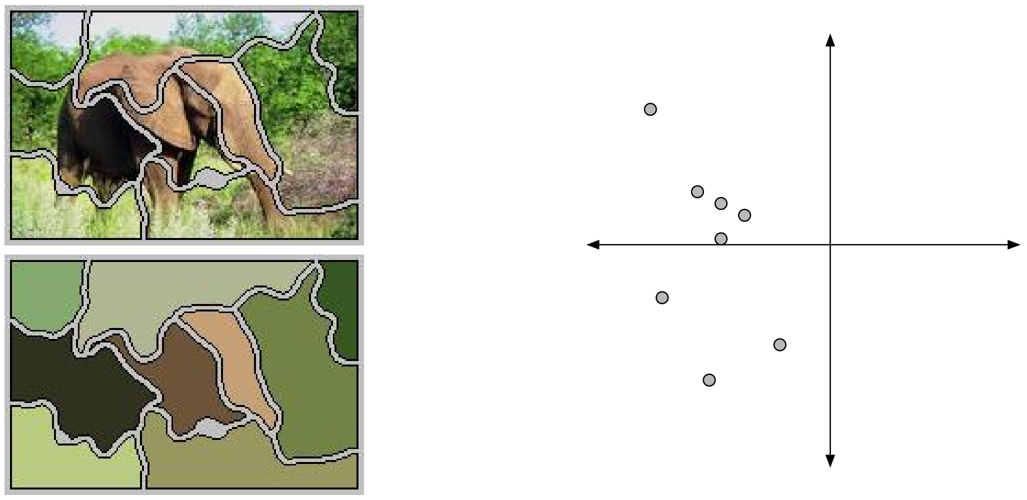


Figure 3.5: Asymmetry of multi-instance labels. When an image does not contain any tigers $Y = F$, then one can infer that all segments $\mathbf{x} \in \mathbf{X}$ can not be tiger segments, and therefore $y(\mathbf{x}) = F$. Each of the image segment feature vectors is colored grey to indicate that it is a non-tiger feature vector. This examples shows that polymorphism does not necessarily imply ambiguity. A segmented image comprises an ambiguous example only when it contains a tiger $Y = 1$.

Transductive and semi-supervised inductive inference deal with the problem of learning from examples, a subset of which are ambiguous because they are not labeled. There are two interpretations, $(\mathbf{x}, T)$ and $(\mathbf{x}, F)$, for unlabeled examples. If the classifier is defined by a hyperplane *through the origin*, then one can map unlabeled inputs to ambiguous examples $(\mathbf{X}, Y) \equiv (\{\mathbf{x}, -\mathbf{x}\}, T)$, reducing these problems to multiple-instance learning.

In general, learning from ambiguous examples can include multi-label and multi-class problems. Multi-class problems are classification problems where the outputs assume more that two different values. Multi-label problems are classification problems where the outputs can have several members. However, the current focus is on binary classification case.

3.2.2 Semi-supervised learning with ambiguous examples

Learning from ambiguous examples refers to the pattern classification problem of deriving a classifier $\hat{Y} : \mathbb{X} \to \mathbb{Y}$ from a finite set of *ambiguous examples* of input-output pairs $\mathcal{D} = \{(\mathbf{X}_i, Y_i) \,|i = 1, \ldots, m\}$. Note that the domain of the classifier is $\mathbb{X}$ and not $2^{\mathbb{X}}$. This semi-supervised learning problem encompasses *multiple-instance learning*, *semi-supervised inductive inference* and *transductive inference*. As described above, each of these problems can be treated as a multiple-instance learning problem. It is assumed that a subset of features is sufficient to predict labels of individual member instances and, using the 1-of-N label semantics, to predict labels of multi-instances with reasonable accuracy.

As before, examples are assumed to be independently and identically distributed according to a stationary but unknown probability distribution $(\mathbf{X}_i, Y_i) \overset{i.i.d.}{\sim} p\left(2^{\mathbb{X}}, 2^{\mathbb{Y}}\right)$, and the classifier should generalize in accordance with $p\left(2^{\mathbb{X}}, 2^{\mathbb{Y}}\right)$. Specifically, this means the classifier should predict individual instance labels $\hat{y}(\mathbf{x})\,, \ \forall \mathbf{x} \in \mathbf{X}$ and hence multi-instance labels $\hat{Y}(\mathbf{X}) = \bigvee_{\mathbf{x} \in \mathbf{X}} \hat{y}(\mathbf{x})$ consistent with the multi-instance label $Y = Y(\mathbf{X})$ for unseen examples from $p\left(2^{\mathbb{X}}, 2^{\mathbb{Y}}\right)$.

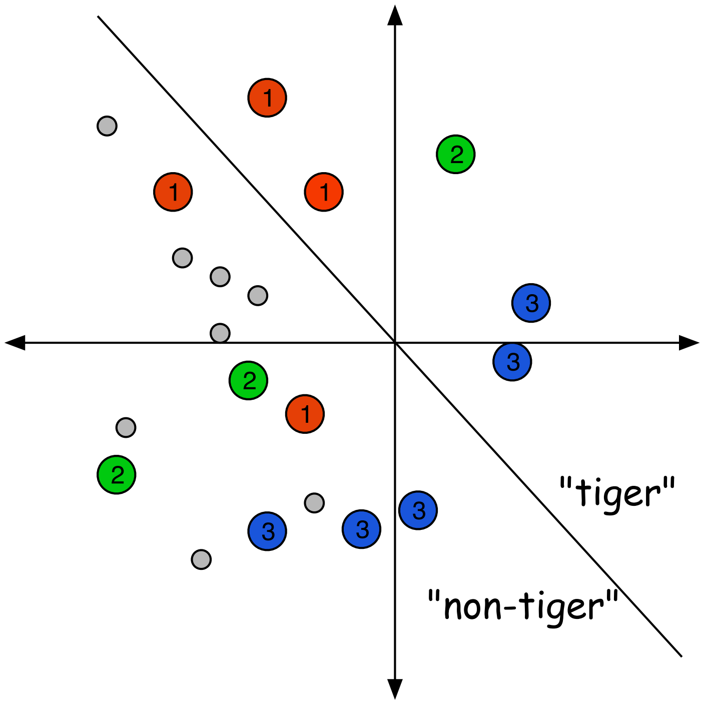


Figure 3.6: Learning from ambiguous examples. In order for a segment-level classifier $\hat{y}(\mathbf{x})$ to correctly classify each image according to the rule $\hat{Y}(\mathbf{X}) = \bigvee_{\mathbf{x} \in \mathbf{X}} \hat{y}(\mathbf{x})$, the classifier must ensure that at least one segment from each "tiger" image is classified as a "tiger" segment, and all segments from "non-tiger" images are classified as "non-tiger" segments. Assuming that the halfspace *above and to the right of the hyperplane* corresponds to tigers, this classifier succeeds. Two segments from the red image and the blue image are classified as tiger segments. Only one segment from the green image is classified as a tiger segment. All segments from the grey image are classified as non-tiger images. Each multi-instance labels is thus predicted correctly. This example emphasizes that it is possible to have more than one segment classified as a tiger segment. For example, when the quality of the image segmentation is poor, there can be multiple interpretations of the ambiguous example that are equally consistent with the remaining ambiguous examples.

3.2.3 Risk minimization with ambiguous examples

In the derivation of an optimal classifier through risk minimization, the fundamental difference between the supervised and semi-supervised learning problems is the method used to bound the *capacity* of the classifier to ensure generalization. In the supervised case, the capacity of the classifier is effectively controlled by the overall margin on the training data set. As defined in Equation (3.8), this is the minimum of the margins obtained on examples $(\mathbf{x}_i, y_i)\,,\ 1 \leq i \leq m$ from the training set.

For *ambiguous examples*, which have several interpretations as input-output pairs, the definition of the individual and overall margins must be refined. For the ambiguous example $(\mathbf{X}, T)$ valid interpretations are

$$\{(\mathbf{x}, T)\,|\mathbf{x} \in \mathbf{X}\} \tag{3.16}$$

each of which can be thought of as an unambiguous input-output pair. The process of selecting interpretations is called *disambiguation*. Once an interpretation is chosen, computing the maximum-margin hyperplane proceeds as in the supervised case. The margin for each example, as well as the overall margin, are computed using Equations (3.6) and (3.8). The problem is, the number of disambiguations of the training set $\mathcal{D} = \{(\mathbf{X}_i, Y_i)\,|1 \leq i \leq m\}$ is exponential in the number of instances. How does one disambiguate the training data - i.e. select a set of consistent interpretations - so that the resulting classifier generalizes? To answer this question, the SRM principle is used. The key intuition that is that the true interpretation of ambiguous inputs is self-reinforcing.

Multiple-instance learning assumes T/F labels. For convenience of notation in the sequel, binary labels belonging to $\mathbb{Y} = \{+1, -1\}$ are assumed, with $+1$ corresponding to a (T) logical truth value. An expression for the multi-instance label that applies to ± 1 labels and is equivalent to Equation (3.15) is

$$Y(\mathbf{X}) = \max_{\mathbf{x}\in\mathbf{X}} y(\mathbf{x})\ . \tag{3.17}$$

A family of threshold classifiers of the form $\hat{y}(\mathbf{x}) = \operatorname{sgn} f(\mathbf{x})$ is assumed, where $f : \mathbb{X} \to \mathbb{R},\ \ f \in \mathcal{F}$ is a *linear discriminant function* of the form $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}\rangle + b$ and $\|\mathbf{w}\| = 1$.

The classifier $\hat{y}(\mathbf{x})$ for individual instances can be used to predict multi-instance labels, using Equation (3.17),

$$\hat{Y}(\mathbf{X}) = \max_{\mathbf{x}\in\mathbf{X}} \hat{y}(\mathbf{x}) = \max_{\mathbf{x}\in\mathbf{X}} \operatorname{sgn} f(\mathbf{x})\ . \tag{3.18}$$

The "most positive instance" has a significant role in Equation (3.18), since alone, this member of the multi-instance is sufficient to determine the multi-instance label by thresholding. An equivalent expression for the multi-instance label that emphasizes the role of the most-positive instance is obtained by switching the order of the *maximum* and *sign* functions

$$\hat{Y}(\mathbf{X}) \quad = \quad \operatorname{sgn} \max_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})\ . \tag{3.19}$$

By virtue of taking the maximum operation, the implicit disambiguation of the ambiguous examples $(\mathbf{X}, Y)$ is encoded in Equation (3.19). The "most positive" instance $\mathbf{x} = \operatorname{argmax}_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$ yields the interpretation $(\mathbf{x}, T)$ used for disambiguation. This choice of interpretations is justified by the principle of structural risk minimization below.

Using these definitions, a modified *zero-one loss* is defined.

Definition 14. *The multi-instance loss $\mathcal{L} : \mathbb{Y} \times \mathbb{R} \to \mathbb{R}$ quantifies the number of errors made by a classifier $\hat{y}(\mathbf{x})$ on multi-instance label predictions*

$$\mathcal{L}\left(Y, \max_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})\right) = \begin{cases} 1 & Y \max_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}) < 0 \\ 0 & \textit{otherwise} \end{cases}\ . \tag{3.20}$$

Note that the errors are the result of predicting incorrect labels $\hat{Y}(\mathbf{X})$ for ambiguous examples $(\mathbf{X}, Y)$ using Equation (3.19).

Also using the above definitions, one can express the risk and empirical risk, as in Equations (3.4) and (3.5). Finally, the functional margin and the overall functional margin can be defined for ambiguous examples.

Definition 15. *The* functional margin of an **ambiguous** example *is*

$$\gamma(\mathbf{X}, Y) = \gamma_f(\mathbf{X}, Y) = Y \max_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x}) \tag{3.21}$$

Definition 16. *The* overall functional margin for the training set *is defined by*

$$\gamma(\mathcal{D}) = \gamma_f(\mathcal{D}) = \min_{1 \le i \le m} \gamma_f(\mathbf{X}_i, Y_i) \ . \tag{3.22}$$

The SRM principle adapted for learning from ambiguous examples is as follows:

Definition 17. Structural Risk Minimization for Ambiguous Examples, or Multi-Instance Structural Risk Minimization (MI-SRM) *The optimal discriminant function $f \in \mathcal{F}$ is the one that maximizes the minimum functional margin of ambiguous examples over the training set $\gamma(\mathcal{D}) = \min_{1 \le i \le m} \gamma(\mathbf{X}_i, Y_i)$.*

To justify this use of the MI-SRM principle to the problem of learning from ambiguous examples, observe that the optimal discriminant function $f(\mathbf{x})$ obtains the largest functional margin (lowest capacity). The functional margin of the optimal discriminant function $f(\mathbf{x})$ under the chosen interpretation is greater than that obtainable by any discriminant function $f \in \mathcal{F}$ under any other interpretation of the ambiguous examples. This implies that, after disambiguation of the training data set, the classifier $\hat{y}(\mathbf{x})$ has the lowest possible expected error bound. Therefore, MI-SRM selects the classifier $\hat{y}(\mathbf{x})$ that yields the lowest possible expected error on multi-instances.

A similar justification was used in large margin methods for transductive inference [39, 25]. In the subsequent text, the term margin will be used interchangeably with the term functional margin.

3.2.4 Maximum-margin separating hyperplanes with ambiguous examples

As before, the maximum-margin separating hyperplane is defined as the solution to an optimization problem defined with respect to a training data set. The input is a set of examples $\mathcal{D} = \{(\mathbf{X}_i, Y_i),\ i = 1, \dots, m\}$, where $\mathbf{X} \in 2^{\mathbb{X}}$ and the labels are binary $Y \in \mathbb{Y} = \{-1, 1\}$. The maximum-margin separating hyperplane is a linear discriminant function $f \in \mathcal{F}$ that robustly separates two categories in feature space $\mathbb{X} \subseteq \mathbb{R}^d$. It is defined as the hyperplane that minimizes the overall margin on ambiguous examples. By the structural risk minimization principle, this results in a classifier having a bounded generalization error.

Definition 18. *The* maximum-margin separating hyperplane *for the problem of learning from ambiguous examples is defined by the following optimization problem*

MIL Max-Margin
$$\max_{\mathbf{w}, b} \gamma(\mathcal{D}) \tag{3.23}$$
s.t.
$$Y_i \max_{\mathbf{x} \in \mathbf{X}_i} (\langle \mathbf{w}, \mathbf{x} \rangle + b) \ge \gamma(\mathcal{D}), \ \forall i, \tag{3.24}$$
$$\|\mathbf{w}\| \le 1 \, . \tag{3.25}$$

This is essentially the same optimization problem as in Equations (3.11 -3.13). The only difference is the substitution of the multi-instance margin, from Equation (3.21). The weight vector is restricted to unit length, since $f \in \mathcal{F}$, ensuring that the objective function is bounded. The norm $\|\cdot\|$ used to bound $\mathbf{w}$ has a significant impact on the algorithm used to solve it. The hyperplane defined by parameters $(\mathbf{w}, b)$ *separates* the data set $\mathcal{D}$ if the margin quantities $\gamma\left(\mathbf{X}_i, Y_i\right)$ are all positive; in other words $\gamma(\mathcal{D}) > 0$. A data set $\mathcal{D}$ is *MI-linearly separable* when there exists a hyperplane that *separates* all ambiguous examples in $\mathcal{D}$.

3.3 Further considerations

The goal of engineering a polymorphic representation from a given input is to best capture the implied meaning of the multi-instance label. Clearly, domain knowledge is required. As mentioned, techniques such as decomposition, segmentation or sampling can be used. However, there is a significant degree of freedom for a designer to explore. The engineering of polymorphic representations within this framework can take place independent of labeling, in particular after the labeling has been performed, and need not be performed by the same person.

As a concrete example, consider the automatic image annotation application. A designer of polymorphic representations for images can experiment with several segmentation algorithms. Each segmentation algorithm will generate a potentially unique polymorphic representation for the training data set. With a set of fixed image-level labels, the multi-instances become ambiguous examples of the visual concept. After training with each respective set of ambiguous examples, the performance of the resulting classifier on a validation set can be used to select among the segmentation algorithms.

The framework for learning from ambiguous examples also allows one to explore alternative label semantics, i.e. *interpretations* of the polymorphic ambiguity other than 1-of-N from Equation (3.15). Recall that the label semantics capture an assumed relationship between unobserved instance-level labels and the label that was provided. Alternative, first-order predicates can be used to represent multi-part models satisfying inter-part logical and spatial relationships. One caveat concerning multi-part models is that the computations involved can become prohibitively expensive due to the higher-order interactions. As noted in [57, 27], the ability to learn disjunctive concepts is at the core of the difference between attribute-value learning and inductive logic programming.

As a concrete example, one could use 2-of-N label semantics to recognize objects that have replicated parts. For these semantics, computations involving all pairs of members from a multi-instance are considered.

3.4 A look ahead

This chapter has outlined a framework for learning from ambiguous examples based on risk minimization. In the framework, a set of unambiguous input-output pairs is inferred from the ambiguous training set via a process called disambiguation. Disambiguation is encoded implicitly by selecting the “most positive” instance in each multi-instance. In this way, the principle of structural risk minimization drives the selection of the disambiguation to obtain the best bound on the generalization error of the classifier. The maximum-margin hyperplane is characterized by an optimization problem.

The primary challenge of learning from ambiguous examples is the disambiguation of ambiguous examples to support inductive learning. In the following chapters, four algorithms for learning from ambiguous examples are proposed and compared. The goal of each algorithm is the solution to the maximum-margin hyperplane problem. Variations in the formulation here depend on the norm used to define the margin and to bound the weight vector.

The first algorithm, multi-instance support vector machines (MI-SVM), proposed in [4], uses the 2-norm to bound the weight vector. This results in an optimization with a quadratic objective function and non-linear constraints. Integer variables are introduced to represent the selection of an interpretation. This results in a challenging mixed-integer quadratic optimization problem, and an efficient greedy local search algorithm is derived. Despite the lack of guaranteed global convergence, the algorithm significantly outperforms a baseline method on several benchmarks. Because the approach is general, it works equally well for document categorization and automatic image annotation. Later in the thesis, this algorithm is modified to operate with the 1-norm formulation.

The second algorithm, disjunctive programming boosting (DPBoost), proposed in [2], uses the 1-norm to bound the weight vector, so the resulting optimization problem has a linear objective function and non-linear constraints. The problem is approximated by replacing the non-linear feasible region with a larger convex set that is derived using lifting techniques from disjunctive programming. This yields a large linear program; where the size is related to the quality of the approximation. The benefit of this approach is convexity. This implies that global optimization can be used, and a unique optimal solution can be found. In this way, local minima are avoided at the expense of solving an approximation to the original problem. While the approximation is useful itself, it also helps to characterize the quality of true feasible points of the original problem. Unfortunately, the algorithm had limited applicability due to technical difficulties with the software used in the implementation in conjunction with the complexity of the algorithm itself.

A third algorithm, lift-and-project boosting (LNPBoost), proposed in [3], also uses the 1-norm bound the weight vector. This approach was motivated by the need to limit the size and management difficulties of the linear program in DPBoost. The lifted formulation used in DPBoost has a worst case space complexity that is quartic in the size of the data set. The difference in the third approach is the technique used to approximate the feasible region by a convex set. Instead of using lifting, a sequential compact representation of the convex hull is computed resulting in an approximate solution to the convex program. A feature selection algorithm extends the technique. This algorithm can be applied to large, high-dimensional data sets, and outperforms MI-SVM.

In order to carefully analyze the effectiveness of LNPBoost, a 1-norm formulation of MI-SVM is introduced. This shares many properties with the 2-norm version mentioned above but uses a linear objective.

Chapter 4

Related work

4.1 Definition

Learning from multi-instance examples was first introduced by [27] to address the drug-activity prediction problem (see Section 2.2), although others [40] had considered similar problems even earlier. Their multi-instance MUSK data sets have become an official benchmark for this problem. The authors of [27] present three learning algorithms which construct classifiers based on axis-parallel rectangles (APR) in feature space. The results obtained by these algorithms are among the most competitive on the MUSK data sets today.

4.2 Learning theory

After the multi-instance problem was conceived, a number of learning theoretic works were published on the topic. These included results that showed the problem was well-defined. For example, in [45], [5] and [19], theoretical arguments are given for the PAC-learnability of multi-instance concepts. Under the assumption that member instances are independent, a polynomial time algorithm is provided [45]. However, as shown in [5], without this assumption learning may be NP-hard.

4.3 Learning algorithms

Many supervised learning algorithms have been extended to deal with multi-instance inputs. In some cases, the modularity of these learning algorithms makes the extension to multi-instances relatively straight-forward, since only the components that deal with input features need to be changed. A number of algorithms for learning from multi-instances are discussed in the sequel. The algorithms are grouped into two sections, according to the type of model (conditional or joint probability density) from which they are derived.

4.3.1 Conditional models

The algorithms in the section are based on approximations to conditional probability density functions of a label given the input pattern. These algorithms generate classifiers for multi-instances that are not directly applicable to the individual instances.

In [71], two variants of the K-NN algorithm were developed. The algorithm is modified by substituting the Hausdorff metric, which measures the distance between sets of points (multi-instances), in place of the metric used to measure distance between individual points (instances). The Hausdorff distance between two sets of points is defined in terms of the pairwise distances between individual points. The application to multi-instances is immediate. In later work [34], a generalization of the Hausdorff metric is used as well.

In [79], the authors extend decision trees. They do so by defining entropy and coverage functions for multi-instances instead of individual instances.

Neural network were first extended to the multi-instance setting by [58]. In their approach, the network outputs obtained on the individual instances of each multi-instance are combined using a differentiable function that approximates the maximum of its inputs. Another application of neural network appears in [76]. The authors propose a modification of the error functional that takes into account the 1-of-N semantics of bags. No effort is made to achieve differentiability of the global error function.

More recently, [32, 33] developed a multi-instance kernel (similarity metric) and applied it to support vector machine learning. The similarity of multi-instances is defined in terms of the similarities of all possible pairs of members. One advantage of this approach is that research into kernel methods is very active and advanced.

Finally, in [30, 31], the authors propose a wrapper method used to extend an existing learning algorithm which 1) accepts instance weights, and 2) computes estimates of conditional label probabilities for individual instances. They express the conditional label probability of a multi-instance as the average of conditional label probabilities of member instances. Weights on the instances are used to balance the influence of individual member instances across all training examples. Their approach uses a symmetric treatment of positive and negative multi-instances. Using modifications when necessary, they apply this approach to extend support vector machines, decision trees, adaboost, logistic regression, nearest neighbor, and naive bayes learning algorithms. Of these, their method is best suited to logistic regression and Adaboost [31].

In [42], a Bayesian approach is adopted to learn a conditional label model. The conditional label model is expressed in terms of a sparse kernel expansion. Hidden variables are used for the member instance labels. A prior distribution over model parameters is specified and used to integrate out the uncertainty in the model parameters in the conditional model. The prior on member instance labels is truncated so that the only instantiations of labels with non-vanishing probability are those in accordance with the 1-of-N semantics of multi-instance labels. They employ a Metropolised blocked Gibbs sampler to approximate the otherwise intractable integral.

4.3.2 Joint models

A separate line of research concerns the modeling of joint probability density functions.

Early work by [49, 48] learn a concept that consists of a single target point in feature space. They define an "inverse generative" probability model through the specification of a likelihood of this target point over feature space that conditions on the training data. The so-called diverse density is largest for points that are close to at least one member instance from each positive multi-instance and far from all member instances from all negative multi-instances. They use Bayes rule and the assumption that multi-instance inputs are independent. A noisy-or model is used to express the 1-of-N semantics probabilistically. A gradient based optimization procedure is used to find the solution.

Extending this work, [75, 74] derive EM-DD. They introduce hidden variables to indicate which member instance is responsible for the multi-instance label in accordance with the 1-of-N semantics. They use the EM algorithm to jointly maximize over the hidden variables and the target concept. The E-step selects member instances that are most likely given the model. The M-step optimizes the diverse density given the selection of member instances made in the E-step.

In the computer vision area of object recognition, the multi-instance problem appears in a probabilistic setting where a hidden variable indexes the member instances of a multi-instance input. The authors of [29] define a generative probability model for an image, in terms of the appearance and relative location statistics of several interesting points. The image probability is expressed as a marginal over a hidden assignment variable h assigning keypoints from the image to foreground or background parts of the model. The likelihood ratio used for classification is expressed as a summation over h as well. Expectation maximization is used during learning, since the assignment h for each image is not observed.

4.4 Generalizations

In [57], the author argues that the multi-instance learning problem is the "natural frontier" between classical attribute-value learning and inductive logic programming (ILP). This is because all ILP problems can be converted into multi-instance problems, but typically can not be converted to attribute-value learning problems in a practical manner. For this reason, the multi-instance learning problem is viewed as an important extension of attribute-value learning.

In [64][67], the 1-of-N semantics of the multiple-instance learning problem are generalized to m-of-N semantics. The target concept is composed of a set of points. A multi-instance is labeled positively if and only if there is a subset of m target points that are each close to some point of the multi-instance. Furthermore, a set of repulsion points are used. None of the points in a positive bag are allowed to be located near the repulsion points. The Winnow algorithm used for learning a linear threshold encoding the concept. In [72], three further generalizations of the 1-of-N semantics are considered, including presence-based, threshold-based, and count-based multi-instance and multi-part concepts.

Several researchers have used multi-instance learning to solve ILP problems. The problems addressed can be significantly more general than that considered in the multi-instance setting, and include multi-part problems and multi-relational problems [23, 79, 22, 56, 54, 50]. In the domain of relational learning, a class of operations known as *aggregations* are defined to convert multi-instances into individual instances [56].

4.5 Related work on consistency

The notion of *consistency* has been considered in several semi-supervised learning contexts. In transductive learning, a subset of examples have labels associated with them, while others do not and are therefore ambiguous. The task is to infer labels for the unlabeled examples. The problem is called semi-supervised due to the incomplete specification of the training examples. The notion of a *consistent disambiguation* was used in this context by [39], and [14, 25]. In both cases, the theoretical justification is *structural risk minimization*. In [39], the author presents the underlying learning theory: the definition of hypothesis equivalence classes, the construction of a structure of increasing VC-dimension, the selection of an element from this structure via the maximization of a separating margin quantity over hyperplane and disambiguation parameters, and the relevance this has on the generalization error.

In [39], a mixed-integer formulation is presented that extends the quadratic formulation of the support vector machine. A heuristic rule for updating the unknown labels is interleaved with the update of hyperplane parameters using a support vector machine package. While it is shown that the algorithm converges after a finite number of iterations, it is only guaranteed to find local minima of the objective. Tests are performed on three text data sets.

A mixed-integer formulation based on the support vector machine is proposed in [14, 25]. This problem is solved to optimality for small data sets using global optimization techniques for integer-programming (branch-and-cut). However, branch-and-cut was unable to solve large-scale problems typical in machine learning. To deal with larger problem sizes, a localized version of the algorithm is developed. Performance is compared on several benchmark data sets. An additional non-convex formulation was considered and a heuristic algorithm was presented. Results were not conclusive in this case.

4.6 Combinatorial optimization

The combinatorial optimization community has dealt with optimization over integer variables, calling it integer programming. Branch-and-bound and branch-and-cut methods have previously been developed. However, the techniques used in combinatorial optimization were not originally developed to solve large-scale machine learning problems. As such, they do not always scale to meet the needs of the problems of interest to the machine learning community. For example, enumerative techniques were shown to be quite limited as shown in [14]. This is due to the large number of integer variables that are required. However, these authors did not explore this direction further. It is possible that a branch-and-bound, or branch-and-cut technique could be carefully modified to derive a suitable approximate solution to the problem of learning from ambiguous examples of moderate size.

Column generation and cutting-plane methods have also been adapted to suit the needs of the machine learning community. For example, the linear programming boosting algorithm of [26] is based on the idea of column generation. Another example is the relaxed online max-margin algorithm (ROMMA). This technique is a supervised learning algorithm due to [44] that is an adaptation of a cutting-plane algorithm. What makes this algorithm interesting is that it is an online learning algorithm in which the model parameters consist of a simple hyperplane and the training data set is processed sequentially. Hence, only a small subset of the data set is stored in memory at one time. As such, this technique can learn from data sets with millions of data points; a number much larger than the number of integer variables 1000-10000 used in benchmarking combinatorial optimization algorithms [18].

Disjunctive programming provides a framework for solving 0-1 programs in the area combinatorial optimization [9, 7]. Aspects of this theory are adapted in this thesis to address learning from ambiguous examples.

Chapter 5

MI-SVM

This chapter describes an algorithm, proposed in [4], for computing a maximum-margin separating hyperplane. It is called *multi-instance support vector machines* (MI-SVM). Unique to its development is the selection of the 2-norm that is used to bound the weight vector, which is naturally complemented by the 2-norm that is used to measure the width of the margin.

For unambiguous examples, the separating hyperplane problem has a quadratic objective function and linear constraints. For ambiguous examples, it has a quadratic objective function and non-linear constraints. Integer variables are introduced to represent the selection of an interpretation. This results in a challenging mixed-integer quadratic optimization problem, and a local search algorithm is derived. Nevertheless, when compared on the standard multi-instance benchmark data sets, the algorithm frequently and significantly outperforms a baseline approach. The approach works equally well document categorization and automatic image annotation.

After reviewing the 2-norm formulation of the maximum-margin separating hyperplane problem for unambiguous examples, the extension to ambiguous examples is described, a greedy local search algorithm is proposed, and experiments are presented.

5.1 Introduction

A brief review of the maximum-margin separating hyperplane formulation is included for clarity and easy reference. Focused readers who have good recall may wish to advance to Subsection 5.1.1.

As described in Chapter 3, Section 3.1.4, the objective of maximum-margin learning is a linear discriminant function that robustly separates two categories in feature space $\mathbb{X} \subseteq \mathbb{R}^d$. The theoretical foundation of maximum-margin learning is based on risk minimization. In this context, a bound on the generalization performance of the resulting classifier is derived. The bound depends on a measure of class separation by the linear discriminant function known as the *margin.*

The learning algorithm takes as input a set of examples $\mathcal{D} = \{(\mathbf{x}_i, y_i)\,,\ i = 1, \dots, m\}$, where $\mathbf{x} \in \mathbb{X}$ and the output labels are binary $y \in \mathbb{Y} = \{-1, 1\}$. The maximum-margin separating hyperplane was described by the optimization problem in Equations (3.11-3.13) and is re-stated here for clarity.

$$\text{Max-Margin} \qquad \max_{\mathbf{w},b} \gamma(\mathcal{D}) \tag{5.1}$$

$$\text{s.t.} \quad y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) \geq \gamma(\mathcal{D}), \forall i, \tag{5.2}$$

$$\|\mathbf{w}\| \leq 1\,. \tag{5.3}$$

5.1.1 2-Norm max-margin

This section outlines the development of an algorithm for solving a specific instantiation of the maximum-margin separating hyperplane problem, where the 2-norm $\|\cdot\|_2$ is used to bound the norm of the weight vector. For a vector $a = (a_1, \ldots, a_n)$, the 2-norm is defined as follows

$$\|a\|_1 = \left(\sum_{1\leq j\leq n} a_j^2\right) .$$

If the 2-norm of the weight vector $\mathbf{w}$ equals a constant, then the *functional margin* of a point $\mathbf{x}$ is proportional to the distance between the point $\mathbf{x}$ and the closest point $\mathbf{p}$ on the hyperplane $\langle \mathbf{w}, \mathbf{p}\rangle = 0$ measured with respect to the 2-norm $\|\cdot\|_2$ [46]. The quantity $\frac{\langle \mathbf{w}, \mathbf{x}\rangle + b}{\|\mathbf{w}\|_2}$ is called the *signed geometric margin measured with respect to the 2-norm*.

Because $\|\mathbf{w}\|_2 = 1$ for $f \in \mathcal{F}$, the optimization problem in Equations (5.1-5.3) is maximizing the minimal *geometric margin* measured with respect to the 2-norm. As shown in [46], there is an equivalent formulation in which the norm of the weight vector is minimized, while constraining all *functional margins* to be greater than or equal to 1. This is the so-called primal SVM formulation. It is derived by way of variables $\dot{\mathbf{w}} = \mathbf{w}/\gamma(\mathcal{D})$ and $\dot{b} = b/\gamma(\mathcal{D})$. Notice, in this case, that $\|\dot{\mathbf{w}}\|_2 = \frac{\|\mathbf{w}\|_2}{\gamma(\mathcal{D})}$ and therefore minimizing $\|\dot{\mathbf{w}}\|_2$ is equivalent to maximizing the margin subject to $\|\mathbf{w}\|_2 = 1$.

The revised expression for the maximum-margin separating hyperplane is as follows

$$\text{2-Norm SVM} \qquad \min_{\dot{\mathbf{w}},\dot{b}} \frac{1}{2}\|\dot{\mathbf{w}}\|_2^2 \tag{5.4}$$

$$\text{s.t.} \quad y_i\left(\langle \dot{\mathbf{w}}, \mathbf{x}_i\rangle + \dot{b}\right) \geq 1, \forall i. \tag{5.5}$$

The constraints of the primal are called *margin constraints*. The overall margin quantity $\gamma(\mathcal{D})$ can be recovered from $\|\dot{\mathbf{w}}\|_2$. It is evident that the problem is a quadratic programming problem, since the objective is quadratic and the constraints are linear. In the sequel, the dots over variables $\left(\dot{\mathbf{w}}, \dot{b}\right)$ are omitted. Specialized software for solving this structured quadratic programming problem has been developed [38].

Figure 5.1 demonstrates the connection between the shortest weight vector satisfying the margin constraints, and the maximum-margin separating hyperplane.

5.1.2 Non-separable case

If the constraints of the SVM primal formulation are infeasible, then there is no plane with positive margin $\gamma(\mathcal{D}) > 0$. To address this problem, a degree of flexibility is coded in the margin constraints by introducing

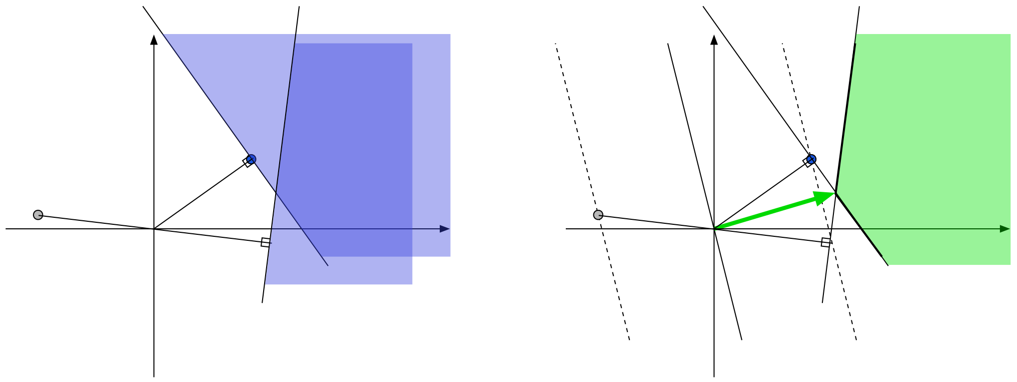


Figure 5.1: Geometry of 2-Norm SVM. In this example, there are only two training examples with $y = 1$ and $y = -1$ (blue and grey respectively). For simplicity, the bias term b is omitted. The two margin constraints correspond to the two shaded halfspaces on the left side. The shortest weight vector, measured with respect to the 2-norm, that satisfies both margin constraints must lie in the intersection of these regions, as shown on the right side.

soft-margins. An example $(\mathbf{x}_i, y_i)$ may violate the margin constraint by an amount $\xi_i \geq 0$, called the *slack*, at the expense of adding a penalty proportional to ξ_i to the objective. The modified primal SVM formulation is thus

$$\begin{aligned} &\text{2-Norm SVM} && \min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^{m} \xi_i && (5.6) \\ &\text{(Soft Margins)} \quad \text{s.t.} && y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1 - \xi_i,\ \forall i, && (5.7) \\ & && \xi_i \geq 0,\ \forall i. && (5.8) \end{aligned}$$

5.1.3 Feature mappings

In Section 3.1.3 Equation 3.2, the set of admissible discriminant functions $f \in \mathcal{F}$ is restricted to be *linear* in terms the components of $\mathbf{x} \in \mathbb{X}$. While this restriction may seem severe, the restriction to linear classifiers is independent of how the feature space $\mathbb{X}$ is defined and can be circumvented as described next.

A common technique used to maximize the generalization ability of linear classifiers involves mapping the inputs from the initial feature space $\mathbb{X} \subseteq \mathbb{R}^d$ to a higher-dimensional space $\mathbb{X}' \subseteq \mathbb{R}^n$ using a non-linear function

$$\mathbf{x} \mapsto \mathbf{x}' = \Phi\left(\mathbf{x}\right) \ . \qquad (5.9)$$

The dimension n of the image is typically much larger than the dimension of the original feature space. To include pairwise variable interactions, for example, the following feature mapping is used

$$\Phi\left(\mathbf{x}\right) = \left(x_1^2, x_2^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_1x_d, \sqrt{2}x_2x_3, \ldots, \sqrt{2}x_{d-1}x_d, 1\right) \ .$$

The feature mapping is designed to facilitate large margin separation of the training data using a hyperplane $(\mathbf{w}', b')$ in the high-dimensional feature space parameterized by a weight vector $\mathbf{w}' \in \mathbb{R}^n$ and offset b'.

The additional dimensions will also increase the margin obtained by the maximum-margin separating hyperplane. In turn, the principle of risk minimization can be used to obtain a better bound on the generalization performance of the resulting classifier. The optimization problem characterizing the maximum-margin separating hyperplane is written in the same form as shown in Equations (5.6-5.8) with $\mathbf{x}', \mathbf{w}'$ replacing $\mathbf{x}, \mathbf{w}$ respectively.

The linear classifier $f(\mathbf{x}') = \operatorname{sgn}(\langle \mathbf{w}', \mathbf{x}' \rangle + b')$ defined by the maximum-margin separating hyperplane $(\mathbf{w}', b')$ from $\mathbb{X}'$, can be composed with the non-linear function $\Phi(\mathbf{x})$ to define an equivalent, non-linear classifier $(f \circ \Phi)(\mathbf{x})$ on the original feature space $\mathbb{X}$. Therefore, an algorithm that solves Equations (5.6-5.8) for a linear classifier can in theory be generalized to learn non-linear classifiers using feature mappings.

5.1.4 Duality, support vectors & kernelization

Appealing to Lagrangian techniques for constrained optimization the primal SVM problem with soft margins is re-formulated in dual form. The advantages of the dual formulation are: 1) a simplified set of constraints, 2) a parameterization that is readily interpretable, and 3) an optimization problem that admits kernelization [24, 65].

The dual problem is expressed in terms of the dual variables α_i, one for each training example

$$\text{SVM-Dual} \qquad \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{5.10}$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \ \forall i, \tag{5.11}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0\,. \tag{5.12}$$

Kernelization is achieved by replacing the inner product $\langle \mathbf{x}_i', \mathbf{x}_j' \rangle = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ as follows

$$\text{SVM-Dual} \qquad \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{5.13}$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \ \forall i, \tag{5.14}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0\,. \tag{5.15}$$

The kernel emulates the computation of the mapping $\Phi(\mathbf{x})$ and subsequent inner product of the two inputs. In certain cases, there are closed-form expressions for the kernel value in terms of the original feature vectors $\mathbf{x}_i, \mathbf{x}_j$ that do not require the explicit computation of the feature mapping $\Phi(\mathbf{x})$. This is advantageous when the feature mapping is high-dimensional and thus the inner product is expensive to compute.

It is therefore easy to experiment with alternate feature mappings of the training data. Notice in Equations (5.13-5.15) that the kernel is the only term through which the features can influence learning. This makes it easy to experiment with different feature mappings; all one needs is to redefine the kernel function in accordance with a chosen $\Phi(\mathbf{x})$. More generally, even without knowing the feature mapping $\Phi(\mathbf{x})$, any binary function satisfying Mercer's condition [24, 65] can be used for the kernel, further increasing the flexibility of this approach.

Further analysis of the primal-dual equivalence may be used to demonstrate that, the solution of the dual SVM problem will be sparse; only a subset of the dual variables will be non-zero [24, 65]. Points for which $\alpha_i > 0$ in the solution are called the *support vectors*, since they define or "support" the hyperplane.

5.2 Dealing with ambiguity

This section outlines the MI-SVM formulation and algorithm as an extension of the 2-norm SVM that is solved by quadratic programming.

As described in Chapter 3, Section 3.2.4, the objective of maximum-margin learning *from ambiguous examples* is a linear discriminant function that robustly separates two categories in feature space $\mathbb{X} \subseteq \mathbb{R}^d$. The theoretical foundation of maximum-margin learning is based on risk minimization. In this context, a bound on the generalization performance of the resulting classifier is derived. The bound depends on a measure of class separation by the linear discriminant function known as the *multi-instance margin.*

The maximum-margin separating hyperplane was described by the optimization problem in Equations (3.23-3.25) and is re-stated here for clarity.

$$\text{MIL Max-Margin} \qquad \max_{\mathbf{w},b} \gamma\left(\mathcal{D}\right) \tag{5.16}$$

$$\text{s.t.} \quad Y_i \max_{\mathbf{x}\in\mathbf{X}_i} \left(\langle\mathbf{w},\mathbf{x}\rangle + b\right) \geq \gamma\left(\mathcal{D}\right), \ \forall i, \tag{5.17}$$

$$\|\mathbf{w}\| \leq 1\,. \tag{5.18}$$

This section outlines the formulation of an optimization problem for the maximum-margin separating hyperplane. Unique to this development is the use of the 2-norm used to bound the norm of the weight vector, which is complemented by the 2-norm measuring the width of the margin.

5.2.1 2-Norm max margin with ambiguous examples

In this section, the 2-norm $\|\cdot\|_2$ is used to bound the norm of the weight vector. As shown in [46], there is an equivalent formulation in which the norm of the weight vector is minimized, while constraining all *functional margins* to be greater than or equal to 1. Soft-margins are also introduced, allowing the ambiguous example $(\mathbf{X}_i, Y_i)$ to violate the multi-instance margin constraint, at the expense of adding a penalty proportional to ξ_i to the objective.

The revised expression for the maximum-margin separating hyperplane is the following quadratic programming problem with non-linear constraints, which is called MI-SVM

$$\textit{MI-SVM} \qquad \min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i \tag{5.19}$$

$$\text{s.t.} \quad Y_i \max_{\mathbf{x}\in\mathbf{X}_i} \left(\langle\mathbf{w},\mathbf{x}\rangle + b\right) \geq 1 - \xi_i, \ \forall i \tag{5.20}$$

$$\xi_i \geq 0, \ \forall i. \tag{5.21}$$

As discussed in Chapter 3, Section 3.2.3 a self-reinforcing disambiguation is encoded in the multi-instance margin constraint.

Negatively labeled multi-instances $(\mathbf{X}_i, Y_i)$ are treated differently because they are not ambiguous. This fact is manifest in Equation (5.20). When $Y_i = -1$, the non-linear multi-instance margin constraint can be replaced by several linear constraints

$$-\max_{\mathbf{x}\in\mathbf{X}_i} (\langle \mathbf{w}, \mathbf{x}\rangle + b) \geq (1-\xi_i) \quad \Longrightarrow \quad (\langle \mathbf{w}, \mathbf{x}\rangle + b) \leq -(1-\xi_i) \ , \ \forall \mathbf{x} \in \mathbf{X}_i \tag{5.22}$$

$$\Longrightarrow \quad -(\langle \mathbf{w}, \mathbf{x}\rangle + b) \geq 1-\xi_i \ , \ \forall \mathbf{x} \in \mathbf{X}_i \tag{5.23}$$

These constraints are tied together by a single slack variable ξ_i which complicates the implementation. For simplicity, negatively-labeled multi-instances are separated, and the instances are treated as though they came from independent unambiguous training examples $(\{\mathbf{x}\}, -1)$ where $\mathbf{x} \in \mathbf{X}_i$. This step introduces new slack variables into the objective. If necessary, the penalty factor C for the newly created slack variables can be scaled.

5.2.2 Mixed-integer formulation

Learning from ambiguous examples can be cast as a mixed-integer program MI-SVM. For positively-labeled multi-instances, an integer variable $1 \leq z(i) \leq |\mathbf{X}_i|$ is used to indicate the "most positive" member $\mathbf{x}_{z(i)} \in \mathbf{X}_i$. The multi-instance margin constraint from Equation (5.20) reduces to $\langle \mathbf{w}, \mathbf{x}_{z(i)}\rangle + b \geq 1-\xi_i$. Additional constraints are added to enforce the fact that $\mathbf{x}_{z(i)} \in \mathbf{X}_i$ is the "most positive" member. Optimization occurs over the additional integer variables.

$$\text{MI-SVM (mixed integer)} \qquad \min_{z} \min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \tag{5.24}$$

$$\text{s.t.} \quad Y_i\left(\left\langle \mathbf{w}, \mathbf{x}_{z(i)}\right\rangle + b\right) \geq 1-\xi_i, \ \forall i, \tag{5.25}$$

$$\left(\left\langle \mathbf{w}, \mathbf{x}_{z(i)}\right\rangle + b\right) \geq (\langle \mathbf{w}, \mathbf{x}\rangle + b), \forall \mathbf{x} \in \mathbf{X}_i \ \forall i, \tag{5.26}$$

$$\xi_i \geq 0, \ \forall i. \tag{5.27}$$

In this formulation, every positively-labeled multi-instance $\mathbf{X}_i$ is thus effectively represented by a single member instance $\mathbf{x}_{z(i)} \in \mathbf{X}_i$. Notice that the remaining instances have no impact on the objective. The derivations also hold for general kernel functions $K(\mathbf{x}_i, \mathbf{x}_j)$.

A simple example showing the maximum margin MI-separating hyperplane is illustrated, with explanation, in Figure 5.2.

5.2.3 Greedy local search heuristics

The following observation motivates the proposed greedy local search heuristics. If the currently selected "most positive" instance $\mathbf{x}_{z(i)}$ for the i-th ambiguous example participates in an *active* margin constraint, and yet the difference $\max_{1\leq k\leq K_i} f(\mathbf{x}_k) - f\left(\mathbf{x}_{z(i)}\right) > 0$, then the update $z(i) = \arg\max_{1\leq k\leq K_i} f(\mathbf{x}_k)$ will weaken the constraint and possibly decrease the objective (increase the margin). The condition is simple to evaluate. Notice that a decrease in the objective is not guaranteed since the remaining constraints in the problem may prevent the margin from growing. After making the update, however, the objective can be computed and compared to its previous value to determine whether the update was successful.

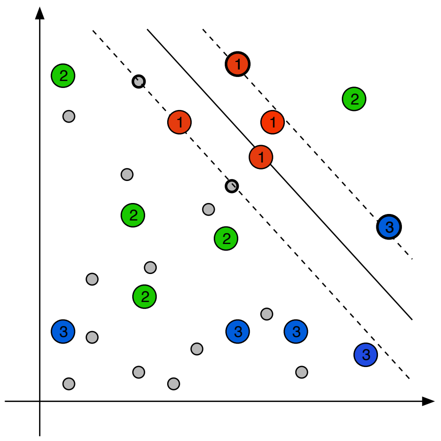


Figure 5.2: Maximum margin MI-separating hyperplane. The MI-separating hyperplane is the solid line while the margins are indicated with dotted lines. Members of positively-labeled multi-instances are displayed as large circles with numbers i, and colors, that encode multi-instance membership $\mathbf{x} \in \mathbf{X}_i$, while members of negatively-labeled multi-instances are displayed as small gray circles. The individual member instances chosen as support vectors are bold. Notice that *all* members of negatively-labeled multi-instances obtain a large margin in the negative halfspace [down and to the left], and *at least one* member from each positively-labeled multi-instance ($i = 1, 2, 3$) obtains a large margin in the positive halfspace [up and to the right].

A simple 2-stage local search heuristic alternates the following steps: (i) for given integer variables, solve the associated QP and find the optimal hyperplane, (ii) for a given hyperplane, update the integer variables in a way that (locally) minimizes the objective. For given integer variables $z\left(i\right)$, the problem reduces to a QP that can be solved exactly. The latter step may involve the simultaneous update of all integer variables $z\left(i\right)$ using the condition stated above, or the update of just a small number of integer variables, perhaps just one. These two update strategies will be called synchronous and asynchronous, respectively. The asynchronous algorithm selects integer variables $z\left(i\right)$ to be updated based on the size of the potential decrease to the objective, which is estimated using the difference: $\max_{1\leq k\leq K_i}\ f\left(\mathbf{x}_k\right) - f\left(\mathbf{x}_{z(i)}\right)$. Figure 5.3 demonstrates how the solution changes as integer variables are updated on a simple example.

Updates can be performed efficiently because the integer variables are decoupled given the hyperplane. Also notice that the QP-solver can be re-initialized at every iteration with the previously found solution to speed-up subsequent optimization. To initialize the scheme, one can set the witness $\mathbf{x}_{z(i)}$ equal to the centroid of the multi-instance $\mathbf{X}_i$. The 2-stage alternating search heuristic terminates when the integer variables reach a fixed-point, or when the value of the objective function reaches a local minimum. A local minimum is detected when the subsequent update to the integer variables results in an increase to the objective. Since the objective is bounded below by zero, and there are a finite number of disambiguations, this algorithm terminates in a finite number of steps. Algorithm 5.1 summarizes the algorithm using *synchronous updates* in pseudo-code.

The MI-SVM algorithm was implemented in C++ with the help of version 4.0 of the SVMlight solver [38]. In order to facilitate efficient restarting of the SVM, the solver required extensive modification. In particular, the dual variables from the previous solution were used for initialization.

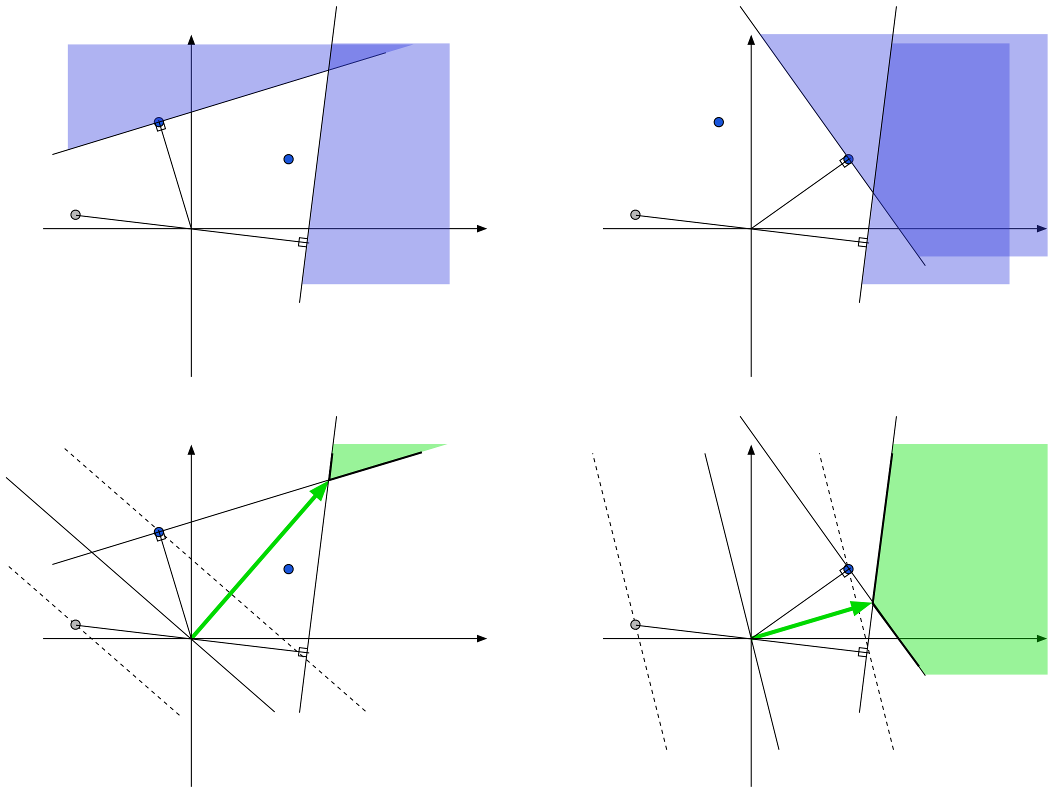


Figure 5.3: Progression of mixed-integer local search. In this example, there are only two training examples. The first training example is ambiguous with $Y = 1$ whose multi-instance representation consists of two blue points. The second example is unambiguous with $Y = -1$ and consists of one grey point. For simplicity, the bias term b is omitted.
In the first column, the upper most blue point is selected as the witness for the ambiguous example. The multi-instance margin constraint reduces to the halfspace constraint defined by this point. Then, with this setting of the integer variables, the shortest weight vector measured with respect to the 2-norm is as shown within the intersection of the two shaded halfspaces. Notice, that the instances from the ambiguous example are both classified positively.
In the next iteration, displayed in the second column, the witness is subsequently updated to reflect the current "most positive" member instance, it will switch to the right-most blue point. The result is a shorter weight vector and a wider margin. Only one of the instances from the ambiguous example is classified positively, which is sufficient according to Equation (5.20).

Algorithm 5.1 Pseudo-code for MI-SVM optimization heuristics (synchronous update).

```
1: **input**: training sample $\mathcal{D}$
2: initialize $\mathbf{x}_{z(i)} = \sum_{\mathbf{x} \in \mathbf{X}_i} \mathbf{x} / |K_i|$ for every positive input $\mathbf{X}_i$
3: **repeat**
4:     compute SVM solution $(\mathbf{w}, b)$ for disambiguated training examples $\left\{ \left( \mathbf{x}_{z(i)}, Y_i \right), \; 1 \leq i \leq m \right\}$
5:     compute outputs $f\left(\mathbf{x}_k\right) = \langle \mathbf{w}, \mathbf{x}_k \rangle + b$ for all member instances $\mathbf{x}_k \in \mathbf{X}_i, \;\; 1 \leq k \leq K_i$
6:     set $z\left(i\right) = \arg\max_{1 \leq k \leq K_i} \; f\left(\mathbf{x}_k\right) \; , \forall i$
7: **until** selector variables $z\left(i\right)$ do not change, or objective increases
8: **output**:$(\mathbf{w}, \; b)$ corresponding to local minima
```

5.2.4 An alternative mixed-integer formulation

The following algorithm was introduced together with MI-SVM in [4]. This algorithm was also derived within the disambiguation and learning framework. However, the notion of an *interpretation* of an ambiguous example is slightly more general. This affects the notion of disambiguation and its relation to the multi-instance margin.

The basic idea is to introduce explicit integer label variables $y(\mathbf{x}) = \pm 1$, $\forall \mathbf{x} \in \mathbf{X}$ for the unknown instance-level labels. These variables are related to $Y(\mathbf{X})$ via the label semantics from Equation (3.17). The main difference with this formulation is that all of the instances within a multi-instance can jointly participate in defining the hyperplane; there is no single "most positive" instance. Each interpretation of an ambiguous example involves a set of instance-level labels, one for every member, subject to the constraint that at least one guessed label must be positive. There are many more distinct interpretations in this formulation than considered with MI-SVM. For example, in the MI-SVM formulation, an ambiguous example $(\mathbf{X}, Y)$ with $Y = 1$ and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ has k interpretations as a *single* unambiguous example $\{(\mathbf{x}_i, Y) \,|\, i = 1, \ldots, k\}$. With mi-SVM, there are $2^k - 1$ interpretations each comprised of k unambiguous examples

$$\left\{\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_k, y_k)\} \,|\, (y_1, \ldots, y_k) \in \{\pm 1\}^k \text{ and } \max_i y_i = +1\right\}.$$

The revised expression for the maximum-margin separating hyperplane is the following mixed-integer quadratic programming, which is called mi-SVM

$$\textit{mi-SVM} \qquad \min_{y(\mathbf{x})} \min_{\mathbf{w}, b, \xi} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1}^{m} \xi_i \tag{5.28}$$

$$\text{s.t.} \quad y(\mathbf{x})\left(\langle \mathbf{w}, \mathbf{x}\rangle + b\right) \geq 1 - \xi_i, \ \forall i, \ \forall \mathbf{x} \in \mathbf{X}_i \tag{5.29}$$

$$\text{s.t.} \quad Y_i = \max_{\forall \mathbf{x} \in \mathbf{X}_i} y(\mathbf{x}), \ \forall i \tag{5.30}$$

$$\xi_i \geq 0, \ \forall i. \tag{5.31}$$

A two-stage greedy optimization procedure that is similar to MI-SVM was proposed in [4]. For initialization, all instance-level integer variables are set to $+1$. Results for this algorithm are included below.

5.3 Experiments

5.3.1 Methodology

Experiments were performed on MIL benchmarks to evaluate the proposed technique and compare it to other methods. Specifically, two MUSK data sets, in addition to three Corel image data sets (Tiger, Elephant, Fox) and seven TREC9 text data sets (1,2,3,4,7,9,10) were used. Details can be found in Chapter 2.

The reported results are from 10-fold crossvalidation experiments, as described in [35]. Each data set is split into 10 equal size partitions, where the size of a partition is based on the number of ambiguous examples it contains. Because the splitting is according to ambiguous examples, multi-instances are preserved. The training and testing data sets for each experiment are constructed using 9 partitions for training and 1 partition for testing so that each of the 10 partitions is used for testing in one of the folds. After training, the ambiguous examples were classified using the multi-instance classifier defined in Equation 3.19. The accuracy of the classifier was then measured as $\frac{\text{num correct}}{\text{num total}} = \frac{\left|\left\{i | \hat{Y}(\mathbf{X}_i) = Y(\mathbf{X}_i)\right\}\right|}{m}$; in other words, the accuracy is measured on multi-instance label predictions. The learning algorithm is run 10 times on the different training data sets, each time evaluating the accuracy on test set. Finally, the average of the 10 accuracies is reported.

Results are shown for the synchronous versions of mi-SVM and MI-SVM. Three kernels were used: linear $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, polynomial $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + S)^2$, and radial basis function (RBF) $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$. The parameters S and γ were coarsely optimized on a single fold on the MUSK1 data set, and the optimal parameters were then used for crossvalidation experiments with both MUSK1 and MUSK2 data sets. A similar coarse parameter-tuning step was used for the Corel image data sets, using a single fold from the Tiger data set, and for the TREC9 text data sets using a single fold from test set number 7.

As a reference method the expectation maximization diverse density method (EM-DD) of [75] was implemented. In their report, competitive results have been reported on the MUSK benchmark. However, the description seems to indicate that the test data was used to select the optimal solution obtained from multiple runs of the algorithm. In the pseudo-code formulation of EM-DD, D_i is used to compute the error for the i-th data fold, where it should in fact be $D_t = D - D_i$ (using the notation of [75]). The corrected version of the algorithm is used to generate the results reported here. The accuracies using the corrected code are more in line with previously published results.

The majority of methods in the literature do not specify the standard deviation of the crossvalidated accuracies, and so testing significance for these cases requires weaker significance tests or further implementation.

5.3.2 Drug activity prediction

The first reported results concern the drug activity prediction task using the ambiguous data sets that were introduced in Chapter 2, Section 2.2. Table 5.1 compares MI-SVM and mi-SVM with various kernels to EM-DD.

For both MUSK1 and MUSK2 data sets, mi-SVM with the RBF kernel achieves competitive accuracy values; in fact, for MUSK1 the improved performance over EM-DD is statistically significant. While MI-SVM with the RBF kernel outperforms mi-SVM on MUSK2, it is significantly worse on MUSK1. With linear and polynomial kernels, the performance is generally poor.

There has been much recent progress with multi-instance learning algorithms in recent years. Most algorithms are tested on the MUSK data sets. Table 5.2 summarizes cross-validation results for a selection. The performance of MI-SVM and mi-SVM do not clearly exceed any of these algorithms. However, MI-SVM and mi-SVM both output a classifier that can be applied to individual patterns as well as multi-instances. This property is not shared by the algorithms in Table 5.2. Another distinguishing feature of MI-SVM and mi-SVM is that they are applicable to a variety of domains as is demonstrated in the subsequent results.

Data Set Category	EM-DD	mi-SVM			MI-SVM		
		linear	poly	rbf	linear	poly	rbf
MUSK1	84.8	76.3↓	84.0	**87.4**↑	75.7↓	72.7↓	77.9↓
MUSK2	**84.9**	71.5↓	73.9↓	83.6	76.0↓	67.5↓	84.3

Table 5.1: Classification accuracy for the proposed methods as compared to EM-DD on the MUSK image data sets.

	DD-SVM [21]	Bagging-APR [77]	EM-DD [75]	MI-NN [58]	RELIC [62]	DD [49]	Multinst [5]	IAPR [27]
MUSK1	85.8	92.8	84.8	88.9	83.7	88.0	76.7	92.4
MUSK2	91.3	93.1	84.9	82.0	87.3	82.5	84.0	89.2

Table 5.2: Comparison of algorithms by 10-fold crossvalidated accuracy on the MUSK data sets.

5.3.3 Automatic image annotation

The next experiments concern the automatic image annotation task and corresponding ambiguous data sets that were introduced in Chapter 2, Section 2.1.

The results are summarized in Table 5.3. They show that both algorithms, mi-SVM and MI-SVM, generally achieve similar accuracies and outperform EM-DD by a few percent. While MI-SVM performed marginally better than mi-SVM, both heuristic methods were susceptible to other nearby local minima. Evidence of this effect was observed through experimentation with asynchronous updates, as described in Section 5.2.3. By changing the number of integer variables that are updated in each iteration, the algorithm will discover alternate local minima.

Note that the MI-SVM algorithm with the RBF kernel is significantly worse than EM-DD on the Elephant and Tiger data sets. Otherwise, the majority of results represent statistically significant improvements over EM-DD.

5.3.4 Text categorization

The next experiments concern the text categorization task and corresponding ambiguous data sets that were introduced in Chapter 2, Section 2.3.

Using linear and polynomial kernel functions, which are generally known to work well for text categorization, both methods show statistically significant improved performances over EM-DD in almost all cases.

Data Set Category	EM-DD	mi-SVM			MI-SVM		
		linear	poly	rbf	linear	poly	rbf
Elephant	78.3	**82.2**↑	78.1	80.0↑	81.4↑	79.0	73.1↓
Fox	56.1	58.2↑	55.2	57.9↑	57.8↑	**59.4**↑	58.8↑
Tiger	72.1	78.4↑	78.1↑	78.9↑	**84.0**↑	81.6↑	66.6↓

Table 5.3: Classification accuracy of different methods on the Corel image data sets.

Data Set Category	EM-DD	mi-SVM			MI-SVM		
		linear	poly	rbf	linear	poly	rbf
TST1	85.8	93.6↑	92.5↑	90.4↑	**93.9**↑	93.8↑	93.7↑
TST2	84.0	78.2↓	75.9↓	74.3↓	**84.5**	84.4	76.4↓
TST3	69.0	**87.0**↑	83.3↑	69.0	82.2↑	85.1↑	77.4↑
TST4	80.5	82.8	80.0	69.6↓	82.4↑	**82.9**↑	77.3↓
TST7	75.4	**81.3**↑	78.7↑	**81.3**↑	78.0↑	78.7↑	64.5↓
TST9	65.5	**67.5**↑	65.6	55.2↓	60.2↓	63.7↓	57.0↓
TST10	78.5	79.6↑	78.3	52.6↓	79.5↑	**81.0**↑	69.1↓

Table 5.4: Classification accuracy of different methods on the TREC9 document categorization sets.

On the other hand, with the RBF kernel, both MI-SVM and mi-SVM are somewhat unpredictable; sometimes significantly better and sometimes significantly worse than EM-DD. This could be due to issues of local minima. No significant difference between MI-SVM and mi-SVM is evident for the text classification task.

5.4 Discussion

This chapter presented a novel approach to multiple-instance learning based on a generalization of the maximum margin idea used in SVM classification. Although the formulation leads to a difficult mixed-integer optimization problem, even simple greedy local search heuristics already yield quite competitive results compared to the baseline approach, namely EM-DD. As far as the MIL research is concerned, this algorithm was the first to be successfully applied to a diverse collection of benchmarks from varied domains.

The subsequent chapters consider optimization techniques that can avoid unfavorable local minima.

Chapter 6

DPBoost

The second algorithm called disjunctive programming boosting (DPBoost), proposed in [2]. This algorithm differs from the mixed-integer approach presented in Chapter 5 primarily because the 1-norm is used to bound the weight vector instead of the 2-norm. As explained below, the 1-norm bound is complemented by the use of the ∞-norm for the measurement of the margin width.

For unambiguous examples, the separating hyperplane problem has a linear objective function and linear constraints, while for ambiguous examples, it has a linear objective function and non-linear constraints. In the later case, the problem is simplified by replacing the non-linear constraints with a larger convex set that is derived using lifting techniques from disjunctive programming. The resulting optimization problem is a very large linear program, and global optimization is used. Although the algorithm solves an approximation, bounds can be derived to evaluate the quality of approximation.

After reviewing the 1-norm formulation of the maximum-margin problem for supervised learning, the DPBoost method is described, and experiments are presented. Due to time and memory limitations imposed by the linear programming utilities that were used, these experiments were restricted to a small scale.

6.1 Introduction

A brief review of the maximum-margin separating hyperplane formulation is included for clarity and easy reference. Focused readers who have good recall may wish to advance to Subsection 6.1.1. As described in Chapter 3, Section 3.1.4, the objective of maximum-margin learning is a linear discriminant function that robustly separates two categories in feature space $\mathbb{X} \subseteq \mathbb{R}^d$. The theoretical foundation of maximum-margin learning is based on risk minimization.

The learning algorithm takes as input a set of examples $\mathcal{D} = \{(\mathbf{x}_i, y_i)\,,\ i = 1, \ldots, m\}$, where $\mathbf{x} \in \mathbb{X}$ and the output labels are binary $y \in \mathbb{Y} = \{-1, 1\}$. The maximum-margin separating hyperplane is characterized by the solution to the following optimization problem from Chapter 3, Equations (3.23-3.25).

$$\text{Max-Margin} \qquad \max_{\mathbf{w},b} \gamma(\mathcal{D}) \tag{6.1}$$

$$\text{s.t.} \quad y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq \gamma(\mathcal{D}), \forall i, \tag{6.2}$$

$$\|\mathbf{w}\| \leq 1\,. \tag{6.3}$$

6.1.1 1-Norm max-margin

This section outlines the development of an algorithm for solving a specific instantiation of the maximum-margin separating hyperplane problem, where the 1-norm $\|\cdot\|_1$ is used to bound the norm of the weight vector. For a vector $a = (a_1, \ldots, a_n)$, the 1-norm is defined as follows

$$\|a\|_1 = \sum_{1 \leq j \leq n} |a_j| \ ,$$

while the ∞-norm is defined as

$$\|a\|_\infty = \max_{1 \leq j \leq n} |a_j| \ .$$

If the 1-norm of the weight vector $\mathbf{w}$ equals a constant, then the *functional margin* of a point $\mathbf{x}$ is proportional to the distance between the point $\mathbf{x}$ and the closest point $\mathbf{p}$ on the hyperplane $\langle \mathbf{w}, \mathbf{p} \rangle = 0$ measured with respect to the ∞-norm $\|\cdot\|_\infty$ [46]. This distance is referred to as the geometric margin.

Definition 19. *For linear classifiers, the* signed geometric margin measured with respect to the ∞-norm *is the following value*

$$\frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|_1} \,. \tag{6.4}$$

Because $\|\mathbf{w}\|_1 = 1$ for $f \in \mathcal{F}$, the 1-norm SVM is maximizing the minimal *signed ∞-norm geometric margin*. As shown in [46], there is an equivalent formulation in which the norm of the weight vector is minimized, while constraining all *functional margins* to be greater than or equal to 1. It is derived by way of variables $\dot{\mathbf{w}} = \mathbf{w}/\gamma(\mathcal{D})$ and $\dot{b} = b/\gamma(\mathcal{D})$. Notice, in this case, that $\|\dot{\mathbf{w}}\|_1 = \frac{\|\mathbf{w}\|_1}{\gamma(\mathcal{D})}$ and therefore minimizing $\|\dot{\mathbf{w}}\|_1$ is equivalent to maximizing the margin subject to $\|\mathbf{w}\|_1 = 1$.

The revised expression for the maximum-margin separating hyperplane is as follows

$$\text{1-Norm SVM} \qquad \min_{\dot{\mathbf{w}},\dot{b}} \|\dot{\mathbf{w}}\|_1 \tag{6.5}$$

$$\text{s.t.} \quad y_i\left(\langle \dot{\mathbf{w}}, \mathbf{x}_i \rangle + \dot{b}\right) \geq 1, \forall i. \tag{6.6}$$

The constraints of the primal are called *margin constraints*. The overall margin quantity $\gamma(\mathcal{D})$ can be recovered from $\|\dot{\mathbf{w}}\|_1$. In the sequel, the dots over variables $\left(\dot{\mathbf{w}}, \dot{b}\right)$ are omitted.

It is evident that the problem is a linear programming problem, since the objective and constraints are linear. Linear programming software packages [37] are designed to solve this type of structured linear programming problem. In this context, a bound on the generalization performance of the resulting classifier has been derived [26]. The bound depends on a measure of class separation by the linear discriminant function known as the *margin.*

Note that the formulation is valid even when all training examples have the same label, and may be used for dimension reduction and novelty detection applications [59].

6.1.2 Non-separable case

In case no plane with positive margin $\gamma(\mathcal{D}) > 0$ exists, the constraints of the 1-norm SVM formulation can not be met. To address this problem, a degree of flexibility is coded in the margin constraints by introducing *soft-margins*. An example $(\mathbf{x}_i, y_i)$ may violate the margin constraint by an amount $\xi_i \geq 0$, called the *slack*, at the expense of adding a penalty proportional to ξ_i to the objective. The modified primal SVM formulation is thus

$$\begin{aligned} &\text{1-Norm SVM} && \min_{\mathbf{w},b} \|\mathbf{w}\|_1 + C\sum_{i=1}^{m} \xi_i && (6.7)\\ &\text{(Soft Margins)} \quad \text{s.t.} && y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1 - \xi_i,\ \forall i, && (6.8)\\ & && \xi_i \geq 0,\ \forall i. && (6.9) \end{aligned}$$

6.1.3 Feature mappings

As in Section 5.1.3, in order to to maximize the generalization ability of linear classifiers, the inputs are transformed to a high-dimensional feature space $\mathbb{X}' \subseteq \mathbb{R}^n$

$$\mathbf{x} \mapsto \mathbf{x}' = \left(h_1\left(\mathbf{x}\right), \ldots, h_n\left(\mathbf{x}\right)\right) \ . \tag{6.10}$$

The notation differs from that of Section 5.1.3. Here, the mapping is defined in terms of its component functions $h_k\left(\mathbf{x}\right)$ which are called basis functions. For example, one common set of basis functions, sometimes called an *empirical kernel map*, are defined using radial basis functions centered on the training instances $h_k\left(\mathbf{x}\right) = \exp\left(-\left\|\mathbf{x} - \mathbf{x}_k\right\|^2 / \sigma^2\right)$, $\forall k$, where $\mathbf{x}_k$ is the k-th instance in the training data set, and $\sigma \in \mathbb{R}$ is a tunable parameter. The feature mapping is designed to facilitate large margin separation of the training data using a hyperplane $(\mathbf{w}', b')$ in the high-dimensional feature space parameterized by a weight vector $\mathbf{w}' \in \mathbb{R}^n$ and offset b'. The additional dimensions will also increase the margin obtained by the maximum-margin separating hyperplane, and the principle of risk minimization can be invoked to obtain a better bound on the generalization performance of the resulting classifier.

The weight vector is $\mathbf{w} \equiv (\alpha_1, \ldots, \alpha_n) \in \mathbb{X}'$ is parameterized by variables $\alpha_k \in \mathbb{R}$, $1 \leq k \leq n$, which is a notation borrowed from ensemble learning techniques [63]. In the transformed space, a linear discriminant function *without bias* term takes the following form

$$f\left(\mathbf{x}\right) = \langle \mathbf{w}', \mathbf{x}' \rangle = \sum_{k=1}^{M} \alpha_k h_k\left(\mathbf{x}\right) \ . \tag{6.11}$$

Notice that the discriminant function is expressed as an *ensemble*. The basis functions $h_k\left(\mathbf{x}\right)$ are called weak hypotheses and the coefficients α_k are called combination weights. The problem variables including $\alpha_k \geq 0$ are constrained to be positive, as is the convention of linear programming [55, 15]. This is not a restriction assuming that the set of basis functions h_k is closed under negation. This also allows the 1-norm of $\mathbf{w}'$ to be expressed as $\sum_{k=1}^{M} \alpha_k$.

The resulting 1-norm SVM formulation takes the following form

$$\text{1-Norm SVM} \qquad \min_{\alpha,\xi} \sum_{k=1}^{M} \alpha_k + C \sum_{i=1}^{m} \xi_i \tag{6.12}$$

$$\text{(Feature Map)} \quad \text{s.t.} \quad y_i \sum_{k=1}^{n} \alpha_k h_k(\mathbf{x}_i) + \xi_i \geq 1 \ \forall i, \tag{6.13}$$

$$\alpha_k, \xi_i \geq 0 \ \forall k, i \ . \tag{6.14}$$

Since the number of weak hypotheses $h_k(\mathbf{x})$ is large, the1-norm SVM requires the solution of a potentially large linear program. Unlike the 2-norm SVM presented in Chapter, the 1-norm SVM formulation does not admit kernelization. This is because the technique of kernelization is founded upon the Representer Theorem [70], which requires that the 2-norm be used as the bound on the weight vector.

Therefore, this chapter assumes that an explicit feature mapping is defined through $h_k(\mathbf{x})$. If a feature mapping is not required, then the basis functions are assumed to be projection functions that select individual components of the feature vector $\mathbf{x}$.

6.1.4 Geometric interpretation

To gain insight into the geometric structure of the feasible region of 1-norm SVM, it is useful to introduce set notations that correspond to constraints in Equations (6.13) and (6.14). Let

$$\mathcal{Q} = \{(\alpha, \xi) : \ \alpha, \xi \geq 0\} \tag{6.15}$$

denote the positive quadrant, and

$$H_i \equiv \left\{ (\alpha, \xi) : \ y_i \sum_k \alpha_k h_k(\mathbf{x}_i) + \xi_i \geq 1 \right\} \tag{6.16}$$

denote the region of feasibility corresponding to the i-th example. Using these notations, the feasible region is expressed

$$(\alpha, \xi) \in \mathcal{Q} \cap \bigcap_{i=1}^{m} H_i \ . \tag{6.17}$$

The feasible region is therefore a convex region defined in a space with dimensions for each α_k, $k = 1, , n$, and ξ_i, $i = 1, \ldots, m$. The intersection of this feasible region with a 2-dimensional subspace is depicted in the top-right of Figure 6.1. The remaining images in Figure 6.1 contrast the 1-norm and 2-norm formulations when trained on a simple example with only two examples.

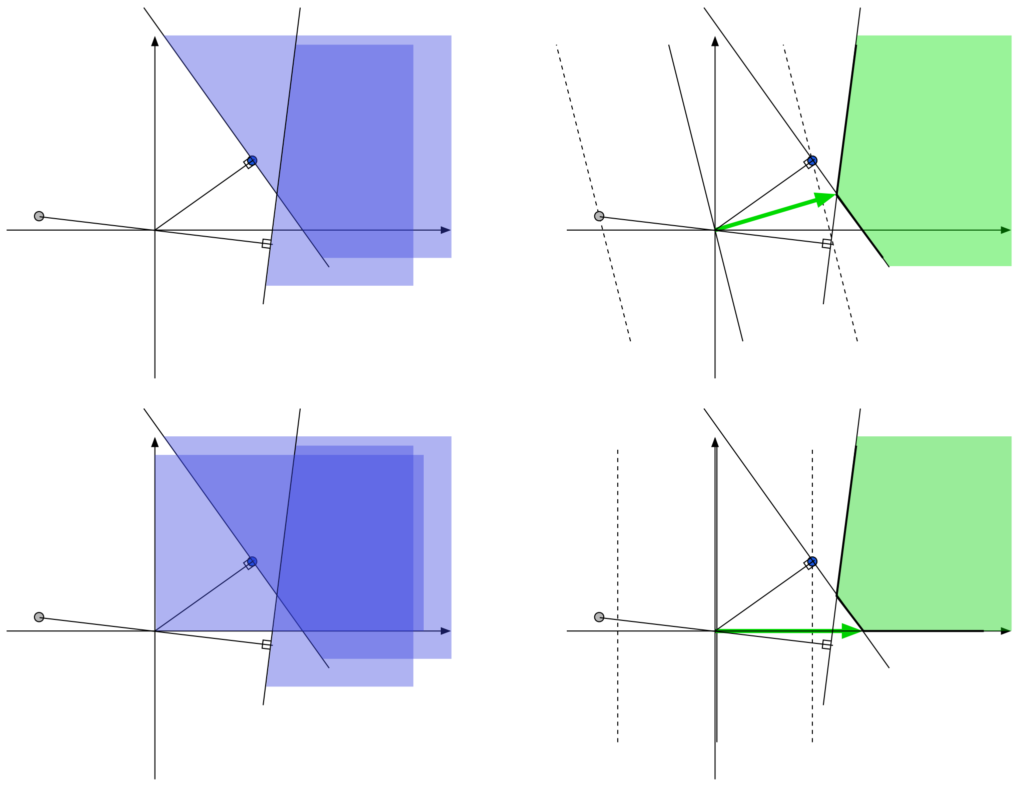


Figure 6.1: Comparison of 1-Norm and 2-norm SVM. In this small example, there are only two training examples with $y = 1$ and $y = -1$ (blue and grey circles respectively). For simplicity, assume that the bias term b is fixed at zero, so all hyperplanes go through the origin, and maximum-margin separating hyperplanes are simply the shortest weight vectors that are feasible. In the 2-norm case, the feasible region is the intersection of these halfspaces depicted in the top-left. The top-right image shows the 2-norm solution, depicted as a green vector. Dotted lines shown in this image correspond to the margin. Notice that both points lie outside the margin on either side of the hyperplane. For the 1-norm, the feasible region is restricted to the positive quadrant and hence is the intersection of the three regions depicted in the bottom-left figure. It turns out that this is not a fundamental restriction on the 1-norm solution, because the collection of features is typically closed under negation (see text). The green vector in the bottom-right image is the shortest vector measured with respect to the 1-norm. Observe that the 2-norm solution has 2 non-zero coefficients (w_1, w_2), while 1-norm solution has only one $(w_1, 0)$.

6.1.5 Dual formulation

To characterize the solution of the LP, it is useful to consider the dual problem and conditions of complementarity as in [26]. The dual of the optimization problem in Equations (6.12-6.14) can be written as

$$\max_{u} \quad \sum_{i=1}^{m} u_i \tag{6.18}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} u_i y_i h_k(\mathbf{x}_i) \leq 1 \quad \forall k \tag{6.19}$$

$$0 \leq u_i \leq C \quad \forall i \tag{6.20}$$

In [26], the dual variables u_i are described as *misclassification costs*.

The solutions of the primal and dual problems are depend closely on one another. This can be seen by analyzing the the conditions of complementarity [55] relating these two solutions. These conditions are as follows

$$u_i \left(y_i f(\mathbf{x}_i) + \xi_i - 1\right) = 0 \tag{6.21}$$

$$\alpha_k \left(\sum_{i=1}^{m} u_i y_i h_k(\mathbf{x}_i) - 1 \right) = 0\,. \tag{6.22}$$

Since the optimal values of the slack variables are implicitly determined by α as $\xi_i(\alpha) = [1 - y_i f(\mathbf{x}_i)]_+$, the first set of conditions states that the misclassification costs are zero $u_i = 0$ whenever $y_i f(\mathbf{x}_i) > 1$. On the other hand, examples with non-vanishing misclassification costs $u_i > 0$ must have with tight margin constraints. The second set of conditions ensures that $\alpha_k = 0$ if $\sum_{i=1}^{m} u_i y_i h_k(\mathbf{x}_i) < 1$, while on the other hand, weak hypotheses with non-zero weights α_k must satisfy $\sum_{i=1}^{m} u_i y_i h_k(\mathbf{x}_i) = 1$. Thus, a typical ensemble solution may be sparse in two ways: 1) only a small number of weak hypothesis with $\alpha_k > 0$ will contribute to the ensemble, and 2) the solution will depend only on a subset of training examples with $u_i > 0$.

6.1.6 A boosting algorithm using column generation

The simplex method [55] for solving LP maintains a *tableau* representing the current *basic feasible solution (bfs)* and some additional values called relative costs. The tableau has as many rows as there are constraints in the LP, and as many columns as there are variables in the LP. The columns that correspond to the basic feasible solution are sufficient to compute the objective. As the simplex algorithm solves a linear program, it updates the tableau to transition from one bfs to another. When the number of variables is large however, it is impractical, and moreover unnecessary, to store and update the entire tableau.

As proposed in [26], a practical solution to this problem is a technique called column generation (CG). They motivate the solution by column generation in an ensemble learning framework, resulting in an algorithm called *linear programming boosting* (LPBoost).

CG solves a sequence of smaller problems using only a subset of the columns and thus a smaller tableau. Variables of the excluded columns are fixed at zero. Hence, these are called *restricted master problems*. The dual of the restricted master problem contains only a subset of the original dual constraints. Constraints of the dual may be violated when their corresponding primal variables are constrained to zero, or they may be satisfied automatically. Thus the dual solution of a restricted master problem will either correspond to the solution , or to an infeasible point of the original dual problem.

Once a restricted master problem is solved, its dual solution is analyzed to determine which constraints are still violated. The corresponding primal variables and columns of the restricted master problem are selected to be added to the tableau. The criterion for adding a new column (variable) is the amount by which its corresponding dual constraint is violated. After selection, the updated tableau is efficiently re-optimized using the previous solution.

LPBoost [26] employs CG to select weak hypotheses of the ensemble in rounds. The violation of dual constraints in Equation (6.19) is quantified by the following *score* function

$$S(k) = \sum_{i=1}^{m} u_i y_i h_k(\mathbf{x}_i) - 1 \,.$$

LPBoost selects weak hypotheses indexed by k for which $S(k)$ most positive. The algorithm terminates when there are not more constraint violations.

6.2 Dealing with ambiguity

This section outlines the DPBoost formulation and algorithm as an extension of the 1-norm SVM that is solved by linear programming with column generation.

The input is a set of ambiguous examples $\mathcal{D} = \{(\mathbf{X}_i, Y_i)\,,\ i = 1, \ldots, m\}$, where $\mathbf{X} \in 2^{\mathbb{X}}$ and the output labels are binary $Y \in \mathbb{Y} = \{-1, 1\}$.

As described in Chapter 3, Section 3.2.4, the objective of maximum-margin learning *from ambiguous examples* is a linear discriminant function that robustly separates two categories in feature space $\mathbb{X} \subseteq \mathbb{R}^d$. The theoretical foundation of maximum-margin learning is based on risk minimization. In this context, a bound on the generalization performance of the resulting classifier is derived. The bound depends on a measure of class separation by the linear discriminant function known as the *multi-instance margin.*

The maximum-margin separating hyperplane is characterized as the solution of the optimization problem in Equations (3.23-3.25) and is re-stated here for clarity.

$$\text{MIL Max-Margin} \qquad \max_{\mathbf{w},b} \gamma(\mathcal{D}) \tag{6.23}$$

$$\text{s.t.} \quad Y_i \max_{\mathbf{x}\in\mathbf{X}_i} \left(\langle \mathbf{w}, \mathbf{x}\rangle + b\right) \geq \gamma(\mathcal{D})\,, \forall i, \tag{6.24}$$

$$\|\mathbf{w}\| \leq 1\,. \tag{6.25}$$

This section outlines the formulation of an optimization problem for the maximum-margin separating hyperplane. Unique to this development is the use of the 1-norm bound on the weight vector, which is complemented by a ∞-norm measuring the width of the margin.

6.2.1 1-Norm max margin with ambiguous examples

In this section, the 1-norm $\|\cdot\|_1$ is used to bound the norm of the weight vector. As shown in [46], there is an equivalent formulation in which the norm of the weight vector is minimized, while constraining all *functional margins* to be greater than or equal to 1. Soft-margins are also introduced, allowing the ambiguous example $(\mathbf{X}_i, Y_i)$ to violate the multi-instance margin constraint, at the expense of adding a penalty proportional to ξ_i to the objective. Furthermore, the maximum-margin separating hyperplane can be expressed using the ensemble notations (features $h_k(\mathbf{x})$ and combination weights α_k) from Section 6.1.3.

The revised expression for the maximum-margin separating hyperplane is the following linear programming problem with non-linear constraints, which is called DPBoost

$$\text{DPBoost} \qquad \min_{\alpha,\xi} \sum_{k=1}^{M} \alpha_k + C \sum_{i=1}^{m} \xi_i \tag{6.26}$$

$$\text{s.t.} \quad y_i \max_{\mathbf{x}\in\mathbf{X}_i} \left(\sum_{k=1}^{n} \alpha_k h_k(\mathbf{x}_i) \right) + \xi_i \geq 1,\ \forall i \tag{6.27}$$

$$\alpha_k, \xi_i \geq 0\ \forall k, i\ . \tag{6.28}$$

As discussed in Chapter 3, Section 3.2.3 a self-reinforcing disambiguation is encoded in the multi-instance margin constraint.

As in Chapter 5, Section 5, negatively labeled multi-instances are separated, and treat the member instances as though they came from independent training examples $(\{\mathbf{x}\}, y_i)$ where $\mathbf{x} \in \mathbf{X}_i$ such that $y_i = -1$. So, for example, if an ambiguous training data set consists of labeled multi-instances

$$(\mathbf{X}_1, +1), (\mathbf{X}_2, +1), (\mathbf{X}_3, +1), (\mathbf{X}_4, -1), (\mathbf{X}_5, -1), (\mathbf{X}_6, -1) \tag{6.29}$$

containing $|\mathbf{X}_i| = 3, 6, 2, 2, 7, 4$ members respectively, then the transformation will generate a data set with the same positively labeled multi-instances ($i = 1, 2, 3$) and a new set of negatively labeled multi-instances $(\mathbf{X}_i, -1)$ with $|\mathbf{X}_i| = 1$ for $i = 4, \ldots, 16$. The index set $i = 1, \ldots, m$ of the training set is augmented as necessary depending on the number of member instances in the affected multi-instances.

Once separated, these independent negatively labeled multi-instances behave like standard input-output pairs $(\mathbf{x}, -1)$. Each pair now gives rise to an independent margin constraint as in Equation (6.27), having its own slack variable ξ_i that is penalized in the objective in Equation (6.26). An apparent side effect of this problem transformation is a bias encoded in the new objective function toward solutions that misclassify positive multi-instances. This is because there is only one active slack variable when a positive multi-instance violates its margin constraint. In contrast there could be several active slack variables corresponding to a negatively labeled multi-instance that fails to obtain the desired margin. To compensate for this imbalance, the penalty factor C for the newly created slack variables can be scaled.

In order to deal with the remaining non-linear constraints for positively labeled multi-instances in Equation (6.27), a framework known as *disjunctive programming* [7, 43] is employed. To apply these techniques, it is first necessary to rewrite the optimization problem in a new form.

6.2.2 Geometric interpretation

The following proposition provides a geometric interpretation of the multi-instance margin constraint for each ambiguous example $(\mathbf{X}_i, y_i)$ that appears in Equation (6.27).

Proposition 2. *The following equivalence of sets is a logical consequence of the maximum and union operations contained in their definitions.*

$$\left\{\mathbf{w} \in \mathbb{R}^d \middle| \max_{\mathbf{x}\in\mathbf{X}} \langle \mathbf{w}, \mathbf{x}\rangle \geq 1\right\} = \bigcup_{\mathbf{x}\in\mathbf{X}} \left\{\mathbf{w} \in \mathbb{R}^d \middle| \langle \mathbf{w}, \mathbf{x}\rangle \geq 1\right\} \ . \tag{6.30}$$

Notice that the right hand side describes a union of halfspaces. Proposition 2 implies that the multi-instance margin constraint $y_i \max_{\mathbf{x}\in\mathbf{X}_i} \left(\sum_{k=1}^n \alpha_k h_k(\mathbf{x})\right) + \xi_i \geq 1$ defines a region of feasibility for (α, ξ) that is a union of halfspaces

$$(\alpha, \xi) \in \bigcup_{\mathbf{x}\in\mathbf{X}_i} H_i(\mathbf{x}) \quad , \tag{6.31}$$

where

$$H_i(\mathbf{x}) \equiv \left\{(\alpha, \xi) : \ y_i \sum_k \alpha_k h_k(\mathbf{x}) + \xi_i \geq 1\right\} \tag{6.32}$$

is the region of feasibility corresponding to a single instance $\mathbf{x} \in \mathbf{X}_i$, defined as in Section 6.1.4. Figure 6.2 depicts the disjunctive set described in the Proposition 2.

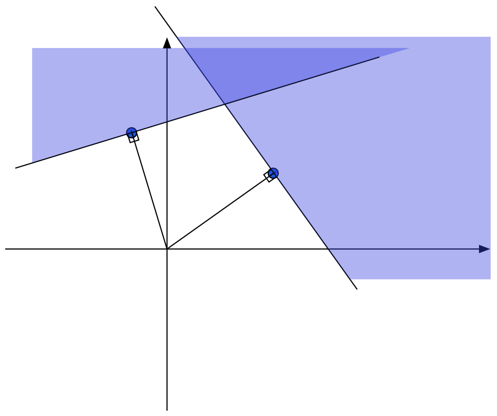


Figure 6.2: Geometry of disjunctive margin constraints. Each ambiguous example defines a disjunctive margin constraint that can be represented as a union of $k = |\mathbf{X}_i|$ halfspaces; in this case $k = 2$.

Under the assumption that negatively labeled multi-instances have only one member, the DPBoost problem is

$$\text{DPBoost} \qquad \min_{\alpha,\xi} \sum_{k=1}^n \alpha_k + C\sum_{i=1}^m \xi_i \tag{6.33}$$

$$\text{(Geometric)} \quad \text{s.t.} \quad (\alpha, \xi) \in \bigcap_{i=1}^m \bigcup_{\mathbf{x}\in\mathbf{X}_i} H_i(\mathbf{x}) \tag{6.34}$$

$$(\alpha, \xi) \in \mathcal{Q} \tag{6.35}$$

One way to visualize the feasible region is to consider the region that is excluded by each multi-instance margin constraint, in other words the complement of this set, written $\left(\bigcup_{\mathbf{x}\in\mathbf{X}_i} H_i\left(\mathbf{x}\right)\right)^c$. By de Morgan's Law, the excluded region is equal to $\bigcap_{\mathbf{x}\in\mathbf{X}_i} H_i\left(\mathbf{x}\right)^c$ which is convex because it is an intersection of halfspaces [61]. The feasible region can thus be constructed starting with the positive quadrant Q and removing a finite number of "convex bites" corresponding to the regions *excluded* by the disjunctive margin constraints of ambiguous examples. For example, the *unshaded* convex region in Figure 6.2 defines one such "convex bite". Removing this region from the positive orthant Q leaves a feasible region as shown in Figure 6.3.

Notice that if $|\mathbf{X}_i| \geq 2$ then the constraint imposed by $\mathbf{X}_i$ is non-convex, since it is defined via a union of halfspaces. However, for multi-instances with $|\mathbf{X}_i| = 1$, the resulting constraints are the same as in Equation (6.17). In other words, the constraint reduces to a single halfspace $\bigcup_{\mathbf{x}\in\mathbf{X}_i} H_j(\mathbf{x}) = H_j$. Since these two cases are treated quite differently in the sequel, the following index sets $A = \{i : |\mathbf{X}_i| \geq 2\}$ and $B = \{i : |\mathbf{X}_i| = 1\}$ are used. Note that the index set B contains the set of negatively labeled inputs $\{i : y_i = -1\}$, since these have been separated.

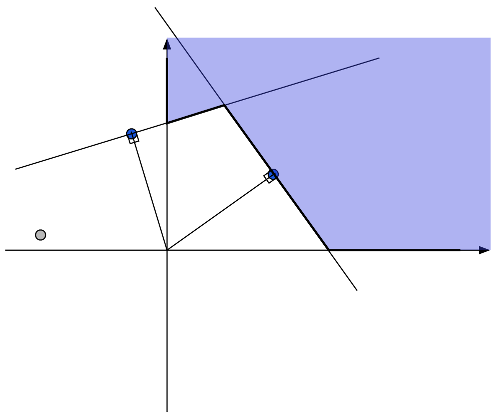


Figure 6.3: "Convex bites". Each ambiguous example excludes a convex region from the positive orthant Q. Here, the corner of Q has been removed by the disjunctive margin constraint from Figure 6.2.

6.2.3 Disjunctive programming

A framework known as *disjunctive programming* has been proposed by [6, 7, 8] for solving optimization problems having non-convex feasible regions known as *disjunctive sets* S that can be expressed as follows

$$S = \bigcap_{i} \bigcup_{j\in I(i)} H_j \ , \tag{6.36}$$

where H_j is a halfspace. Observe that the multi-instance margin constraints in Equation (6.34) form a disjunctive set, as derived in Section 6.2.2. Likewise, the positive quadrant $\mathcal{Q} = \{(\alpha, \xi) : \ \alpha, \xi \geq 0\}$ can be expressed as an intersection of axis-aligned halfspaces, one for each coordinate of α and ξ.

The basic idea of the disjunctive programming is to generate an approximation to the feasible region that is convex, easy to compute, and closely approximates the original disjunctive set. At this point, it suffices to note that one can define a hierarchy of enclosing convex sets S_t, so that $S \subseteq S_t$ and $S_{t+1} \subseteq S_t$. Each set in this hierarchy is convex and has a representation as a set of linear inequalities. Traversing this hierarchy allows one to trade-off the quality of the approximation to S with the complexity of the representation for the current approximation. A complete introduction to this topic is provided in [6, 7, 8].

6.2.4 Convexification

Disjunctive programming replaces the non-convex feasible region in Equation (6.34) with an enclosing convex surrogate that is easily computed. Since the convex surrogate is larger than the original feasible region, a *relaxation* of the disjunctive program is obtained. The relaxed problem can be expressed as a linear program for which the global minimum can be computed efficiently. However, the solution may not satisfy the multi-instance margin constraints. If the convex surrogate is close to the original feasible region, then there reason to believe that resulting discriminant function obtains a large overall margin.

Let $\mathrm{conv}(S)$ denotes the *convex hull* of S. It is a convex set consisting of all convex combinations of elements from S [61] and can be expressed as

$$\mathrm{conv}\,(S) = \left\{ \mathbf{x} | \mathbf{x} = u_1\mathbf{x}_1 + \ldots + u_m\mathbf{x}_m,\ \mathbf{x}_k \in S,\ 0 \le u_k \le 1\ \forall k, \sum_{k=1}^{m} u_k = 1 \right\} .$$

This definition is included as an aid to the reader, and is not used to represent the convex hull in practice.

A convex surrogate known as the *hull-relaxation* is obtained from the region in Equations (6.34-6.35) by the following replacement

$$(\alpha, \xi) \in \bigcap_{i \in A} \mathrm{conv} \left[\bigcup_{\mathbf{x} \in \mathbf{X}_i} H_i(\mathbf{x}) \cap \mathcal{Q} \right] \cap \bigcap_{j \in B} H_j\ . \tag{6.37}$$

Notice that for the unambiguous examples, indexed by $j \in B$, there is no need to employ convexity; the convex hull of a halfspace is the halfspace itself. Figure 6.4 shows how convexity is applied individually to a disjunctive margin constraint in the hull-relaxation. Compare the solution in this figure to those depicted in Figure 6.1. The only difference between these toy problems is the fact that one of the examples was augmented so that it became ambiguous. Despite the ambiguity, the solution vector in the hull-relaxation is close to the 1-norm solution of the unambiguous problem.

Figure 6.5 depicts the overall hull-relaxation of Equation (6.37) and compares it to the region defined by Equation (6.34).

6.2.5 Parallel reductions

A hierarchy of such relaxations can be built, using the fundamental fact that

$$\mathrm{conv}(S) \cap T \supseteq \mathrm{conv}(S \cap T)\,, \tag{6.38}$$

when T is a convex set [7]. A tighter convex relaxation is obtained, if the convexification of S occurs after the intersection with T. Thus, the hierarchy of relaxations is formed by recursively “folding” the convex sets

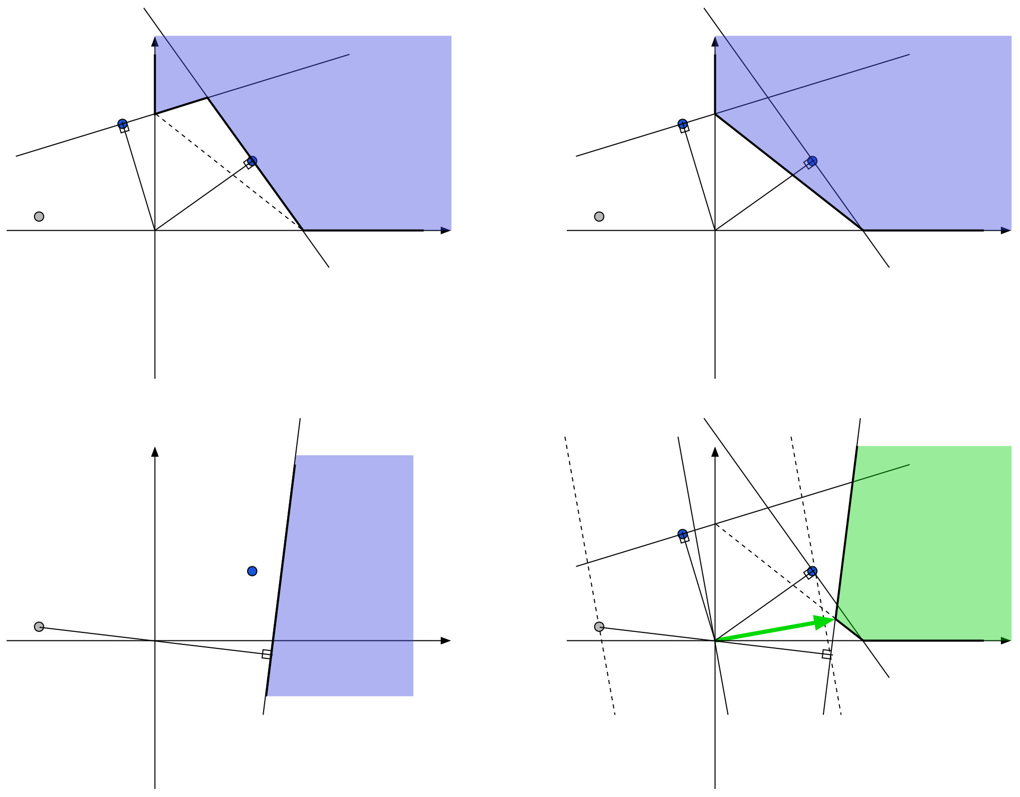


Figure 6.4: The effect of hull-relaxation for the ongoing example. The top row shows the single disjunctive margin constraint before and after convexification. The final region of feasibility is obtained by intersecting the region in the top-right with the set $\bigcap_{j \in B} H_j$. However, since there is only one unambiguous example with $Y_j = -1$ (the grey point), the set to be intersected is the single halfspace depicted in the bottom-left. The bottom-right image depicts the result of this intersection, in addition to: the shortest weight vector in this region measured with respect to the 1-norm; the corresponding hyperplane; and two parallel hyperplanes depicting the margin. Compare the solution in the bottom-right with those in Figure 6.1.

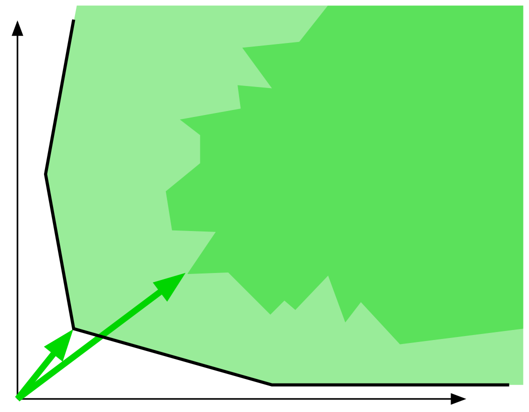


Figure 6.5: The overall effect of the hull-relaxation in Equation (6.37). The shorter arrow is the approximate solution, while the longer arrow is the exact solution to the original disjunctive program DP.

into one another. Each folding operation is a *reduction step*. See Figure 6.6 for a geometric perspective of parallel reductions.

In the context of Equation (6.37), reductions can be made with the unambiguous constraints $\bigcap_{j \in B} H_j$. These convex sets can be folded, in parallel, into each convexified region $\mathrm{conv}\left[\bigcup_{\mathbf{x} \in \mathbf{X}_i} H_i(\mathbf{x})\right], \forall i$. This is called a *parallel reduction* step [7]. The convex surrogate of the constraints in Equation (6.31) becomes

$$(\alpha, \xi) \in \bigcap_{i \in A} \mathrm{conv}\left[\bigcup_{\mathbf{x} \in \mathbf{X}_i} \left(H_i(\mathbf{x}) \cap \mathcal{Q} \cap \bigcap_{j \in B} H_j\right)\right]. \tag{6.39}$$

Results by Balas [6, 7] demonstrate that the maximum number of facets of the convex hull grows exponentially with the number of reductions. For this reason, it is important to perform parallel reductions only with halfspaces H_j that will affect the hull-relaxation. Figure 6.7 depicts the gradual strengthening of the hull-relaxation using parallel reductions. The selection of parallel reductions is described in Section 6.2.8.

6.2.6 Linearization

There is a lift-and-project representation for the convex surrogates in Equation (6.39), i.e. one can characterize the feasible set as a projection of a higher dimensional polyhedron which can be explicitly characterized, as shown in [7].

Proposition 3. *Assume a set of non-empty linear constraints $H_i \equiv \left\{z : A^i z \geq b^i\right\} \neq \emptyset$ is given. Then $z \in \mathrm{conv} \bigcup_i H_i$ if and only if there exist z^j and $\eta^j \geq 0$ such that*

$$\begin{aligned}\sum_j z^j &= z \\ \sum_j \eta^j &= 1 \\ A^j z^j &\geq \eta^j b^j .\end{aligned}$$

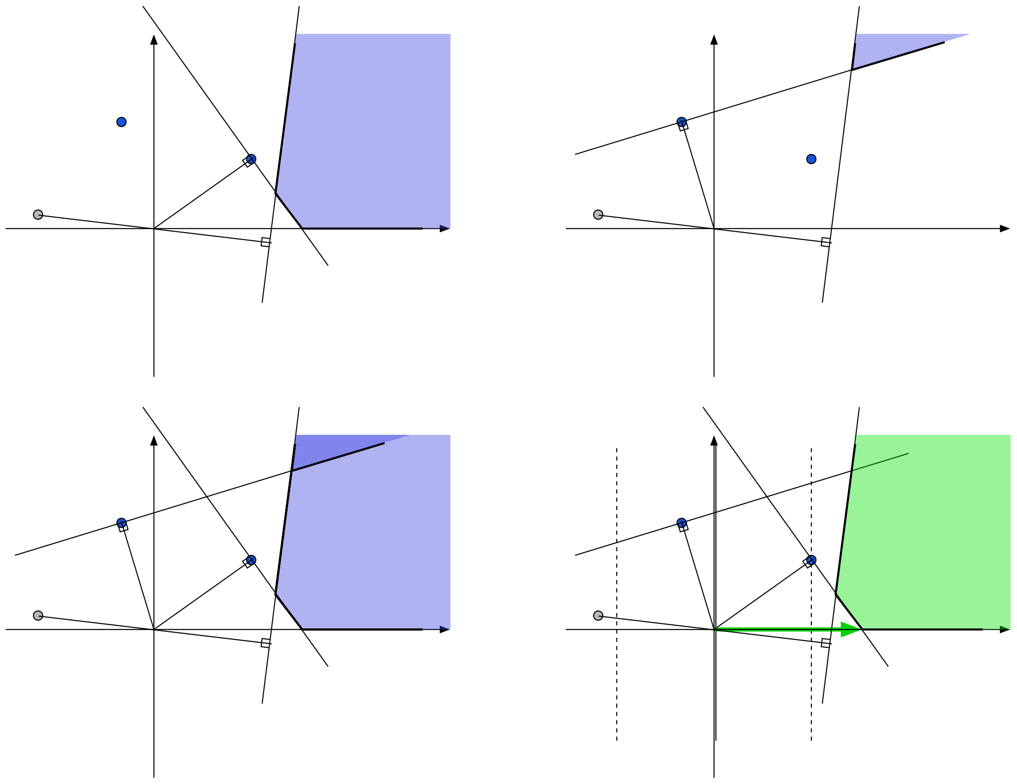


Figure 6.6: The effect of parallel reductions on a small example. Denote by $\mathbf{X}_i = \{\mathbf{x}_1, \mathbf{x}_2\}$ the single ambiguous example having two instances (blue circles), and by $\mathbf{X}_j = \{\mathbf{x}_j\}$ the single unambiguous example (grey circle). Parallel reductions are performed by intersecting $H_j(\mathbf{x}_j)$ with each disjunct before convexification. The top row depicts the two relevant intersections $H_i(\mathbf{x}_1) \cap \mathcal{Q} \cap H_j$ and $H_i(\mathbf{x}_2) \cap \mathcal{Q} \cap H_j$. The bottom-left shows the union of these regions before the convexification step in Equation (6.39); both shaded regions are included in the union. Since the union is convex, the region does not change upon convexification. Therefore, this is the final hull-relaxation after performing the parallel reduction step. Also depicted are: the resulting shortest weight vector measured with respect to the 1-norm; the corresponding hyperplane; and two parallel hyperplanes depicting the margin. Compare the solution in the bottom-right with those in Figure 6.1 and 6.4.

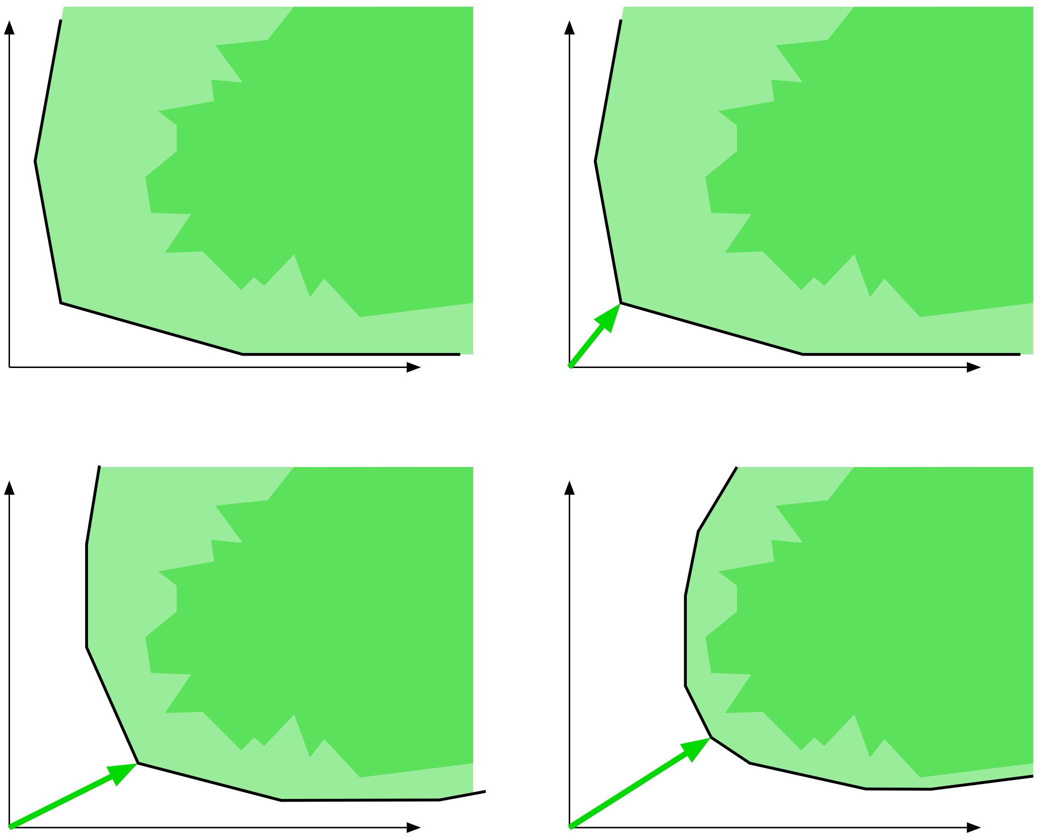


Figure 6.7: The overall effect of applying parallel reductions incrementally to the hull-relaxation.

Proof. [7] $\square$

Applying this linearization to every convex hull in Equation (6.39) individually, notice that one needs to introduce duplicates $\alpha^{\mathbf{x}}, \xi^{\mathbf{x}}$ of the parameters α and slack variables ξ, for every $\mathbf{x} \in \mathbf{X}_i$. In addition to the constraints $\alpha_k^{\mathbf{x}}, \xi_i^{\mathbf{x}}, \xi_j^{\mathbf{x}}, \eta_i^{\mathbf{x}} \geq 0$, the relevant constraint set for *each individual* multi-instance $\mathbf{X}_i,\ i \in A$ of the resulting LP can be written as

$$\forall \mathbf{x} \in \mathbf{X}_i: \quad y_i \sum_k \alpha_k^{\mathbf{x}} h_k(\mathbf{x}) + \xi_i^{\mathbf{x}} \geq \eta_i^{\mathbf{x}}, \tag{6.40}$$

$$\forall \mathbf{x} \in \mathbf{X}_i, \forall j \in B: \quad y_j \sum_k \alpha_k^{\mathbf{x}} h_k(\mathbf{x}_j) + \xi_j^{\mathbf{x}} \geq \eta_i^{\mathbf{x}}, \tag{6.41}$$

$$\forall j \in B: \quad \xi_j = \sum_{\mathbf{x} \in \mathbf{X}_i} \xi_j^{\mathbf{x}}, \tag{6.42}$$

$$\forall k: \quad \alpha_k = \sum_{\mathbf{x} \in \mathbf{X}_i} \alpha_k^{\mathbf{x}}, \tag{6.43}$$

$$\xi_i = \sum_{\mathbf{x} \in \mathbf{X}_i} \xi_i^{\mathbf{x}}, \tag{6.44}$$

$$1 = \sum_{\mathbf{x} \in \mathbf{X}_i} \eta_i^{\mathbf{x}} \tag{6.45}$$

The first margin constraint in Equation (6.40) is the one associated with the specific instance $\mathbf{x}$, while the second set of margin constraints in Equation (6.41) stems from the parallel reduction performed with unambiguous member examples.

6.2.7 Duality

Following the approach taken in LPBoost, the dual of the above problem is computed. The key result from this derivation is a complicated constraint in terms of the dual u-variables that will be used for column generation

$$\forall i,\, \forall \mathbf{x} \in \mathbf{X}_i : y_i u_i^{\mathbf{x}} h_k(\mathbf{x}) + \sum_{j \in B} y_j u_j^{\mathbf{x}} h_k(\mathbf{x}_j) \leq \rho_{ik}, \quad \sum_{i \in A} \rho_{ik} = 1\,. \tag{6.46}$$

The primal variables are α_k, $\alpha_k^{\mathbf{x}}$, ξ_i, $\xi_i^{\mathbf{x}}$, $\xi_j^{\mathbf{x}}$, and $\eta_i^{\mathbf{x}}$. Dual variables $u^{\mathbf{x}}$ and $u_j^{\mathbf{x}}$ are introduced for the margin constraints, and dual variables ρ_{ik}, σ_i, and θ_i are introduced for the equality constraints on α_k, ξ and η, respectively. The Lagrangian is formed by adding each constraint from the primal problem to the objective, multiplied by the corresponding dual variable

$$\begin{aligned} L = &\sum_k \alpha_k + C\left(\sum_i \xi_i + \sum_j \xi_j\right) - \sum_i \sum_{\mathbf{x} \in \mathbf{X}_i} u_i^{\mathbf{x}} \left(y_i \sum_k \alpha_k^{\mathbf{x}} h_k(\mathbf{x}) + \xi_i^{\mathbf{x}} - \eta_i^{\mathbf{x}}\right) \\ &- \sum_i \sum_{\mathbf{x} \in \mathbf{X}_i} \sum_j u_j^{\mathbf{x}} \left(y_j \sum_k \alpha_k^{\mathbf{x}} h_k(\mathbf{x}_j) + \xi_j^{\mathbf{x}} - \eta_i^{\mathbf{x}}\right) + \sum_i \theta_i \left(1 - \sum_{\mathbf{x} \in \mathbf{X}_i} \eta_i^{\mathbf{x}}\right) \\ &- \sum_{i,k} \rho_{ik} \left(\alpha_k - \sum_{\mathbf{x} \in \mathbf{X}_i} \alpha_k^{\mathbf{x}}\right) - \sum_i \sigma_i \left(\xi_i - \sum_{\mathbf{x} \in \mathbf{X}_i} \xi_i^{\mathbf{x}}\right) - \sum_{i,j} \sigma_{ij} \left(\xi_j - \sum_{\mathbf{x} \in \mathbf{X}_i} \xi_j^{\mathbf{x}}\right) \\ &- \sum_i \sum_{\mathbf{x} \in \mathbf{X}_i} \sum_k \tilde{\alpha}_k^{\mathbf{x}} \alpha_k^{\mathbf{x}} - \sum_i \sum_{\mathbf{x} \in \mathbf{X}_i} \tilde{\xi}_i^{\mathbf{x}} \xi_i^{\mathbf{x}} - \sum_i \sum_{\mathbf{x} \in \mathbf{X}_i} \sum_j \tilde{\xi}_j^{\mathbf{x}} \xi_j^{\mathbf{x}} - \sum_i \sum_{\mathbf{x} \in \mathbf{X}_i} \tilde{\eta}_i^{\mathbf{x}} \eta_i^{\mathbf{x}}\,. \end{aligned}$$

Taking derivatives w.r.t. primal variables, and simplifying, leads to the following dual problem

$$
\begin{aligned}
\text{DPboost-dual} \qquad & \max \sum_i \theta_i \\
\text{s.t.} \quad & \theta_i \le u_i^{\mathbf{x}} + \sum_j u_j^{\mathbf{x}} \;, \\
& u_i^{\mathbf{x}} \le C \;, \\
& u_j^{\mathbf{x}} \le \sigma_{ij}, \\
& \sum_i \sigma_{ij} \le C \;, \\
& y_i u_i^{\mathbf{x}} h_k(\mathbf{x}) + \sum_j y_j u_j^{\mathbf{x}} h_k(\mathbf{x}_j) \le \rho_{ik}, \\
& \sum_i \rho_{ik} \le 1 \;.
\end{aligned}
$$

6.2.8 Row and column selection

The size of the resulting problem is significant. To make this explicit, first define quantities $q = \sum_{i \in A} K_i$ and $r = \sum_{i \in B} K_i$ to denote the number of instances in ambiguous and unambiguous bags, plus the quantity $p = |A|$ denoting the number of ambiguous bags. Then, counting the variables in the primal (columns), note that there are $O(q \cdot n)$ replicated $\alpha_k^{\mathbf{x}}$ variables, and $O(q \cdot r)$ replicated $\xi_j^{\mathbf{x}}$ variables; and that both q and r are $O(m)$. Thus, as a result of linearization and parallel reductions, the number of variables in the primal LP is now $O\left(m \cdot n + m^2\right)$, compared to $O(n+m)$ of the standard LPBoost. Similarly, counting dual variables (rows), note that there are $O(q \cdot r)$ margin constraints due to the parallel reductions, and $O(p \cdot n)$ constraints due to the summation constraints on $\alpha_k^{\mathbf{x}}$. Thus, the number of constraints has also been inflated significantly from $O(m)$ in LPBoost, to $O\left(m^2 + m \cdot n\right)$.

In order to maintain the spirit of LPBoost in dealing efficiently with a large-scale linear program, a column generation scheme is used to select one or more $\alpha_k^{\mathbf{x}}$ in every round. Notice that all $\alpha_k^{\mathbf{x}}$ variables are linked through the equality constraints $\sum_{\mathbf{x} \in \mathbf{X}_i} \alpha_k^{\mathbf{x}} = \alpha_k$ for all $\mathbf{X}_i$. Thus, the column selection can not proceed for every multi-instance $\mathbf{X}_i$ independently. In particular, $\alpha_k^{\mathbf{x}} > 0$ implies $\alpha_k > 0$, so that $\alpha_k^{\mathbf{w}} > 0$ for at least some $\mathbf{w} \in \mathbf{X}_i$ for each $i \in A$. All columns $\{\alpha_k^{\mathbf{x}} : \mathbf{x} \in \mathbf{X}_i,\ i \in A\}$ involving the same weak hypothesis are added simultaneously. In order to select a feature h_k, the following score is computed

$$
S(k) = \sum_i \bar{\rho}_{ik} - 1, \quad \bar{\rho}_{ik} \equiv \max_{\mathbf{x} \in \mathbf{X}_i} \left[y_i u_i^{\mathbf{x}} h_k(\mathbf{x}) + \sum_{j \in B} y_j u_j^{\mathbf{x}} h_k(\mathbf{x}_j) \right] \;. \tag{6.47}
$$

Notice that due to the block structure of the tableau, working with a reduced set of columns also eliminates a large number of inequalities (rows). However, the large set of $q \cdot r$ inequalities for the parallel reductions is still prohibitive.

In order to address this problem, incremental row selection is performed in an outer loop. A subset of rows corresponding to the most useful parallel reductions is computed using the column basis for the current relaxed LP. One can use the magnitude of the margin violation as a heuristic to perform this row selection.

Hence the following score is used

$$T(\mathbf{x}, j) = \eta_i^{\mathbf{x}} - y_j \sum_k \alpha_k^{\mathbf{x}} h_k(\mathbf{x}_j), \quad \text{where } \mathbf{x} \in \mathbf{X}_i,\ i \in A,\ j \in B \tag{6.48}$$

This means that for current values of the duplicated ensemble weights $\alpha_k^{\mathbf{x}}$, one selects the parallel reduction margin constraint associated with ambiguous member instance $\mathbf{x}$ and unambiguous instance j that is violated most strongly.

The margin constraints imposed by unambiguous training instances $(\mathbf{x}_j, y_j)$ are redundant after the parallel reduction steps in Equation (6.39), however, it may not be possible to perform all reductions. Therefore, they are also added to the problem in unreduced form. This helps feature selection during the initial rounds. The following constraints are thus added to the primal

$$y_j \sum_k \alpha_k h_k(\mathbf{x}_j) + \xi_j \geq 1, \quad \forall j \in J\,, \tag{6.49}$$

which will introduce additional dual variables u_j, $j \in J$. Notice that in the worst case where all inequalities imposed by training multi-instances $\mathbf{X}_i$ are vacuous, this will make sure that one recovers the standard LPBoost formulation on the individual member instances. One can then think of the row generation process as a way of deriving useful information from ambiguous examples. As rows are added to the approximation, the feasible regions corresponding to ambiguous candidate member instances are eliminated from consideration if they are not consistent with the unambiguous constraints. This information takes the form of linear inequalities in the high dimensional representation of the convex hull and will sequentially reduce the version space, i.e. the set of feasible (α, ξ) pairs.

Given a solution to the relaxed problem, one can easily obtain lower and upper bounds on the solution value of the original problem. First, since the convex surrogate is a superset of the feasible region of the original problem, its minimal value is a lower bound on the original problem $H^*_{relaxed} \leq H^*_{dpboost}$. An upper bound $H^*_{dpboost} \leq H^*_{projected}$ is obtained by projecting the solution of the relaxed problem back onto the original feasible space.

A heuristic projection is performed iteratively as follows. An ambiguous example is chosen, and a "most-positive" instance from the example is selected. The most-positive instance and the multi-instance label are added to the problem as an unambiguous example. The remaining instances from the multi-instance are discarded. The feasible region is updated and the objective is re-optimized. This process converges when there are no more ambiguous examples. The final solution must satisfy at least one halfspace constraint, corresponding to the most-positive instance from each ambiguous example. Therefore the final solution lies within the original disjunctive feasible region, providing the upper bound.

The difference $H^*_{projected} - H^*_{relaxed}$ can be used as a termination condition during learning. It signals when the algorithms has already found an optimal solution to the original disjunctive programming problem. If needed, the upper and lower bounds can be further strengthened by using them in combination with branch-and-bound search.

The resulting method Disjunctive Programming Boosting (DPBoost). Pseudo-code for this algorithm is presented in Box 6.1

Algorithm 6.1 DPBoost Algorithm

```
1:  input: training sample $\mathcal{D}$
2:  initialize $H = \emptyset$, $C = \{\xi_i : i \in A \cup B\}$, $R = \{u_i^{\mathbf{x}} : \mathbf{x} \in \mathbf{X}_i, i \in A\} \cup \{u_j : j \in B\}$
    $u_j = \frac{1}{|J|}$, $u_i^{\mathbf{x}} = 0$, $\xi_i = 0$
3:  repeat
4:      repeat
5:          column selection: select $h_k \notin H$ with maximal $S(k)$
6:          $H = H \cup \{h_k\}$
7:          $C = C \cup \{\alpha_k\} \cup \{\alpha_k^{\mathbf{x}} : \forall \mathbf{x} \in \mathbf{X}_i, \forall i \in A\}$
8:          solve $LP(C, R)$
9:      until $\max_k S(k) < \epsilon$
10:     row selection: select a set $S$ of pairs $(\mathbf{x}, j) \notin R$ with maximal $T(\mathbf{x}, j) > 0$
11:     $R = R \cup \{u_j^{\mathbf{x}} : (\mathbf{x}, j) \in S\}$
12:     $C = C \cup \{\xi_j^{\mathbf{x}} : (\mathbf{x}, j) \in S\}$
13:     solve $LP(C, R)$
14: until $\max_{\mathbf{x}, j} T(\mathbf{x}, j) < \epsilon$
```

6.3 Experiments

The algorithm was implemented in MATLAB[1] using a MEX-based interface to the linear programming solver by ILOG CPLEX [2]. Due to memory handling inefficiencies that were unavoidable in the MEX-based interface, experiments had to be restricted to a small scale.

6.3.1 Proof of concept

A set of synthetic weakly labeled data sets was generated to evaluate DPBoost on a small scale. These were multiple-instance data sets, where the label uncertainty was asymmetric; the only ambiguous bags ($|X_i| > 1$) were positive. More specifically, the generated instances $x \in [0, 1] \times [0, 1]$ were sampled uniformly at random from the white ($y_i = 1$) and black ($y_i = -1$) regions of Figure 6.8, leaving the intermediate gray area as a separating margin. The degree of ambiguity was controlled by generating ambiguous bags of size $k \sim$ Poisson(λ) having only one positive and $k-1$ negative instances. To control data set size, a pre-specified number of ambiguous bags were generated, and the same number of singleton unambiguous bags.

As a proof of concept benchmark, the classification performance of DPBoost was compared with three other LPBoost variants: perfect-knowledge, perfect-selector, and naive algorithms. All variants use LPBoost as their base algorithm and have slightly different preprocessing steps to accommodate the MIL data sets. The first corresponds to the supervised LPBoost algorithm; i.e. the true member instance labels are used. Since this algorithm does not have to deal with ambiguity, it will perform better than DPBoost. The second algorithm uses the true member instance labels to prune negative instances from ambiguous bags and thus solves a smaller and fully supervised problem using LPBoost. This algorithm provides an interesting benchmark, since its performance is the best one can hope for from DPBoost. At the other extreme, the third variant assumes the ambiguous member instance labels are equal to their respective bag labels.

[1]http://www.mathworks.com

[2]http://www.ilog.com

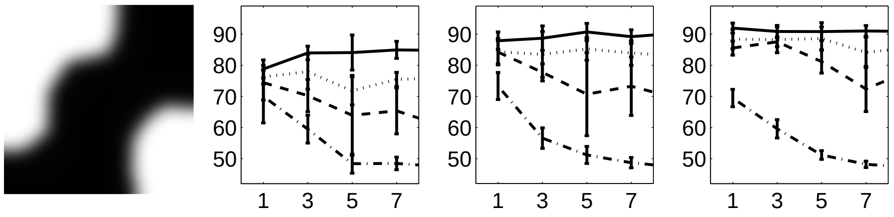


Figure 6.8: (Left) Normalized intensity plot used to generate synthetic data sets. (Right) Performance relative to the degree of label ambiguity. Mean and standard deviation of the classification accuracy for member instances plotted versus λ, for perfect-knowledge (solid), perfect-selector (dotted), DPBoost (dashed), and naive (dash-dot) algorithms. The three plots correspond to data sets of size $|I| = 10,\ 20,\ 30$.

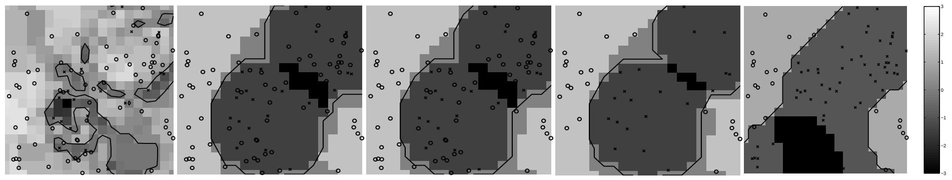


Figure 6.9: Discriminant boundaries learned by naive (accuracy = 53.3 %), DPBoost (85.3 %), DPBoost with projection (85.3%), perfect-selector (86.6 %) and perfect-knowledge (92.7 %) algorithms.

For all algorithms, thresholded RBF features were used. These were defined by thresholding radial basis functions centered on the training instances $h_{kj}(\mathbf{x}) = \operatorname{sgn}\left(\exp\left(-\|\mathbf{x} - \mathbf{x}_k\|^2\right) - t_j\right)$, $\forall kj$, where $\mathbf{x}_k$ is the k-th instance in the training data set, and t_j is the j-th chosen threshold value. An efficient implementation uses inner products to compute the square of the Euclidean norm, and avoids the need to exponentiate by taking the logarithm of the thresholds $t_j, \forall j$.

Figure 6.9 shows the discriminant boundary (black line), learned by each of the four algorithms for a data set generated with $\lambda = 3$ and having 20 ambiguous bags (i.e. $|I| = 20$, no. ambig. $=$ 71, no. total $=$ 91). The ambiguous instances are marked by "o", unambiguous ones "x", and the background is shaded to indicate the value of the ensemble $F(x)$ (clamped to $[-3,\ 3]$). It is clear from the shading that the ensemble has a small number of active features for DPBoost, perfect-selector and perfect-knowledge algorithms. For each classifier, the classification accuracy on individual instances sampled uniformly on a (21 x 21) grid is reported. The sparsity of the dual variables was also verified; less than 20 percent of the dual variables and reductions were active.

Experiments were run using 5-fold cross-validation on the synthetic data sets for $\lambda = 1, 3, 5, 7$ and for data sets having $|I| = 10,\ 20,\ 30$. Figure 6.8 (right side) shows the mean classification accuracy for individual instances with error bars showing one standard deviation, as a function of the parameter λ.

6.4 Discussion

A disjunctive programming framework has been proposed to solve the learning from ambiguous examples problem. The key contribution of this method is a technique for avoiding the local minima that have hampered the search for a globally optimal solution to the maximum-margin separating hyperplane problem in Equations (3.23-3.25).

Using synthetic data, the disambiguation and learning ability of the algorithm has been demonstrated. The current implementation could not handle large data sets due to the MEX-based interface that was used.

Chapter 7

LNPBoost

A third algorithm, lift-and-project boosting (LNPBoost), proposed in [3], also uses the 1-norm bound the weight vector. This approach was motivated by the need to limit the size and management difficulties of the linear program in DPBoost. The lifted formulation used in DPBoost has a worst case space complexity that is quartic in the size of the data set. In the third presented approach, the main difference is the technique used to approximate the feasible region by a convex set. Instead of approximating the convex set using Balas' hull-relaxation and parallel reductions, a sequential compact representation of the convex hull is computed. This approximation technique is much more efficient, converges in the limit to the globally optimal maximum-margin separating hyperplane, and outperforms DPBoost and MI-SVM empirically.

7.1 Introduction

Following Chapter 6, the structural risk minimization principle is used to derive a max-margin formulation of the multiple-instance learning problem. This results in a non-convex formulation of multiple-instance learning that is a natural extension of the 1-norm SVM [78]. In this approach, integer variables are not used in our representation, and instead a linear programming relaxation to this problem is constructed. In theory, this step simply replaces the non-convex feasible region with its convex hull.

In practice, a cutting-plane algorithm is used to construct a compact sequential approximation to the convex hull using linear inequalities in the original feature space. These inequalities are generated using a technique based on *lift-and-project* cuts due to [9]. The generation of lift-and-project cuts is performed in an online fashion whereby only a single ambiguous example is considered in each step. Cut set regularization ensures that the approximation to the convex hull remains compact. This differs significantly from DPBoost which uses the technique of lifting on the entire data set (i.e. in a batch manner) to represent an enclosing set called the *hull-relaxation*. DPBoost was difficult to scale to large data sets due to the memory requirements of lifting. The proposed technique solves this problem. For the sake of comparison, ROMMA [44] uses a 2-norm SVM formulation and approximates the convex hull using a single cutting-plane.

The cutting-plane algorithm is also embedded within a boosting framework for feature selection. In this way, the proposed learning algorithm is able to consider high-dimensional feature mappings required to generate expressive linear classifiers. Since the lift-and-project method is the central component of the cutting-plane algorithm, the entire algorithm is called LNPBoost for *lift-and-project boosting*.

7.2 Dealing with ambiguity

As outlined in Chapter 6, Section 6.2, the input is a set of ambiguous examples $\mathcal{D} = \{(\mathbf{X}_i, Y_i)\,,\ i = 1, \ldots, m\}$, where $\mathbf{X} \in 2^{\mathbb{X}}$ and the output labels are binary $Y \in \mathbb{Y} = \{-1, 1\}$. The objective of maximum-margin learning *from ambiguous examples* is a linear discriminant function that robustly separates two categories in feature space $\mathbb{X} \subseteq \mathbb{R}^d$. The maximum-margin separating hyperplane is characterized as the solution of the optimization problem in Equations (3.23-3.25) and is re-stated here for clarity.

$$\text{MIL Max-Margin} \qquad \max_{\mathbf{w},b} \gamma(\mathcal{D}) \tag{7.1}$$

$$\text{s.t.} \quad Y_i \max_{\mathbf{x}\in\mathbf{X}_i} \left(\langle \mathbf{w}, \mathbf{x}\rangle + b\right) \geq \gamma(\mathcal{D})\,, \forall i, \tag{7.2}$$

$$\|\mathbf{w}\| \leq 1\,. \tag{7.3}$$

The *geometric margin* is defined as the distance between the point $\mathbf{x}$ and the closest point $\mathbf{p}$ on the hyperplane $\langle \mathbf{w}, \mathbf{p}\rangle = 0$, measured with respect to the norm $\|\cdot\|'$ that is dual to $\|\cdot\|$ [46]. Because $\|\mathbf{w}\| = 1$ for $f \in \mathcal{F}$, the principle of structural risk minimization for ambiguous examples MI-SRM from Definition *17 is maximizing the minimal geometric margin*. As shown in [46], there is an equivalent formulation in which the norm of the weight vector is minimized, while constraining all *functional margins* to be greater than or equal to 1. In this case, the constraints are called *margin constraints*.

As in Equations (6.33-6.35), the reformulation using the ∞-norm margin and its dual 1-norm on the weight vector can be written using geometric set notations. Denoting a halfspace parameterized by a vector $\mathbf{x}$ and an offset γ by $H(\mathbf{x}, \gamma) = \left\{\mathbf{w} \in \mathbb{R}^d |\ \langle \mathbf{w}, \mathbf{x}\rangle \geq \gamma\right\}$, the margin constraint for an **unambiguous** example $(\mathbf{x}, y)$ can be expressed using set notation as follows

$$\mathbf{w} \in H(y\mathbf{x}, 1)\,. \tag{7.4}$$

Similarly, the following identity allows one to write the margin constraint for an **ambiguous** example $(\mathbf{X}, Y)$ in set notation

$$\mathbf{w} \in \left\{\mathbf{w} \in \mathbb{R}^d | Y \max_{\mathbf{x}\in\mathbf{X}} \langle \mathbf{w}, \mathbf{x}\rangle \geq 1\right\} = \bigcup_{\mathbf{x}\in\mathbf{X}} H(Y\mathbf{x}, 1)\ . \tag{7.5}$$

Without loss of generality, the optimal solution lies in the positive orthant (see [26]), and furthermore, has 1-norm bounded by some a priori known constant M , i.e.

$$\mathbf{w} \in Q = \left\{\mathbf{w} \in \mathbb{R}^d |\ \mathbf{w} \geq 0\,,\ \|\mathbf{w}\|_1 \leq M\right\}\ . \tag{7.6}$$

Notice that the notation for halfspaces $H(\mathbf{x}, \gamma)$ defined here differs from $H_i(\mathbf{x})$, as defined in Equation (6.32) and used throughout Chapter 6. For ease of exposition the variables α_k, and subsequently ξ_i, are consolidated within $\mathbf{w}$. The disjunctive program (DP) is

$$\text{DP} \qquad \min_{\mathbf{w}} \|\mathbf{w}\|_1 \tag{7.7}$$

$$\text{s.t.} \quad \mathbf{w} \in \bigcap_{i=1}^{m} \bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x}, 1\right) \cap Q \; . \tag{7.8}$$

The non-convexity of the feasible region is the primary challenge of solving this optimization problem. To alleviate problems local-minima, a linear programming relaxation to the problem is formulated by replacing the feasible region with its convex hull

$$\mathbf{F} = \text{conv}\left[\bigcap_{i=1}^{m} \bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x}, 1\right) \cap Q\right] \; . \tag{7.9}$$

Figure 7.1 shows the effect of convexification on the feasible region of DP.

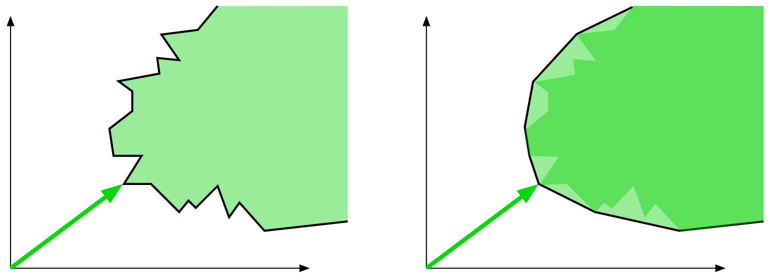


Figure 7.1: The effect of convexification. (Left) Non-convex region. (Right) Convex region.

This results in the following *relaxed* DP

$$\text{RDP} \qquad \min_{\mathbf{w}} \|\mathbf{w}\|_1 \tag{7.10}$$

$$\text{s.t.} \quad \mathbf{w} \in \mathbf{F} \; . \tag{7.11}$$

It can be shown that the solution of RDP corresponds to the global minimum of DP. First, observe that the solution of DP corresponds to the shortest vector measured with respect to the 1-norm in the non-convex DP feasible region. This must also be one of the extreme points of its convex-hull $\mathbf{F}$. Since RDP is optimizing over $\mathbf{F}$, and since the solution of RDP occurs at an extreme point of $\mathbf{F}$, it must also be the solution of DP.

For linear discriminant functions, it is also possible to extend these definitions to allow for a soft-margin using a simple mapping: $\mathbf{x} \leftarrow (\mathbf{x}, Y_i\mathbf{e}_i/C)$, $\forall\mathbf{x} \in \mathbf{X}_i$, $\Sigma_i \leftarrow C\xi_i$, $\mathbf{w} \leftarrow (\mathbf{w}, \Sigma)$, where $\mathbf{e}_i \in \mathbb{R}^m$ is zero apart from a 1 in the i-th dimension, and $\xi_i \geq 0$ is a slack variable. Each margin constraint (ambiguous or unambiguous) can be weakened by increasing the corresponding slack variable. The positive slack variables, scaled by a constant C, are penalized linearly in the objective. If desired, two separate slack penalties C_{ambig} and C_{unambig} can be used to regulate the relative influence of constraint violations made by ambiguous examples over those made by unambiguous examples. The penalties are set using the parameter $C_{\text{balance}} \in [0, 1]$ as follows: $C_{\text{ambig}} = C * C_{\text{balance}}$ and $C_{\text{unambig}} = C * (1 - C_{\text{balance}})$.

7.3 The cutting-plane algorithm

The key idea employed by cutting-plane algorithms is that one is not so much interested in representing the complete feasible region $\mathbf{F}$, but instead in finding a good approximation to it near the optimal solution [41, 52]. Each iteration generates a hyperplane that "cuts-away" a slice of the approximation. It is possible to construct an approximation to the convex $\mathbf{F}$ efficiently in this manner. It is shown that, in the limit, the solution will be inside $\mathbf{F}$. The proof is based on the work of [53] on a sequential convexification algorithm for mixed-integer programming.

7.3.1 Sequential convexification

The cutting-plane algorithm constructs a nested sequence of convex sets denoted $\mathbf{F}_t$ such that $\mathbf{F} \subseteq \mathbf{F}_t$. Each approximation $\mathbf{F}_t$ is defined by a finite intersection of halfspaces

$$\mathbf{F}_t = \bigcap_{i=1}^{N_t} H\left(\alpha_i, \beta_i\right) \ . \tag{7.12}$$

Each defining hyperplane (α, β) is called a *cut*, and the set $\mathcal{C}_t = \{(\alpha_i, \beta_i) \,| i = 1, \ldots, N_t\}$ is called the *cut set*. The weight vector $\mathbf{w}_t$ is defined implicitly as the feasible vector with minimal norm $\mathbf{w}_t = \operatorname{argmin}_{\mathbf{w} \in \mathbf{F}_t} \|\mathbf{w}\|_1$. Although $\mathbf{F}_t$ and $\|\mathbf{w}\|_1$ are convex, the minimum may not be unique since the 1-norm is not a strictly convex function.[1] This does not affect the convergence properties of the algorithm.

The initial cut set $\mathcal{C}_1$ is defined so that $\mathbf{F}_1 = Q$, and by the above definition, the initial solution is $\mathbf{w}_1 = \mathbf{0}$. On the t-th iteration, a single example (unambiguous or ambiguous) denoted $(\mathbf{X}_t, Y_t)$ is presented to the algorithm. An update to the feasible region $\mathbf{F}_t$ is performed when *the example can be used to derive a new cut*, parameterized by (α, β), that properly separates the current solution from $\mathbf{F}$. This is certainly possible if $\mathbf{w}_t \notin \mathbf{D}_i\left(\mathbf{F}_t\right) = \operatorname{conv}\left[\bigcup_{\mathbf{x} \in \mathbf{X}_i} H\left(Y_i \mathbf{x}, 1\right) \cap \mathbf{F}_t\right]$ since it is known that $\mathbf{F} \subseteq \mathbf{D}_i\left(\mathbf{F}_t\right)$ (see Figure 7.2). The problem of deriving such a cut, which is a *separation problem*, is discussed below.

A hyperplane with this separating property is called a *valid cut*. If a valid cut is generated, the cut set is augmented $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{(\alpha_t, \beta_t)\}$, the updated region of feasibility obeys $\mathbf{F}_{t+1} \subset \mathbf{F}_t$. In particular, the new feasible region $\mathbf{F}_{t+1}$ excludes the current solution $\mathbf{w}_t \in \mathbf{F}_t \backslash \mathbf{F}_{t+1}$. The algorithm continues cycling through examples $(\mathbf{X}_t, Y_t)$ from the training data set as long as valid cuts can be generated. Unless the current solution $\mathbf{w}_t$ lies within $\mathbf{F}$, the algorithm will eventually solve a separation problem that generates a valid cut, thereby updating $\mathbf{w}_t$. Due to the role of convexity in each step of the sequence, the process is called *sequential convexification*. The first three steps of the algorithm are depicted in Figures 7.3,7.4,7.5 and 7.6.

To summarize, since $\mathbf{F}_t \supseteq \mathbf{F}$, each iteration solves a *relaxation* of the RDP problem in Equation (7.10), to obtain $\mathbf{w}_t$. Each cut *strengthens* the current relaxation of the RDP problem, and the objective increases monotonically. At convergence, $\mathbf{w}_t \in \mathbf{F}$ and represents a solution to RDP. In order to use CPLEX effectively, the implementation actually solves the dual of the relaxed RDP. From this perspective, the separation sub-problem can be viewed as a *column generation* [1].

[1] Multiple minima exist iff the plane of points p satisfying $\langle \mathbf{1}, p \rangle = \|\mathbf{w}_t\|_1$ intersects $\mathbf{F}_t$ on a d-dimensional face of $\mathbf{F}_t$ for some $d \geq 1$.

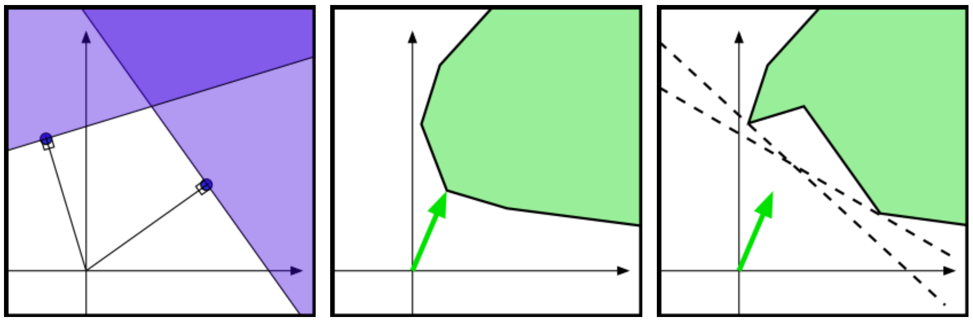


Figure 7.2: The separation problem. (a) Disjunctive feasible region for an ambiguous example with two instances. (b) Current approximation $\mathbf{F}_t$ to feasible region $\mathbf{F}$. (c) Intersection of these regions and two examples of cuts (α, β) separating the intersection from the current solution. Note that the cuts approximate the convex hull of the intersection, which is denoted $\mathbf{D}_i(\mathbf{F}_t) = \operatorname{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap \mathbf{F}_t\right]$.

7.4 Sequential convexification

Our algorithm is based on a technique called sequential convexification that is used to generate the complete feasible region $\mathbf{F}$. This work is related to the sequential convexification approach of [10] and that of [53]. Before proving the convergence, some definitions and preliminary results are required.

7.4.1 Convergence of nested sets

The first definition concerns the closeness of a point to a set.

Definition 20. *For a point x and a set C, let $\operatorname{dist}(x, C) = \min_{\hat{x}\in C} \|\hat{x} - x\|$ be the minimum distance from the point to the set.*

Using the notion of distance between a point and a set, the following convergence of nested sets is defined.

Definition 21. *Let $\{C^t\}$ be a sequence of compact nested sets $C^{t+1} \subseteq C^t$. The sequence $\{C^t\}$* converges *to $\bar{C}$ (i.e. $\lim_{t\to\infty} C^t = \bar{C}$), if:*

1. *$\bar{C} \subseteq C^t$, $\forall t$, and*
2. *$\forall \epsilon > 0$ there exists $\hat{k} \in \mathbb{Z}_+$ such that for all $k \geq \hat{k}$ and $x \in C^k$ the point x is close to $\bar{C}$ in the following sense $\operatorname{dist}\left(x, \bar{C}\right) < \epsilon$.*

The set $\bar{C}$ is called the limit set.

The following lemma will be used in the proof below. The lemma states that the *limit set* of a sequence defined as convex hulls of unions of sets from L convergent sequences, is the convex hull of the union of the L *limit sets* of the sequences. This lemma is a generalized version of a lemma by [53] that concerns unions of $L = 2$ sequences.

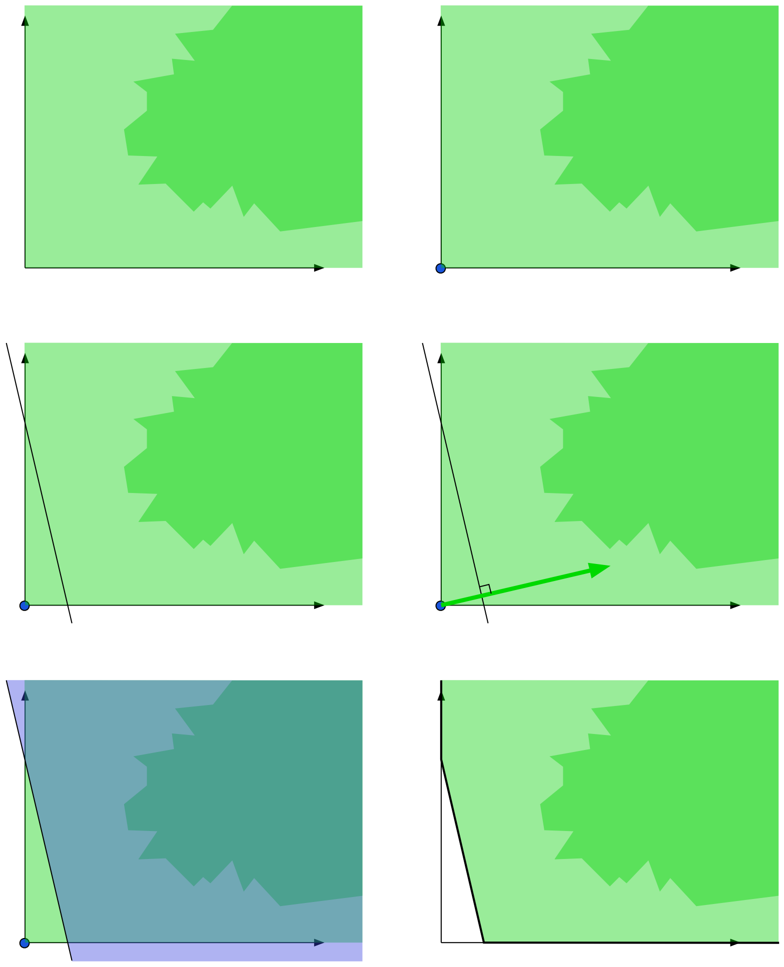


Figure 7.3: Sequential convexification - first iteration. The first row depicts the state of the algorithm prior to making a cut. On the left, the current feasible region is shaded in green, and on the right, the shortest weight vector measured with respect to the 1-norm is drawn as a green arrow. The darkest shaded region corresponds to the true disjunctive feasible region whose convex hull is the desired . The middle row shows how a hyperplane can be used to separate the current solution from the true disjunctive feasible region. The hyperplane is parameterized by a weight vector and offset (α, β). The hyperplane defines a halfspace that can be thought of as the margin constraint of a synthetic example. The bottom row shows how the feasible region is restricted by intersection with the synthetic margin constraint.

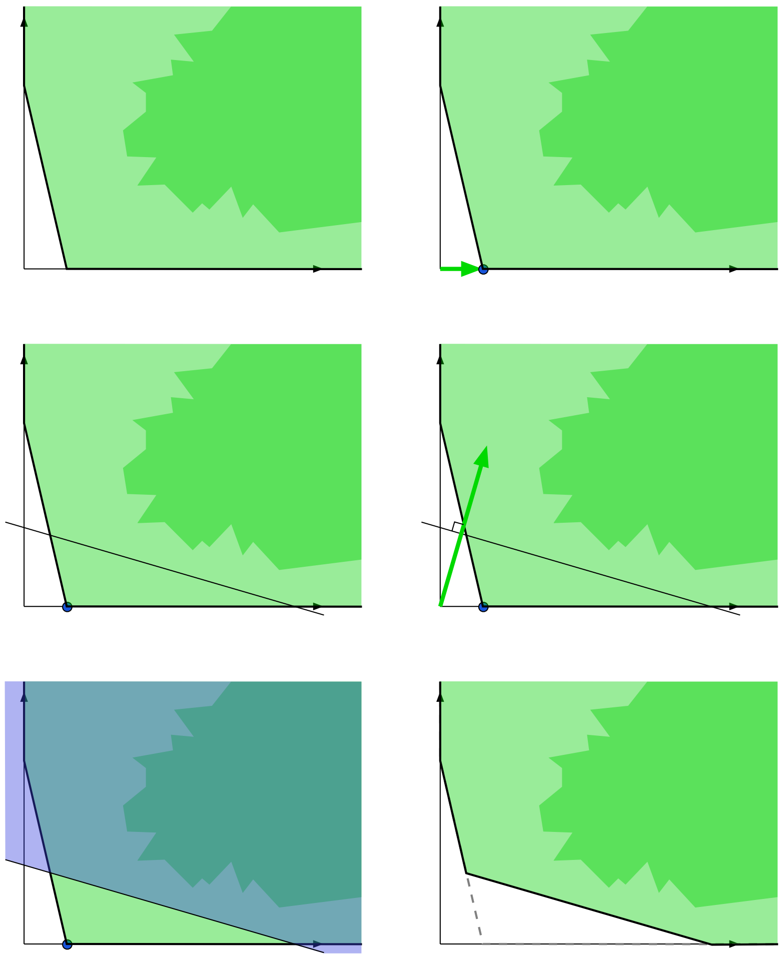


Figure 7.4: Sequential convexification - second iteration. For a description of the sub-figures, refer to Figure 7.3.

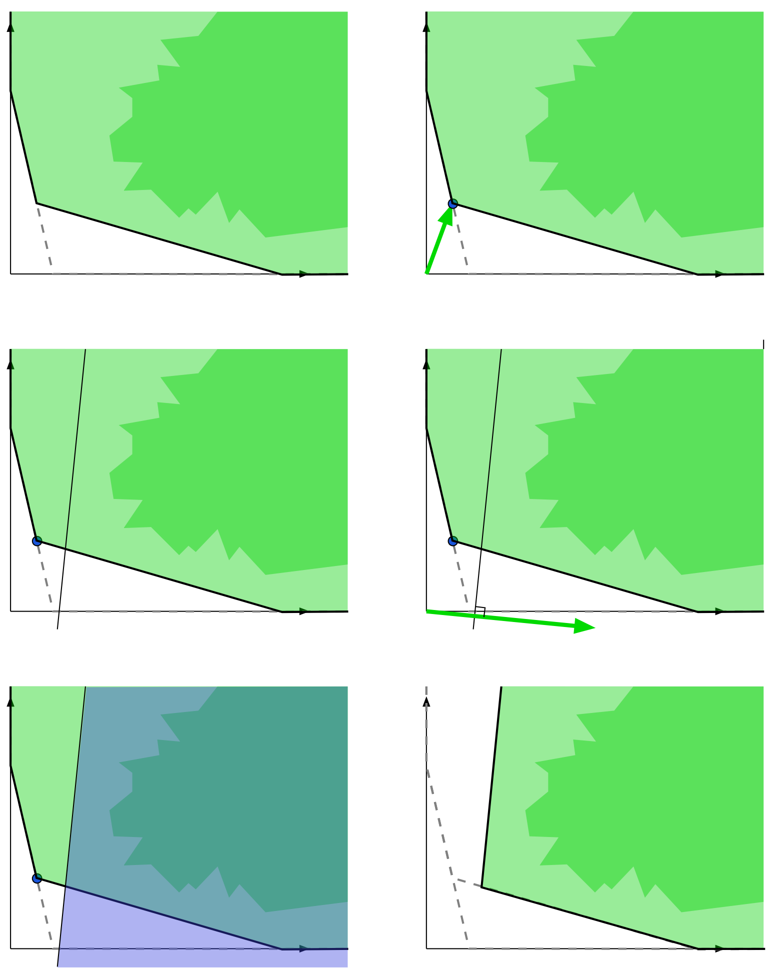


Figure 7.5: Sequential convexification - third iteration. For a description of the sub-figures, refer to Figure 7.3.

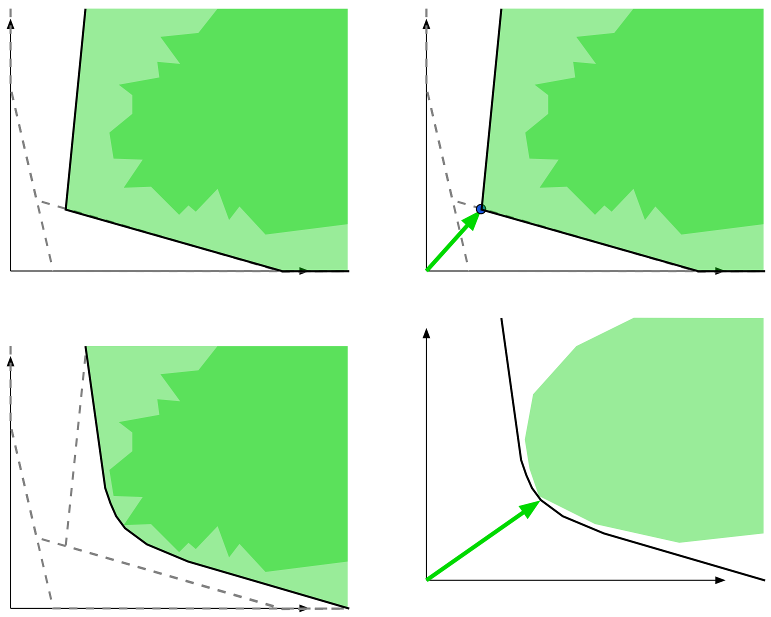


Figure 7.6: Sequential convexification - continued. As the sequential convexification proceeds, the approximation to the current feasible region $\mathbf{F}_t$ improves in the vicinity of optimal solution, as expected for a cutting-plane algorithm [41, 52].

Lemma 1. *Let* $\{C_1^t\},\ldots,\{C_L^t\}$ *be sequences of compact nested sets, which converge to* $\bar{C}_1,\ldots,\bar{C}_L$ *respectively. Then*

$$\lim_{t\to\infty} \operatorname{conv}\left(\bigcup_{k=1}^{L} C_k^t\right) = \operatorname{conv}\left(\bigcup_{k=1}^{L} \bar{C}_k\right).$$

Proof. ($\supseteq$): Let $x \in \operatorname{conv}\left(\bigcup_{k=1}^{L} \bar{C}_k\right)$. Since $\bar{C}_k \subseteq C_k^t\ \forall k \forall t$, this implies $x \in \operatorname{conv}\left(\bigcup_{k=1}^{L} C_k^t\right)$ for all $t \geq 0$, which subsequently implies $x \in \lim_{t\to\infty} \operatorname{conv}\left(\bigcup_{k=1}^{L} C_k^t\right)$.

($\subseteq$): Let $x \in \lim_{t\to\infty} \operatorname{conv}\left(\bigcup_{k=1}^{L} C_k^t\right)$. According to the definition of convergence for nested sequences $\left\{\operatorname{conv}\left(\bigcup_{k=1}^{L} C_k^t\right)\right\}$ this implies that $x \in \operatorname{conv}\left(\bigcup_{k=1}^{L} C_k^t\right)$ for all $t \geq 0$. Thus, by the definition of convexity, there exist sequences $\{x^{t_k}\} \subseteq C_k^t\ \forall k$ and $\{\lambda^{t_k}\} \subseteq [0,1]\ \forall k$ such that $\sum_{k=1}^{L} \lambda^{t_k} = 1$ and $\left\{\sum_{k=1}^{L} \lambda^{t_k} x^{t_k}\right\} \to x$. Since the sequences $\{x^{t_k}\}\ \forall k$ are bounded, they have convergent subsequences $\{x^{t_k}\} \to x^k \in \bar{C}_k\ \forall k$ and $\{\lambda^{t_k}\} \to \lambda^k \in [0,1]$. Thus $x \in \operatorname{conv}\left(\bigcup_{k=1}^{L} \bar{C}_k\right)$.

□

7.4.2 Recursive Approximation

This section provides a detailed theoretical description of the sequential convexification procedure, including a proof of asymptotic convergence. Implementation issues are discussed in Sections 7.4.3 and 7.4.4. Let

$\mathbf{D}_i(K)$ denote the convex hull of the i-th example, constrained to some relaxation set K

$$\mathbf{D}_i(K) \quad = \quad \text{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap K\right] . \tag{7.13}$$

Next, consider the nested recursive sequence $\{P^t\}$ of approximations to $\mathbf{F}$ defined as follows. Let $P^0 = Q$. For $t \geq 0$, let P^{t+1} be defined by the following recursion

$$\begin{aligned} P^{t+1} &= \mathbf{D}_1\left(\mathbf{D}_2\left(\ldots\left(\mathbf{D}_m\left(P^t\right)\right)\ldots\right)\right) & (7.14)\\ &= \text{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_1} H(Y_1\mathbf{x}, 1) \cap \ldots \text{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_m} H(Y_m\mathbf{x}, 1) \cap P^t\right)\ldots\right) . & (7.15)\end{aligned}$$

As is shown next, in the limit the sets defined by this recursive procedure converges to the feasible region $\mathbf{F}$. In other words

$$\lim_{t\to\infty} P^t = \mathbf{F} . \tag{7.16}$$

Practically speaking, this means that, as long as the cutting-plane algorithm can generate valid cutting planes for the set

$$\mathbf{D}_i(K) = \text{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap K\right] \tag{7.17}$$

for any $1 \leq i \leq m$ and for any K, then this result implies that in the limit it will generate the convex hull in the vicinity of the optimal solution.

The sequence $\{P^t\}$ has the following properties:

Proposition 4. *The sequence is nested. In other words $P^{t+1} \subseteq P^t$, $\forall t$.*

Proof. For any sets A and B, the following $\text{conv}(A \cap B) \subseteq \text{conv}(B)$ holds. The results follows from the definition of P^{t+1}. □

Proposition 5. *Elements P^t of the sequence are closed and bounded (i.e. compact).*

Proof. This is because finite intersections and unions of closed sets remain closed. Also, the convex hull of a closed set is closed. Boundedness follows because $P^t \subseteq Q$ and Q is bounded by definition in Equation 7.6. □

Proposition 6. *The feasible set $\mathbf{F} \subseteq P^t$, $\forall t$. In fact, it is also true that $\mathbf{F} \subseteq \mathbf{D}_k\left(\mathbf{D}_{k+1}\left(\ldots\left(\mathbf{D}_m\left(P^t\right)\right)\ldots\right)\right)$ for $k = 1, \ldots, m$ and $\forall t$.*

Proof. A proof by induction follows. Notice that the induction step decreases the k index. To begin, suppose that $\mathbf{F} \subseteq \mathbf{D}_k(\ldots(\mathbf{D}_m(P^t))\ldots)$. Then

$$\begin{aligned}
\mathbf{D}_{k-1}\left(\ldots\left(\mathbf{D}_m\left(P^t\right)\right)\ldots\right) &= \operatorname{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_{k-1}} H\left(Y_{k-1}\mathbf{x},1\right)\cap\mathbf{D}_k\left(\ldots\left(\mathbf{D}_m\left(P^t\right)\right)\ldots\right)\right] && (7.18)\\
&\supseteq \mathbf{F}\cap\operatorname{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_{k-1}} H\left(Y_{k-1}\mathbf{x},1\right)\cap\mathbf{D}_k\left(\ldots\left(\mathbf{D}_m\left(P^t\right)\right)\ldots\right)\right] && (7.19)\\
&\supseteq \operatorname{conv}\left[\bigcap_{i=1}^{m}\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap\mathbf{D}_k\left(\ldots\left(\mathbf{D}_m\left(P^t\right)\right)\ldots\right)\right] && (7.20)\\
&\supseteq \operatorname{conv}\left[\bigcap_{i=1}^{m}\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap\mathbf{F}\right] && (7.21)\\
&\supseteq \mathbf{F}\,. && (7.22)
\end{aligned}$$

The third line follows since $\operatorname{conv}(A\cap B) \subseteq \operatorname{conv}(A)\cap B$ if B is convex, a results that was shown in [7] and was used previously in Equation 6.38. The same essential argument proves that $\mathbf{F} \subseteq \mathbf{D}_m(P^t)$ whenever $\mathbf{F} \subseteq P^t$. The base case is $\mathbf{F} \subseteq P^0 = Q$. This completes the induction argument.

□

Proposition 7. *Successive elements of the sequence are at least as small as that obtained after a single application of* $\mathbf{D}_i(\cdot)$. *In other words*

$$P^{t+1} \subseteq \operatorname{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q\cap P^t\right]\ ,\ \forall t,\ 1\le i\le m\,. \tag{7.23}$$

Proof. Writing the definition of P^{t+1}, and using $\operatorname{conv}(A\cap B)\subseteq\operatorname{conv}(B)$ appropriately, one has

$$\begin{aligned}
P^{t+1} &= \operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_1} H\left(Y_1\mathbf{x},1\right)\cap\ldots\operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_m} H\left(Y_m\mathbf{x},1\right)\cap P^t\right)\ldots\right) && (7.24)\\
&\subseteq \operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_2} H\left(Y_2\mathbf{x},1\right)\cap\ldots\operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_m} H\left(Y_m\mathbf{x},1\right)\cap P^t\right)\ldots\right) && (7.25)\\
&\vdots\\
&\subseteq \operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap\ldots\operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_m} H\left(Y_m\mathbf{x},1\right)\cap P^t\right)\ldots\right)\,. && (7.26)
\end{aligned}$$

In other words, by recursively unwrapping the expression on the right-hand side from the outside and discarding the relevant terms, the set on the right-hand side grows. Similarly, one can apply the same unwrapping from the inside. Since $\operatorname{conv}(A\cap B)\subseteq\operatorname{conv}(B)$, one can discard terms on the inside as well, yielding the desired result

$$\begin{aligned}
P^{t+1} &\subseteq \operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_m\mathbf{x},1\right)\cap\ldots\operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_m} H\left(Y_m\mathbf{x},1\right)\cap P^t\right)\ldots\right) &(7.27)\\
&\subseteq \operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_m\mathbf{x},1\right)\cap\ldots\operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_{m-1}} H\left(Y_{m-1}\mathbf{x},1\right)\cap\operatorname{conv}\left(P^t\right)\right)\ldots\right) &(7.28)\\
&\vdots\\
&\subseteq \operatorname{conv}\left(\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_m\mathbf{x},1\right)\cap\operatorname{conv}\left(P^t\right)\right). &(7.29)
\end{aligned}$$

□

Finally, the main result of this section.

Theorem 1. *Infinite convergence to* **F**. *The sequence of approximations* $\{P^t\}$ *converges to the feasible region* **F**. *In other words,*

$$\lim_{t\to\infty} P^t = \mathbf{F} = \operatorname{conv}\left[\bigcap_{i=1}^{m}\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q\right]. \tag{7.30}$$

Proof. This proof was inspired by a much simpler case considered in [53]. Since the sequence $\{P^t\}$ is nested (Proposition 4), the sets P^t are closed and bounded (Proposition 5), and $\mathbf{F}\subseteq P^t$ (Proposition 6), it is clear that the sequence converges $\{P^t\}\to\hat{P}$ such that $\mathbf{F}\subseteq\hat{P}$.

Now, for any i ($1\le i\le m$), observe that

$$\begin{aligned}
\hat{P} = \lim_{t\to\infty} P^{t+1} &\subseteq \lim_{t\to\infty}\operatorname{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q\cap P^t\right] &(7.31)\\
&\subseteq \operatorname{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q\cap\hat{P}\right]. &(7.32)
\end{aligned}$$

The first inclusion follows from (Proposition 7), while the second inclusion follows from Lemma 1 using the sequences $\{H\left(Y_i\mathbf{x},1\right)\cap Q\cap P^t\}$, $\forall\mathbf{x}\in\mathbf{X}_i$.

Therefore, one can conclude that the extreme points of $\hat{P}$ lie in $\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q$, for all i. Otherwise, a contradiction arises: if any extreme point was excluded, then by the properties of convex sets [61], it could not be a member of conv $\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q\cap\hat{P}\right]$, which contradicts Equation (7.32). In other words ext $\left(\hat{P}\right)\subseteq\bigcap_{i=1}^{m}\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q$. This, in turn, implies the final result

$$\hat{P} = \operatorname{conv}\operatorname{ext}\hat{P}\subseteq\operatorname{conv}\left[\bigcap_{i=1}^{m}\bigcup_{\mathbf{x}\in\mathbf{X}_i} H\left(Y_i\mathbf{x},1\right)\cap Q\right] = \mathbf{F}. \tag{7.33}$$

. □

7.4.3 Separation by lift-and-project cuts

In order to generate the cuts for a given solution $\mathbf{w}_t$ and an **ambiguous** training example $(\mathbf{X}_t, Y_t)$ with the current feasible region $\mathbf{F}_t$, the algorithm considers a family of lift-and-project cuts. These are related to the lift-and-project cuts introduced by [9] for 0-1 programming.

Lemma 2. *Let $(\alpha_0^{\mathbf{x}}, \beta_0^{\mathbf{x}}) = (Y_t\mathbf{x}, 1)$ for all $\mathbf{x} \in \mathbf{X}_t$. The cut defined by $(\alpha, \beta) \in \mathbb{R}^{d+1}$ is a valid cut if and only if, for all $\mathbf{x} \in \mathbf{X}_t$, there exists positive multipliers $(u^{\mathbf{x}}, u_0^{\mathbf{x}}) \in \mathbb{R}^N \times \mathbb{R}$ such that the following linear constraints are satisfied*

$$\alpha \geq \sum_{i=1}^{N} u_i^{\mathbf{x}} \alpha_i + u_0^{\mathbf{x}} \alpha_0^{\mathbf{x}} \tag{7.34}$$

$$\beta \leq \sum_{i=1}^{N} u_i^{\mathbf{x}} \beta_i + u_0^{\mathbf{x}} \beta_0^{\mathbf{x}} \tag{7.35}$$

This characterization of valid cuts for $\mathbf{D}_t(\mathbf{F}_t)$ is a direct consequence of Farkas lemma [9, 61]. A lift-and-project cut is generated by maximizing the separation achieved by a valid cut.

Definition 22. *The* signed depth of a cut (α_t, β_t) *is defined to be the signed functional distance from the solution $\mathbf{w}_t$ to the halfspace $H(\alpha_t, \beta_t)$, which is $\beta_t - \langle \alpha_t, \mathbf{w}_t \rangle$ as derived in [46].*

Maximizing the depth over valid cuts can be formulated as an LP over cut parameters (α, β) and multipliers $(u^{\mathbf{x}}, u_0^{\mathbf{x}})$. In order to ensure that the LP has a finite optimum the coefficients are bounded $\|\alpha\|_1 \leq C_{\text{LNP}}$, where C_{LNP} controls the complexity of the cut, as suggested in [9]. This completes the definition of the lift-and-project cut generating LP.

Definition 23. *The proposed lift-and-project cut for learning from ambiguous examples is defined as the solution of the following linear program.*

$$\textit{LNP} \quad \max_{\beta, \alpha, u^{\mathbf{x}}} \beta - \langle \alpha, \mathbf{w}_t \rangle \quad \textit{s.t.} \quad \|\alpha\|_1 \leq C_{\textit{LNP}} \tag{7.36}$$

$$(\alpha, \beta) \ \textit{is valid.} \tag{7.37}$$

If the multi-instances are restricted to $\mathbf{X}_i = \{-\mathbf{e}_i, \mathbf{e}_i\}$ where $\mathbf{e}_i$ is the standard unit vector along the i-th dimension, then this family of lift-and-project cuts reduces to the set proposed for 0-1 programming by [10].

When the depth is positive, a valid cut exists. The converse is also true: if the point $\mathbf{w}_t \notin \mathbf{D}_t(\mathbf{F}_t)$, the convexity of $\mathbf{D}_t(\mathbf{F}_t)$ guarantees that a valid cut exists (Corollary 11.4.2 pg. 99, [61]). On the other hand, if the depth is negative, then it is clear that $\mathbf{w}_t \in \mathbf{D}_t(\mathbf{F}_t)$. The solution (α, β) can be thought of as the cut that would have been generated by the margin constraint of an unambiguous training example $\left(\left(\frac{Y}{\beta}\alpha\right), Y\right)$.

Note that the cut defined by $(\alpha_t, \beta_t) = (y_t\mathbf{x}_t, 1)$ for the **unambiguous** example $(\mathbf{x}_t, y_t)$ is valid and has depth equal to the standard hinge loss, as long as $\mathbf{w} \notin H(y_t\mathbf{x}_t, 1)$. These are the cuts used by the ROMMA algorithm [44].

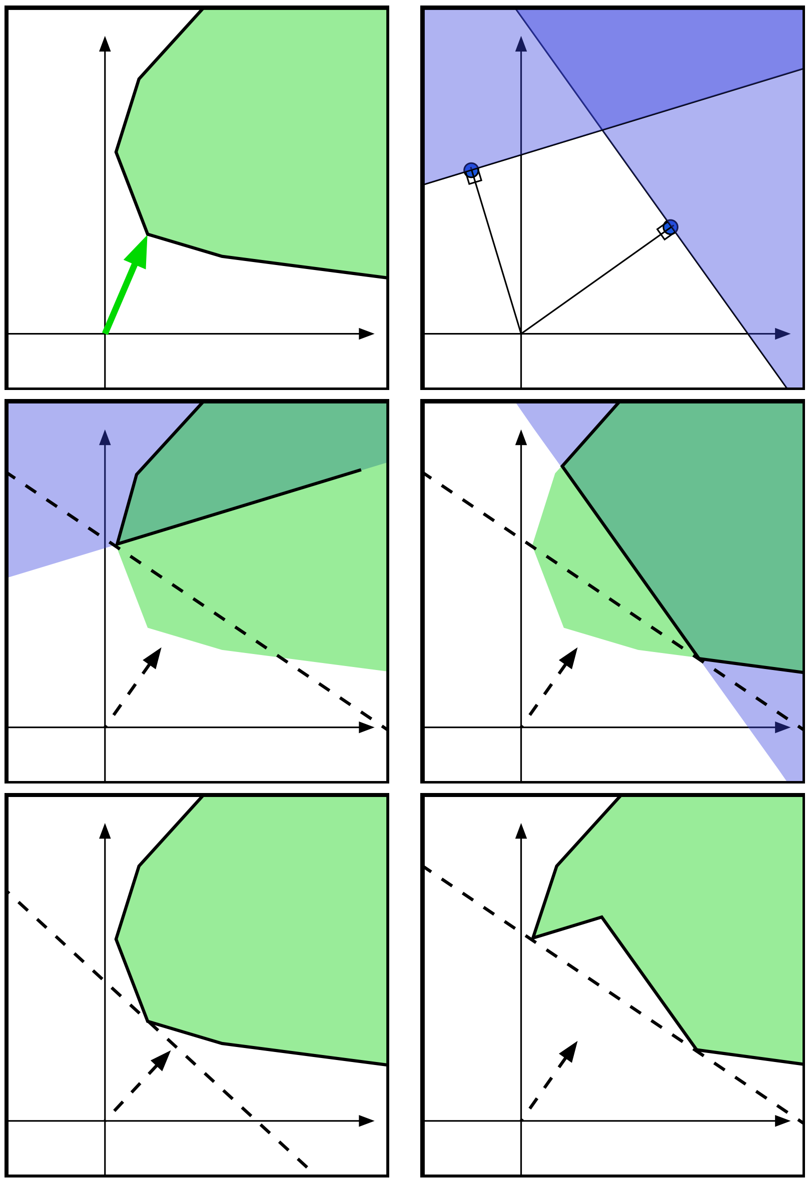
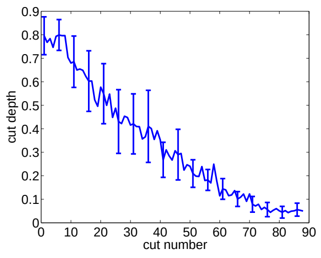
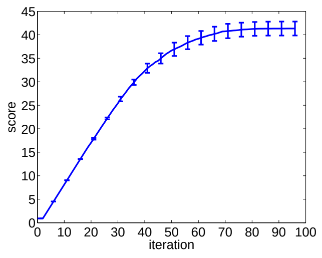


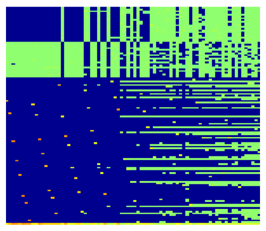
Figure 7.7: The LNP cut is a supporting hyperplane of the convex hull of the intersection $\mathbf{D}_i(\mathbf{F}_t) = \operatorname{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap \mathbf{F}_t\right]$. The top row depicts the current feasible region $\mathbf{F}_t$ and the i-th disjunctive margin constraint; a union of two halfspaces. Necessary and sufficient conditions for such a supporting hyperplane are specified in Lemma 2. Essentially, the conditions of the lemma state that the hyperplane must also support $k = |\mathbf{X}_i|$ other convex regions. These are the convex sets outlined in black in the second row, one for each $\mathbf{x} \in \mathbf{X}_i$, resulting from the intersection of $\mathbf{F}_t$ with the halfspace $H(Y_i\mathbf{x}, 1)$. The dashed lines and vectors in the second row represent supporting hyperplanes and normals for these convex sets. In order to select a supporting hyperplane that excludes as much of the current approximation $\mathbf{F}_t$ as possible, the LNP cut maximizes the distance between the hyperplane and the current solution $\mathbf{w}_t$. The bottom row shows (left) a depth-zero supporting hyperplane for $\mathbf{F}_t$, and (right) the deepest supporting hyperplane for $\mathbf{D}_i$.



(a)



(b)



(c)

Figure 7.8: Analysis of the cutting-plane algorithm on MUSK1. Error bars show 95% confidence intervals over 10 folds. (a) Cut depth. (b) Value of objective. (c) Generated cuts (columns, L to R). Upper bands correspond to features, while lower bands correspond to slack variables. Initial cuts are typically derived from unambiguous examples, including negatively labeled multi-instances. Ambiguous examples play a greater role in the later stages. A subset of slack variables become active in the later cuts, indicating which examples have tight margin constraints.

7.4.4 Convergence and cut set regularization

For 0-1 mixed-integer programming problems, there exists a proof showing that the sequential convexification procedure converges in a *finite* number of steps [9]. Unfortunately, this does not hold for general disjunctive programs. At this time, the theoretically proved asymptotic convergence will be demonstrated empirically.

There is a trade-off between the size of the cut-set, the accuracy of the resulting classifier, and the running time, that may be controlled using various conditions. The size of the cut-set has an significant impact on running time, since the LNP sub-problem constitutes the majority of computation in the entire sequential convexification procedure. However, if our conditions are too strict, then the approximation $\mathbf{F}_t$ will be poor and classifier accuracy will drop. The following two techniques, suggested by [9], were implemented for regulating the size of the cut-set.

The first technique is to avoid shallow cuts in favor of subsequent deeper ones. This idea is implemented by generating LNP cuts in rounds of 10 from which the single deepest cut is applied; the remaining cuts are discarded. Also, a cut selected for the round is discarded if its depth measured relative to the current objective is below a specified tolerance. The second technique is to avoid redundancy due to identical or near parallel cuts. The generation of near-parallel cuts especially near convergence, is a commonly observed behavior and the main cause of slow convergence of most cutting-plane algorithms [13, 1]. To alleviate this problem, the algorithm only adds a cut to the cut-set when the angle made between it and any of the previous cutting-planes of the cut-set is sufficiently large. Empirically, the minimum-angle tolerance has a direct and noticeable effect on both the running time of the overall procedure and the accuracy of the resulting classifier.

The procedure is terminated if the relative increase of the objective falls below a specified tolerance. The steps of the sequential convexification algorithm are depicted in algorithm box 7.1. Figure 7.8 provides a diagnostic view of the running of the cutting-plane algorithm.

7.4.5 Projection

The objective of the proposed technique is the solution of DP in Equation (7.7). As argued above, an exact solution to RDP will correspond to the global minimum of DP.

Algorithm 7.1 Sequential convexification algorithm [SC]

```
1:  input: $\mathcal{D}, \delta_A, \delta_D, \delta_S, N_{\text{LNP}}$
2:  initialize: $\mathcal{C}_1 \leftarrow \{(\mathbf{e}_k, 0) : k = 1, \ldots, d\}$,
    $\mathbf{F}_1 \leftarrow Q$, $\mathbf{w}_1 \leftarrow \operatorname{argmin}_{\mathbf{w} \in \mathbf{F}_1} \|\mathbf{w}\|_1$
3:  repeat
4:     $(\alpha_t, \beta_t) \leftarrow (0, 0)$, $D_t \leftarrow 0$
5:     for $i = 1, \ldots, N_{\text{LNP}}$ do
6:        Get training example $(\mathbf{X}_t, Y_t)$
7:        $(\alpha_i, \beta_i) \leftarrow \text{LNP}(\mathbf{w}_t, \mathbf{F}_t, \mathbf{X}_t, Y_t)$
8:        if $\min_{(\alpha,\beta) \in \mathcal{C}_t} \measuredangle((\alpha_i, \beta_i), (\alpha, \beta)) \geq \delta_A$ then
9:           if depth $((\alpha_i, \beta_i); \mathbf{w}_t) \geq D_t$ then
10:             $(\alpha_t, \beta_t) \leftarrow (\alpha_i, \beta_i)$
11:             $D_t \leftarrow$ depth $((\alpha_t, \beta_t); \mathbf{w}_t)$
12:          end if
13:       end if
14:    end for
15:    if $D_t \geq \delta_D * \|\mathbf{w}_t\|_1$ then
16:       $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{(\alpha_t, \beta_t)\}$, $\mathbf{F}_{t+1} \leftarrow \mathbf{F}_t \cap H(\alpha_t, \beta_t)$
17:       $\mathbf{w}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{w} \in \mathbf{F}_{t+1}} \|\mathbf{w}\|_1$
18:    end if
19:    $t \leftarrow t + 1$
20: until $\|\mathbf{w}_{t+1}\|_1 - \|\mathbf{w}_t\|_1 < \delta_S * \|\mathbf{w}_t\|_1$
```

Unfortunately, solving RDP exactly is time consuming. While the cutting-plane algorithm converges in the limit to a point in $\mathbf{F}$, the algorithm terminates before this point. Since the sequential approximation $\mathbf{F}_t$ is larger than $\mathbf{F}$ ($\mathbf{F} \subseteq \mathbf{F}_t$, $\forall t$), the shortest vector in $\mathbf{F}_T$ may not belong to $\mathbf{F}$ at convergence $t = T$. Therefore, the algorithm does not even guarantee that a true feasible point is found, let alone the global minimum. A simple projection operation, however, can be applied to obtain feasibility.

Projection maps the current solution, which lies in the convex approximation $\mathbf{F}$, onto the original region of feasibility $\bigcap_{i=1}^{m} \bigcup_{\mathbf{x} \in \mathbf{X}_i} H(Y_i \mathbf{x}, 1) \cap Q$. As a result, a feasible point for the original DP is obtained. This means that all multi-instance margin constraints are satisfied. Moreover, the projection identifies specific interpretations $(\mathbf{x}, Y_i)$ with $\mathbf{x} \in \mathbf{X}_i$ of the ambiguous examples. This form of disambiguation is the same as that used for MI-SVM from Chapter 5.

Specifically, to compute the projected solution, the algorithm first selects the most positive instance $\mathbf{x}_i = \operatorname{argmax}_{\mathbf{x} \in \mathbf{X}_i} f(\mathbf{x})$ of each multi-instance using the solution $f(\mathbf{x})$ of RDP. Discarding the remaining instances, the algorithm creates a data set of *unambiguous* examples $\mathcal{D}^* = \{(\mathbf{x}_i, Y_i) \,|\, i = 1, \ldots, m\}$. The projection from the RDP solution is then obtained by solving the following restricted version of DP for the new weight vector $\mathbf{w}$

$$\text{PRDP} \qquad \min_{\mathbf{w}} \|\mathbf{w}\|_1 \tag{7.38}$$

$$\text{s.t.} \quad \mathbf{w} \in \bigcap_{i=1}^{m} H(Y_i \mathbf{x}_i, 1) \cap Q\,. \tag{7.39}$$

This is equivalent to the 1-norm SVM formulation using $\mathcal{D}^*$. It tuns out that the performance of the PRDP solution clearly exceeds that of the RDP solution (see Figure 7.9(a)). There is no guarantee that the projected feasible points will be global minima of DP. The objective values at the solution of RDP and PRDP provide lower and upper bounds on the global minimum of DP. The difference in these values is called the gap. If the gap is relatively small, then the solutions must be close to the solution of DP.

This completes the description of the basic LNPBoost algorithm. A short-coming of this algorithm is that the LNPBoost sub-problem will take a long time to solve when the number of features is large, even when the approximation $\mathbf{F}_t$ fits in memory. The algorithm is extended by a feature selection procedure in Section 7.6 below. Before doing so, a comparison of LNPBoost and MI-SVM from Chapter 5 is given.

7.5 A comparison with MI-SVM

Disregarding the issue of classifier performance, one might wonder how well does LNPBoost perform purely as an optimization technique. In order to evaluate the approximate solution to DP returned by LNPBoost, a comparison with the greedy local search strategy of MI-SVM is performed. To make the comparison valid, MI-SVM is adapted to the 1-norm maximum-margin problem.

7.5.1 1-norm MI-SVM

A new algorithm called 1-norm MI-SVM is proposed. The goal of this algorithm is to solve the 1-norm DP formulation introduced in this chapter. The 2-stage mixed-integer heuristic optimization strategy presented in Chapter 5, Algorithm 5.1 is unchanged.

Specifically, the proposed 1-norm MI-SVM algorithm introduces integer variables $1 \leq z(i) \leq |\mathbf{X}_i|$ to select the "most positive" member $\mathbf{x}_{z(i)} \in \mathbf{X}_i$. The resulting formulation is identical to Equations (5.24-5.27) apart from the linear objective. The 2-stage local search heuristic alternates the following steps: (i) for given integer variables, solve the associated LP and find the optimal hyperplane, (ii) for a given hyperplane, update the integer variables in a way that (locally) minimizes the objective. In the current implementation of 1-norm MI-SVM, the asynchronous update strategy was used. This strategy selects one integer variable to be updated each round based on the difference $\max_{\mathbf{x} \in \mathbf{X}_i} f(\mathbf{x}) - f\left(\mathbf{x}_{z(i)}\right)$. The "most positive" instances are initialized randomly. The algorithm terminates when a local minimum is detected; i.e. when the subsequent integer variable update fails to decrease the objective value. Since there are a finite number of disambiguations, the algorithm terminates in a finite number of steps.

On a practical level, 1-norm MI-SVM can be implemented efficiently. It is convenient to represent and manage the dual LP, due to the column handling routines provided by linear programming software packages. The update of integer variable $z(i)$ affects only one column of the constraint matrix. Saving additional information about the solution (the basis) from previous iterations allows for efficient re-optimization in subsequent rounds.

This completes the description of the basic 1-norm MI-SVM algorithm. The 1-norm MI-SVM is susceptible to local minima, just as MI-SVM was. Another short-coming of 1-norm MI-SVM algorithm is that the constraint matrix will not fit in memory when the number of features is large. For this reason, the basic 1-norm MI-SVM algorithm variant can not be applied to the TREC9 data sets, or the MUSK2 data set when using RBF empirical feature maps. This algorithm can also be extended with feature selection, as described next.

7.6 Feature selection

Feature selection is employed to make the algorithms practical for problems with a large feature sets (e.g. $d < 500$). The proceeding discussion concerns the extension of the sequential convexification algorithm.

As described, the algorithm can only be applied to data sets with a relatively small number of features. This is because the constraint set that represents the sequential approximation $\mathbf{F}_t$ after t iterations is $t \times d$, and the time and memory required to generate cuts via the LNP sub-problem increases with d. However, for problems with large feature sets, such computation is wasteful because there is typically only a small fraction of features that participate in the solution. The idea of feature selection is to identify the relevant subset of features. In this way, the computation incurred by sequential convexification occurs over low-dimensional constraint sets.

Feature selection is performed using the approach of LPBoost [26], described in Section 6.1.6. This can be performed using the projected solution. The same essential idea was also used for feature selection in DPBoost, Section 6.2.8. Features are added when they help to minimize the PRDP upper bound. An important characteristic of the solution to PRDP in Equation (7.39) is the fact that the dual variables u_i are now associated with *instances* in the training data set, instead of *cutting-planes*. This allows one to compute a *feature score* for the unused features that are associated with these instances

$$\text{score}\,(k; u, \mathcal{D}) = \sum_{i=1}^{m} u_i Y_i h_k\,(\mathbf{x}) - 1 \tag{7.40}$$

where the notation $h_*\,(\mathbf{x})$ is used to denote the projection of $\mathbf{x}$ onto the indicated feature. To initialize the procedure, the algorithm begins with a small randomly selected subset of the features (typically 5), denoted by A_T. After solving the PRDP and retrieving its dual variables, the algorithm evaluates the score equation and select new features (typically 2) with greatest score $(k; u, \mathcal{D}) \geq 0$. A detailed account is found in [26]. Box 7.2 shows the augmented algorithm. Figure 7.9 provides a detailed view of the inner workings of the algorithm.

In order to extend the 1-norm MI-SVM algorithm, one can simply evaluate the feature score for the unused features directly, because the dual variables u_i are already associated with *instances* (the “most positive” instances).

With feature selection, both algorithms can be successfully applied to the TREC9 data sets and the MUSK2 data set using the RBF empirical feature map.

Algorithm 7.2 Feature selection algorithm. Steps SC and PRDP are computed using only features from A_T.

```
1:  input: $\mathcal{D}, \delta_A, \delta_D, \delta_S, N_{\text{LNP}}$
2:  initialize: $A_1 \leftarrow random,\ \mathbf{w}_0 \leftarrow \inf$
3:  repeat
4:      $\mathbf{w}_T \leftarrow \text{SC}\left(\mathcal{D}, \delta_A, \delta_D, \delta_S, N_{\text{LNP}}; A_T\right)$
5:      $\mathcal{D}^* \leftarrow \left\{\left(\operatorname{argmax}_{\mathbf{x}\in\mathbf{X}_i} \langle \mathbf{w}_T, h_{A_T}(\mathbf{x})\rangle, Y_i\right) \mid \forall i\right\}$
6:      $u \leftarrow \text{PRDP}\left(\mathcal{D}^*;\ A_T\right)$
7:      $k \leftarrow \operatorname{argmax}_{k\in \bar{A_T}} \text{score}\left(k; u, \mathcal{D}\right)$
8:      $A_{T+1} \leftarrow A_T \cup \{k\}$
9:      $T \leftarrow T+1$
10: until $\|\mathbf{w}_{T-1}\|_1 - \|\mathbf{w}_T\|_1 < \delta_c * \|\mathbf{w}_{T-1}\|_1$
```

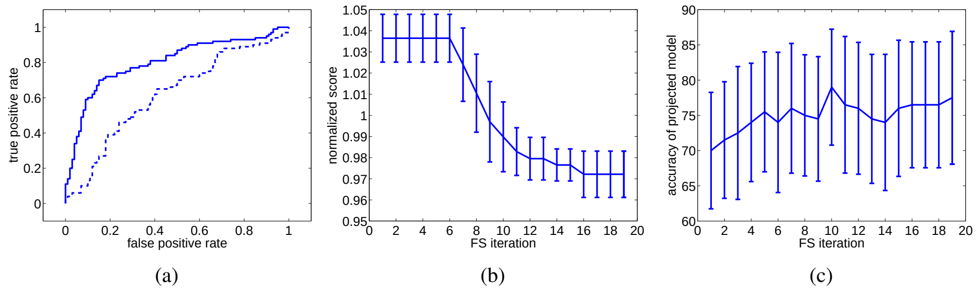


Figure 7.9: Analysis of the effect of projection and boosting on the Elephant data set. (a) ROC curves for RDP (broken) and PRDP (solid) classifiers at convergence of cutting-plane algorithm. (b) RDP objective value vs. boosting round. (c) PRDP accuracy vs. boosting round. Error bars show 95% confidence intervals obtained over 10-folds.

7.7 Experiments

7.7.1 Methodology

Algorithms were implemented in MATLAB[2] using the Java[3] interface in order to solve LP sub-problems using CPLEX[4]. Algorithms were tested on multi-instance learning and transductive learning benchmarks. For MIL, two MUSK data sets, in addition to three Corel image data sets (Tiger, Elephant, Fox) and seven TREC9 text data sets (1,2,3,4,7,9,10) were used. This is an impressive, large, diverse collection of multi-instance data sets. For transductive learning, the USPS handwritten digits data set was used. Details can be found in Chapter 2.

For experiments, 10-fold crossvalidations were used to evaluate two variants of the algorithms: one with linear features, and one with a radial basis function (RBF) features. In both cases, features were preprocessed following [73]. First, each feature was centered and normalized so that its variance on the training data was 1. For the RBF empirical feature map, the inputs were subsequently transformed as follows $\mathbf{x} \mapsto (h_1(\mathbf{x}), \ldots, h_M(\mathbf{x}))$, where $h_j(\mathbf{x}) = \exp\left(-\|\mathbf{x}-\mathbf{x}_j\|^2/\sigma^2\right)$, $\forall j$, and $\mathbf{x}_j$ is the j-th individual from the training data set. One set of experiments was performed with $C = 1$ and another with $C = 0.5$. All

[2]http://www.mathworks.com

[3]http://java.sun.com

[4]http://www.ilog.com

experiments used default parameter settings otherwise: $C_{\text{balance}} = 0.5$, $C_{\text{LNP}} = 1$, $N_{LNP} = 10$, $\delta_D = 10^{-5}$, $\delta_S = 10^{-4}$ and $\sigma \approx \sqrt{d}$. A sequence of experiments with various minimum angle tolerances $\delta_A \in [0, \pi/2]$ was performed for each data set. For boosting, 5 randomly selected features were used for initialization, and 2 features were added in each boosting round. For LNPBoost, the accuracy of the final PRDP solution (i.e. after projection) is reported.

To test whether the performance differences between the compared algorithms are significant, a student's t-test was used. For two independent samples that are assumed to be normally distributed with unequal standard deviations, the student's t-test assesses whether the means are statistically different. The test is able to reject the null hypothesis stating that the means of the distributions are the same, if the computed t-value is larger than a threshold chosen from the appropriate significance table for 95% confidence. This means that there is a 5% chance of observing a significant difference even if the means are the same.

In total there are results for 6 algorithms: EM-DD, MI-SVM, mi-SVM, 1-norm MI-SVM, 1-norm MI-SVM with feature selection, and LNPBoost. These are the only algorithms that have been evaluated on all 12 data sets. The parameter settings and results of all experiments are tabulated in Section 7.9. These results are analyzed in detail in the following sections.

7.7.2 Analysis of LNPBoost

Note that the running times for LNPBoost were between 30 minutes - 3 hours per fold on 1.7 GHz Athlon MP processors for all reported experiments. Over 90% of the time is spent solving LNP sub-problem instances. Fortunately, advances in combinatorial optimization indicate that LNP cuts can be computed much more efficiently [11]. Also, notice that the variance of the accuracies reported for LNPBoost are high. It is hypothesized that his is due to the heuristic projection method used to obtain the PRDP classifier. This could be improved using alternative projections or by combining multiple projections as suggested in [1].

To gain more intuition for the operation of the LNPBoost algorithm, several statistics were measured and studied on the MUSK and Corel benchmark data sets. These trends are evident in the Tables 7.16 and 7.17 of Section 7.9.

In particular, the following quantities are studied: number of features used, running time, RDP score (objective value) and accuracy, and PRDP score and accuracy. The variations in these quantities are studied as the minimum-angle tolerance is decreased. Recall that the minimum-angle tolerance, as described in Section 7.4.4, is a critical parameter for controlling the efficiency of the algorithm by restricting the cuts that are added to the approximation $\mathbf{F}_t$. The value used for the minimum-angle tolerance δ_A can be inferred from the value $\cos(\delta_A)$ that is listed under the "maximum-cosine tolerance" column.

Note that the number of features quantity counts the number of features added to the optimization by the feature selection procedure. This does not imply that all of these features are used in the solution. For example with the TREC9 data sets there are on average seven non-zero features in the projected solutions.

These trends are observed across several experiment sequences. Notice in Tables 7.16 and 7.17 that several experiments are performed for each parameter setting and data set; only the minimum-angle tolerance parameter varies. To analyze the variations in these quantities, a plot of the quantity versus the minimum-angle tolerance (maximum-cosine tolerance) is used. However, instead of viewing each sequence individually, the trends are observed by normalizing and aligning these sequences. For each sequence and each quantity, the range of values is scaled to have a minimum and maximum value of zero and one respectively. The same normalization is performed for the range of maximum-cosine tolerances. In this way the resulting curves may be sub-sampled and averaged. This procedure is performed for all Corel sequences and all MUSK sequences separately, resulting in Figures 7.10 and 7.11 respectively.

A large minimum-angle tolerance will selectively limit the number of cuts added to $\mathbf{F}_t$, thereby limiting the ability of the cutting-plane algorithm to converge to $\mathbf{F}$. The algorithm will converge quickly because the repeated LNP cut-generation sub-problems involve small numbers of constraints only. A smaller minimum-angle tolerance implies that more cuts will be added to $\mathbf{F}_t$, and the running time will increase accordingly. These behaviours are observed in the top row of Figures 7.10 and 7.11. The number of rounds of feature selection also varies. For the Corel data sets, there is a clear increasing trend in the number of features used as the minimum-angle tolerance decreases, while for the MUSK data sets, the increasing trend is not observed. The hypothesis is that, as the approximation $\mathbf{F}_T$ becomes more complex, so does the combination of non-zero dual variables u_i in the projected solution, and therefore a larger number of features will have positive scores.

As the minimum-angle tolerance decreases, the approximation $\mathbf{F}_T$ at convergence will be smaller and should provide a closer approximation to $\mathbf{F}$ near the optimal solution. This is reflected by an increase in the RDP score (the 1-norm of the weight vector) that is also observed in the figures. In theory, as the minimum-angle tolerance decreases, the solution to RDP should approach the true global minimum of DP. Therefore, one might expect that the projected solutions, which upper-bound the DP objective, approach the global minimum of DP *from above*, and so the PRDP scores should decrease. The PRDP score initially decreases in both cases. For the Corel data sets, this curve levels off as the minimum-angle tolerance continues to decrease. The leveling-off suggests that there are many similar-valued feasible points that arise from alternate disambiguations close to the global minimum. On the other hand, for the MUSK data sets, this value makes an unexpected jump before leveling off. It is possible that the projected solutions do not follow the expected trend when the RDP solution is very far from the true DP solution.

As observed in the tables the RDP score is indeed significantly lower than PRDP score. This agrees with the theory because RDP is a relaxation of DP, and PRDP is a restriction of DP.

As the approximation converges to the $\mathbf{F}$, the solution approaches the true minimum of DP and corresponding low-risk classifier motivated by the MI-SRM principle. The hope is that improving the quality of the approximation will also improve the quality of the solution in terms of the accuracy obtained on the test set. The bottom row of figures shows the corresponding behaviour of the resulting classifiers before and after projection. The accuracy of the RDP classifier generally increases. More significant is the increasing trend observed for the corresponding PRDP classifier. This indicates an interesting behaviour of the LNPBoost algorithm. While the PRDP score is mostly unchanging for smaller minimum-angle tolerances (the right side of the plots), the accuracy of the corresponding PRDP classifier increases. This suggests that the LNPBoost algorithm is able to select among otherwise indistinguishable feasible points as it tries to find the global minimum. This is a behavior not shared by the MI-SVM algorithms which can get stuck in local minima.

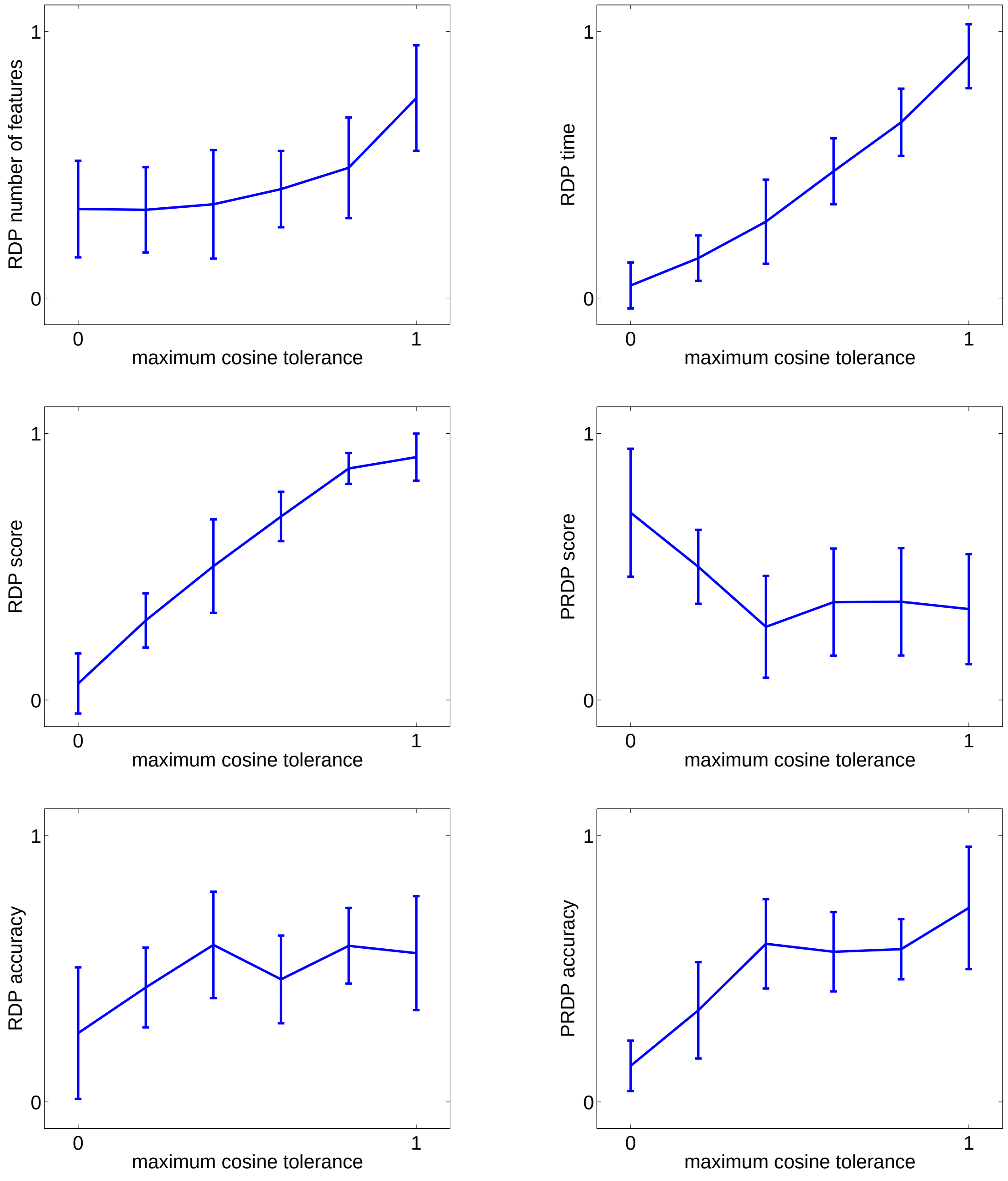


Figure 7.10: Trends of LNPBoost on Corel data sets.

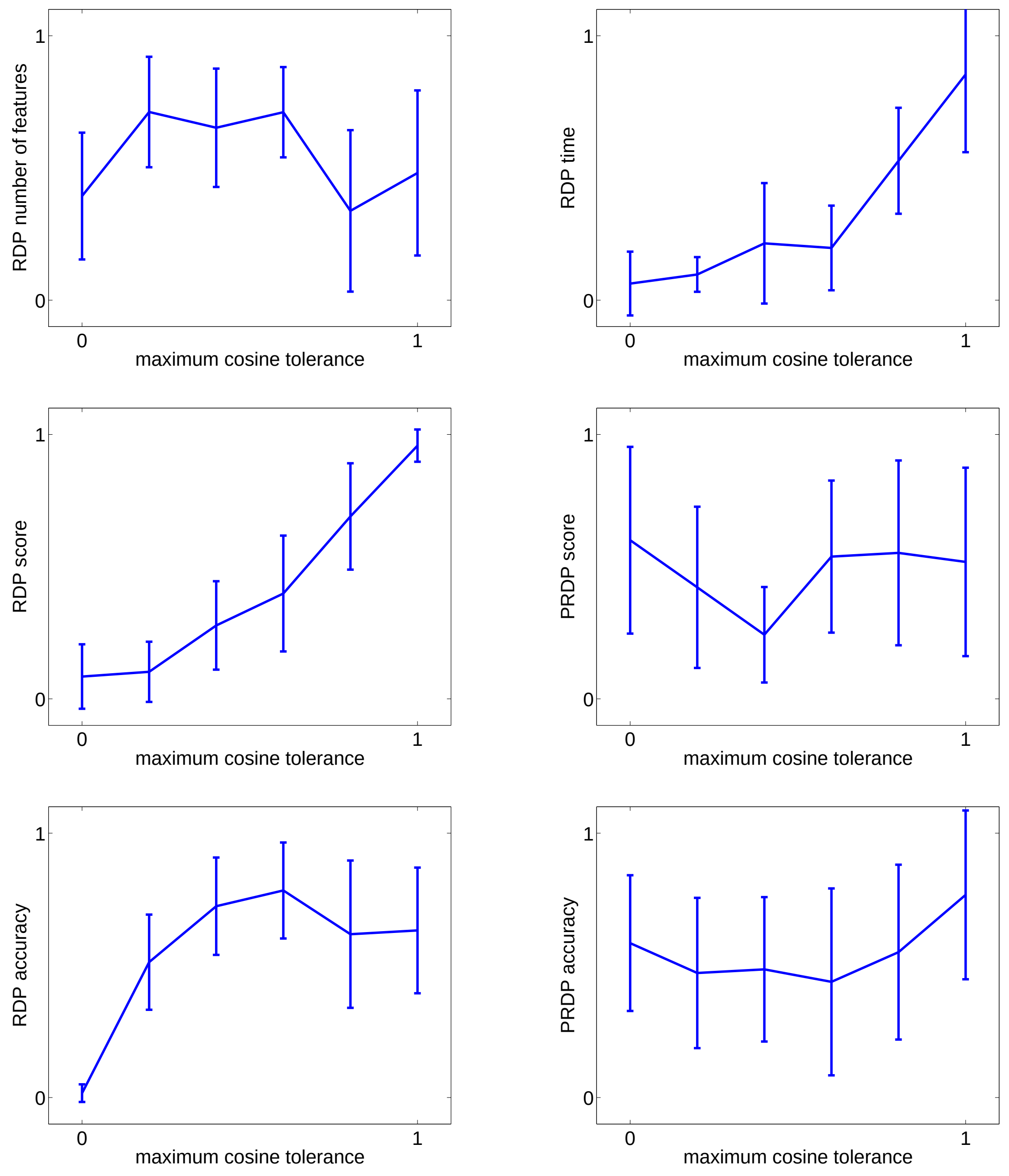


Figure 7.11: Trends of LNPBoost on MUSK data sets.

Data Set	EM-DD		LNPBoost Linear			EM-DD		LNPBoost RBF		
Category	mean	std.	mean	std.	p-value	mean	std.	mean	std.	p-value
MUSK1	**84.8**	3.1	80.0	7.0	0.07	**84.8**	3.1	84.4	9.4	0.90
MUSK2	**84.9**	7.2	69.0	9.9	0.00	84.9	7.2	**85.0**	10.8	0.98
Elephant	78.3	3.5	**85.0**	11.1	0.10	**78.3**	3.5	77.5	13.2	0.86
Fox	56.1	3.0	**57.5**	8.6	0.63	56.1	3.0	**59.5**	6.4	0.15
Tiger	72.1	2.8	**79.5**	8.3	0.02	72.1	2.8	**75.0**	13.3	0.51
TST1	85.8	2.0	**94.8**	4.2	0.00	-				
TST2	84.0	2.0	**86.0**	3.6	0.15	-				
TST3	69.0	2.0	**90.2**	5.5	0.00	-				
TST4	80.5	2.0	**82.5**	8.4	0.48	-				
TST7	75.4	2.0	**78.5**	5.0	0.10	-				
TST9	**65.5**	2.0	60.0	6.7	0.03	-				
TST10	78.5	2.0	**82.3**	6.5	0.11	-				

Table 7.1: Comparison of LNPBoost (linear and RBF) with EM-DD. 10-fold crossvalidation accuracies and p-values from t-tests are shown. Bold values indicate which of the accuracies is greater. The p-values indicate the probability of error that must be accepted in order for the difference to be deemed statistically significant. Thus, there is: (1) essentially zero probability of error in accepting the hypothesis that LNPBoost with linear features performs significantly better than EM-DD on TREC9 test set #3; (2) a 3% chance of error in accepting the hypothesis that EM-DD performs significantly better than LNPBoost with linear features on TREC9 test set #9; and (3) a 98% chance of error in accepting the hypothesis that LNPBoost with RBF features performs significantly better than EM-DD on MUSK2. All but two of the differences between EM-DD and LNPBoost with linear features are statistically significant if an error probability of 15% is tolerated. Only one such difference is significant between EM-DD and LNPBoost with RBF features at the same tolerance.

7.7.3 Comparison with EM-DD

LNPBoost was compared with the EM-DD algorithm of [75]. Separate t-tests were performed between EM-DD and LNPBoost with linear features, and between EM-DD and LNPBoost with RBF empirical feature maps. The results are detailed in Table 7.1. The performance of LNPBoost with linear features demonstrates the effectiveness of the sequential convexification technique as a method for hyperplane learning, when used in conjunction with a technique for feature selection. This also suggests that the proposed framework for learning from ambiguous examples is appropriate for dealing with polymorphism.

7.7.4 Comparison with 2-norm mi-SVM and MI-SVM

LNPBoost is also compared with the mi-SVM and MI-SVM mixed-integer algorithms from Chapter 5. Tables 7.2 shows these results. LNPBoost with RBF features obtains the highest accuracy on MUSK2. LNPBoost with linear features has the best accuracy on Elephant, and LNPBoost with RBF features obtains the highest accuracy on Fox. On TREC9, LNPBoost with linear features obtains the highest accuracy in four cases, and is outperformed by a significant margin in only one case. Due to time constraints tests of significance were not performed.

Two hypotheses to explain these differences are as follows. First, the formulations differ with regard to the definition of the margin and the resulting optimization problems solved. In this regard, the superior

Data Set	mi-SVM			MI-SVM			LNPBoost	
Category	linear	poly	rbf	linear	poly	rbf	linear	rbf
MUSK1	76.3	84.0	**87.4**	75.7	72.7	77.9	80.0	84.4
MUSK2	71.5	73.9	83.6	76.0	67.5	84.3	69.0	**85.0**
Elephant	82.2	78.1	80.0	81.4	79.0	73.1	**85.0**	77.5
Fox	58.2	55.2	57.9	57.8	59.4	58.8	57.5	**59.5**
Tiger	78.4	78.1	78.9	**84.0**	81.6	66.6	79.5	75.0
TST1	93.6	92.5	90.4	93.9	93.8	93.7	**94.8**	-
TST2	78.2	75.9	74.3	84.5	84.4	76.4	**86.0**	-
TST3	87.0	83.3	69.0	82.2	85.1	77.4	**90.2**	-
TST4	82.8	80.0	69.6	82.4	**82.9**	77.3	82.5	-
TST7	**81.3**	78.7	**81.3**	78.0	78.7	64.5	78.5	-
TST9	**67.5**	65.6	55.2	60.2	63.7	57.0	60.0	-
TST10	79.6	78.3	52.6	79.5	81.0	69.1	**82.2**	-

Table 7.2: Comparison of LNPBoost and 2-norm mi-SVM and MI-SVM. 10-fold crossvalidation accuracies are reported.

performance of LNPBoost could be due to the fact that the 1-norm objective is better for dealing with polymorphism. Secondly, the strategies that are used to derive approximate solutions to these problems are quite distinct. LNPBoost's apparent improvement in performance could thus be attributed to a superior strategy for solving disjunctive programs. This hypothesis is supported by the asymptotic convergence result for the sequential convexification algorithm.

To test these hypotheses, the 1-norm MI-SVM algorithm is compared with the 2-norm MI-SVM algorithm and with LNPBoost in the next section.

7.7.5 Comparison with 1-norm MI-SVM

The parameter settings and results for the basic 1-norm MI-SVM algorithm without feature selection, are included in Tables 7.7-7.9. Notice that the results for the TREC9 data sets, as well as the MUSK2 data set with RBF empirical feature maps are not given due to an out-of-memory error. The algorithm is also used with feature selection to generate the results in Tables 7.10-7.12.

In the comparisons below, the version of 1-norm MI-SVM with feature selection is used. Justification for this choice follows. First, this version of the algorithm can be applied successfully to all data sets. Second, note that the accuracies obtained by the 1-norm MI-SVM algorithm with feature selection are generally greater than without. This suggests that the feature selection mechanism provides additional regularization that improves classifier performance. A detailed comparison can be made by studying the tables. Finally, a comparison of LNPBoost with the feature selection version of 1-norm MI-SVM is more relevant because both algorithms use the same technique for feature selection.

First, the 1-norm MI-SVM algorithm with feature selection is compared to the 2-norm MI-SVM and mi-SVM algorithms. One difference in the implementations, apart from the objectives that are minimized, is the update strategies; 1-norm MI-SVM uses asynchronous updates while 2-norm MI-SVM uses synchronous updates. Table 7.3 contains accuracies for these algorithms on the benchmark data sets. Notice that for the

Data Set	mi-SVM			MI-SVM			1-norm MI-SVM	
Category	linear	poly	rbf	linear	poly	rbf	linear	rbf
MUSK1	76.3	84.0	**87.4**	75.7	72.7	77.9	77.8	80.0
MUSK2	71.5	73.9	83.6	76.0	67.5	**84.3**	67.0	80.0
Elephant	**82.2**	78.1	80.0	81.4	79.0	73.1	82.0	75.5
Fox	58.2	55.2	57.9	57.8	59.4	58.8	57.5	**59.5**
Tiger	78.4	78.1	78.9	**84.0**	81.6	66.6	78.0	75.5
TST1	93.6	92.5	90.4	93.9	93.8	93.7	**95.0**	-
TST2	78.2	75.9	74.3	84.5	84.4	76.4	**86.0**	-
TST3	87.0	83.3	69.0	82.2	85.1	77.4	**87.8**	-
TST4	82.8	80.0	69.6	82.4	82.9	77.3	**83.1**	-
TST7	**81.3**	78.7	**81.3**	78.0	78.7	64.5	78.2	-
TST9	**67.5**	65.6	55.2	60.2	63.7	57.0	44.0	-
TST10	79.6	78.3	52.6	79.5	81.0	69.1	**82.0**	-

Table 7.3: Comparison of 1-norm MI-SVM and 2-norm MI-SVM. 10-fold crossvalidation accuracies for 1-norm MI-SVM, and 2-norm mi-SVM and MI-SVM, with various kernels. Bold values indicate the highest accuracy for the data set (row).

high-dimensional TREC9 data sets, the 1-norm formulation of learning from ambiguous examples obtains higher accuracies than the 2-norm formulation. The 2-norm formulation is generally better on the MUSK and Corel data sets. Statistical tests of significance in the observed differences are not performed. One conclusion that might be drawn from this comparison is that there is no clear "winner" in terms of the 1-norm and 2-norm formulations.

This suggests that the improved performance of LNPBoost over 2-norm MI-SVM and mi-SVM that was observed in Section 7.7.4 is due to the superiority of the LNPBoost algorithm for approximating the maximum-margin separating hyperplane for ambiguous examples.

The subsequent discussion concerns the feature selection version of 1-norm MI-SVM and the LNPBoost algorithm. In this discussion, the score and accuracy for LNPBoost refers to that of the projected PRDP solution. For these comparisons, the results for the LNPBoost algorithm at the smallest minimum-angle tolerance (respectively the largest maximum-cosine tolerance) are used.

The 1-norm MI-SVM algorithm with feature selection has a score that is generally less than the PRDP score by a small amount. This can be observed in Tables 7.10-7.12 and Tables 7.15-7.17, when comparing similar parameter settings. This suggests that the local greedy search heuristic is a more successful optimization strategy. This would appear to contradict the hypothesis that LNPBoost is superior. As it turns out, however, these lower scores do not necessarily translate into better higher accuracies.

Interestingly enough, the accuracy of the PRDP solution is marginally better than that of 1-norm SVM with feature selection. However, these differences are mostly insignificant for reasonable t-test error probabilities. Table 7.4 contains the relevant accuracies comparisons for LNPBoost and 1-norm MI-SVM. The conclusion is that sequential convexification is marginally better than the local greedy search heuristic. Moreover, sequential convexification comes with a guarantee that it will converge to the solution of DP in the limit, whereas the greedy local search heuristic does not.

	Linear					RBF				
Data Set	1-norm MI-SVM		LNPBoost			1-norm MI-SVM		LNPBoost		
Category	mean	std.	mean	std.	p-value	mean	std.	mean	std.	p-value
MUSK1	77.8	10.5	**80.0**	7.0	0.59	80.0	12.6	**84.4**	9.4	0.39
MUSK2	67.0	13.4	**69.0**	9.9	0.71	80.0	7.1	**85.0**	10.8	0.24
Elephant	82.0	9.2	**85.0**	11.1	0.52	75.5	11.4	**77.5**	13.2	0.72
Fox	**57.5**	10.1	**57.5**	8.6	1.00	**59.5**	8.3	**59.5**	6.4	1.00
Tiger	78.0	8.2	**79.5**	8.3	0.69	**75.5**	5.5	75.0	13.3	0.91
TST1	**95.0**	4.2	94.8	4.2	0.92	-				
TST2	**86.0**	3.6	**86.0**	3.6	1.00	-				
TST3	87.8	5.1	**90.2**	5.5	0.33	-				
TST4	**83.1**	8.8	82.5	8.4	0.88	-				
TST7	78.2	5.0	**78.5**	5.0	0.89	-				
TST9	44.0	5.0	**60.0**	6.7	0.00	-				
TST10	82.0	5.2	**82.3**	6.5	0.94	-				

Table 7.4: Comparison LNPBoost with 1-norm MI-SVM with feature selection. 10-fold crossvalidation accuracies and p-values, using linear and RBF features. Bold values indicate which of the accuracies is greater. The p-values indicate the probability of error that must be accepted in order for the difference to be deemed statistically significant. Thus, there is: (1) essentially zero probability of error in accepting the hypothesis that LNPBoost performs significantly better than 1-norm MI-SVM with linear features on TREC9 test set #9; (2) essentially 100% probability of error in accepting the hypothesis that LNPBoost performs significantly better than 1-norm MI-SVM with RBF features on the Fox data set; and (3) a 24% probability of error in accepting the hypothesis that LNPBoost performs significantly better than 1-norm MI-SVM with RBF features on the MUSK2 data set. The following observation provides a (rather weak) argument in favor of LNPBoost: if one is willing to accept 72% chance of error, then nine out of 17 comparisons are statistically significant and favor LNPBoost, while none of the significant differences favor the 1-norm MI-SVM algorithm.

Data Set Category	SVM		SDP TSVM		LNPBoost	
	mean	std.	mean	std.	mean	std.
USPS digits	0.75	0.04	0.96	0.04	0.93	0.11

Table 7.5: Performance of LNPBoost on a transductive learning task. The values reported are area under the receiver-operator curve scores (AUC). SVM and SDP TSVM results are from [17]. The difference between LNPBoost and SDP TSVM is not significant with a p-value of 0.05.

7.7.6 Optical character recognition

The next experiment concerns the optical character recognition task and corresponding ambiguous data set that was introduced in Section 2.4. In this case, the learning task is transductive inference; some of the examples are labeled (unambiguously), some of the examples are unlabeled, and the goal is to infer the labels of the unlabeled examples.

As described in Section 3.2.1, unlabeled instances are used to create ambiguous examples. After running LNPBoost, the resulting classifier is used to classify *individual instances*, instead of multi-instances. For this example only, the performance is measured on the level of individual instances. The performance metric used was the area under the receiver operator curve (AUC) [28]. Table 7.5 contains the AUC scores as reported in [17], for comparison with LNPBoost. Using the RBF empirical feature mapping, LNPBoost significantly outperforms an inductive SVM with an RBF kernel. This algorithm is denoted by the acronym SVM in the table. The inductive SVM uses only the labeled points. Furthermore, LNPBoost is competitive with the relaxed transductive SVM algorithm from [17]. This algorithm is solved by semi-definite programming, and hence is denoted SDP TSVM in the table.

This result is significant because it demonstrates an important feature that motivated the proposed formulation of learning from ambiguous examples presented herein: the maximum multi-instance margin separating hyperplane corresponds to a low-risk classifier that is able to accurately classify *individual instances*. Moreover, the technique is competitive with the state-of-the-art.

7.8 Discussion

This chapter proposes LNPBoost, a new algorithm developed within a risk minimization framework for learning from ambiguous examples. LNPBoost solves a convexified disjunctive program that is an extension of 1-norm SVM. The formulation and cutting-plane algorithm admits an intuitive geometric understanding. Also, a proof showing the asymptotic convergence of the approximation to the desired solution is provided.

Experiments show that LNPBoost outperforms previous algorithms on many MIL benchmarks, and is also competitive on a transductive learning benchmark. Included in the evaluation are detailed comparisons of LNPBoost with EM-DD and 1-norm MI-SVM. Better performance for all experiments may be possible by tuning parameters on a held-out validation set, in particular the slack penalty C, the cut complexity C_{LNP}, and σ for the RBF features.

One could clearly extend the local search heuristic used by 1-norm MI-SVM, for example by extending the neighborhood of the local search and running the algorithm for a longer time. The value of such a technique would be greatly increased if it could be coupled with a theoretical argument showing that it will convergence to the global minimum.

7.9 Raw results tables

Table 7.6 contains descriptions for the terminology used in subsequent tables. The column headings that are not included are the mean and standard deviation of running time (in minutes), score (i.e. objective value), and multi-instance classification accuracy as measured on the held-out test set.

Name of column heading	Notation	Description
slack_balance	C_{balance}	The fraction of the slack penalty that is used to scale constraint violations of positive (ambiguous) examples.
slack_penalty	$C * m/C_{\text{balance}}$	A value derived from the slack penalty C, the slack balance C_{balance} and the size of the training data set m. The value of C is either 1 or 0.5.
sigma	σ	The width of the radial basis function empirical feature map.
num_round_cglp	N_{LNP}	The number of LNP cuts that are generated before the deepest is selected and added to the sequential convexification.
run_conv_tol	δ_S	The tolerance on the relative change of the objective for detecting convergence of the algorithm.
run_step_tol	δ_D	The depth tolerance relative to $\|\mathbf{w}_t\|$ for adding cuts to $\mathbf{F}_t$.
max_cosine_tol	$\cos(\delta_A)$	The cosine of the minimum-angle tolerance; the angle that must separate each cut. Cuts are accepted when their direction is unique to within this angle tolerance; in other words, when the cosine of the angle between the candidate cut and each of the existing cuts is smaller than $\cos(\delta_A)$. When this parameter is -1, only one cut is accepted. As this parameter is increases to $+1$, more and cuts are accepted.
use_fs		A binary value indicating whether feature selection was used.
fs_max_iter		The maximum number of feature selection iterations.
fs_min_iter		The minimum number of feature selection iterations.
fs_init		The number of randomly chosen features used for initialization.
fs_num_add		The number of features added in a single iteration of feature selection.
fs_add_tol		The tolerance on the feature score above which features are considered for selection.

Table 7.6: Description of parameters in subsequent tables

7.9.1 1-norm MI-SVM without feature selection

parameter	slack_balance	run_conv_tol	run_step_tol	run_cosine_tol	use_fs
value	0.5	0.0001	1e-05	1.1	0

Table 7.7: Common parameters used by 1-norm MI-SVM without feature selection

run_exp_tag	slack_penalty	feature_map	sigma	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std
tiger_890	360	none	-	2.0	1.1	77.0	12.1	60.0	5.7
tiger_815	180	none	-	1.5	0.7	77.5	11.8	39.6	2.9
tiger_891	360	rbf2	8	5.2	3.1	73.5	8.8	114.0	3.0
tiger_860	180	rbf2	8	4.7	2.6	77.0	9.2	67.7	1.4
elephant_890	360	none	-	1.5	0.8	79.5	9.6	70.9	4.0
elephant_815	180	none	-	1.5	0.8	82.5	9.5	42.5	1.8
elephant_891	360	rbf2	8	5.9	3.5	83.0	8.9	124.0	3.0
elephant_860	180	rbf2	8	5.6	3.2	82.5	9.5	71.4	1.0
fox_890	360	none	-	2.0	0.9	54.5	11.4	110.2	4.6
fox_815	180	none	-	2.0	0.9	53.0	9.2	61.8	1.6
fox_891	360	rbf2	8	5.4	3.1	61.5	7.5	150.1	2.7
fox_860	180	rbf2	8	5.0	2.9	60.0	10.8	83.9	0.9

Table 7.8: Corel results for 1-norm MI-SVM without feature selection

run_exp_tag	slack_penalty	feature_map	sigma	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std
musk1_803	150	none	-	0.5	0.2	81.1	10.5	26.2	1.7
musk1_823	75	none	-	0.4	0.1	75.6	12.6	16.2	0.7
musk1_890	150	rbf2	9	0.6	0.2	82.2	11.9	54.3	1.2
musk1_891	75	rbf2	9	0.7	0.2	83.3	10.8	31.1	0.6
musk2_801	150	none	-	247.4	164.4	69.0	11.0	36.5	1.5
musk2_821	75	none	-	212.5	131.4	67.5	11.6	21.0	0.8
musk2_890	150	rbf2	9	-	-	-	-	-	-
musk2_891	75	rbf2	9	-	-	-	-	-	-

Table 7.9: MUSK results for 1-norm MI-SVM without feature selection. Note that the experiments on MUSK2 using the RBF empirical feature map could not be run due to an out of memory error.

7.9.2 1-norm MI-SVM with feature selection

parameter	slack_balance	run_conv_tol	run_step_tol	run_cosine_tol	use_fs	fs_max_iter	fs_min_iter	fs_init	fs_num_add	fs_add_tol
value	0.5	0.0001	1e-05	1.1	1	50	5	5	2	-0.5

Table 7.10: Common parameters used by 1-norm MI-SVM with feature selection on Corel and MUSK data sets

run_exp_tag	slack_penalty	feature_map	sigma	fs_num_iter_avg	fs_num_iter_std	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std
tiger_790	360	none	-	14.6	2.6	11.6	4.1	78.0	8.2	82.0	9.8
tiger_715	180	none	-	17.2	2.9	12.8	6.5	78.0	11.8	44.1	3.3
tiger_791	360	rbf2	8	20.0	5.6	18.5	9.2	73.5	9.7	129.3	19.0
tiger_760	180	rbf2	8	11.6	1.3	12.5	5.3	56.0	17.4	84.0	9.6
elephant_790	360	none	-	21.6	2.5	11.4	3.5	82.0	9.2	81.2	5.9
elephant_715	180	none	-	16.9	3.4	10.5	5.1	84.0	8.1	47.4	2.9
elephant_791	360	rbf2	8	14.1	3.1	12.6	4.6	75.5	11.4	137.1	6.2
elephant_760	180	rbf2	8	11.8	1.4	13.0	5.5	75.0	12.9	75.7	1.9
fox_790	360	none	-	14.9	4.9	16.1	11.1	57.5	10.1	134.0	5.3
fox_715	180	none	-	16.8	3.9	18.1	7.6	57.0	11.4	69.2	2.0
fox_791	360	rbf2	8	11.5	0.8	12.5	5.8	49.5	8.6	174.8	8.8
fox_760	180	rbf2	8	11.0	0.0	12.7	5.6	45.0	4.1	90.0	0.0

Table 7.11: Corel results for 1-norm MI-SVM with feature selection

parameter	slack_balance	run_conv_tol	run_step_tol	run_cosine_tol	use_fs	fs_max_iter	fs_min_iter	fs_init	fs_num_add	fs_add_tol
value	0.5	0.01	1e-4	1.1	1	50	5	5	2	-0.5

Table 7.13: Parameters used by 1-norm MI-SVM with feature selection on TREC9 data sets

run_exp_tag	slack_penalty	feature_map	sigma	fs_num_iter_avg	fs_num_iter_std	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std
musk1_703	150	none	-	14.9	3.6	2.4	1.0	77.8	10.5	31.3	1.7
musk1_723	75	none	-	14.8	4.0	2.4	0.7	76.7	13.3	17.9	1.0
musk1_790	150	rbf2	9	14.2	3.0	2.5	1.2	80.0	12.6	57.2	1.3
musk1_791	75	rbf2	9	12.4	1.6	2.2	0.6	80.0	10.2	31.1	0.6
musk2_701	150	none	-	15.5	3.4	1845.1	754.1	67.0	13.4	45.6	2.0
musk2_721	75	none	-	14.6	3.1	1630.0	1029.2	69.0	8.8	24.4	0.5
musk2_735	150	rbf2	9	10.2	2.9	1528.9	757.6	68.0	19.2	73.2	2.5
musk2_737	75	rbf2	9	-	-	-	-	-	-	-	-

Table 7.12: MUSK results for 1-norm MI-SVM with feature selection

run_exp_tag	slack_penalty	feature_map	sigma	fs_num_iter_avg	fs_num_iter_std	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std
trec_711	720	none	-	10.1	0.3	73.8	35.0	95.0	4.2	267.8	16.3
trec_712	720	none	-	10.0	0.0	84.5	38.0	86.0	3.6	260.9	2.5
trec_713	720	none	-	10.1	0.3	93.3	41.8	87.8	5.1	289.1	8.3
trec_714	720	none	-	10.0	0.0	102.1	42.9	83.1	8.8	273.6	3.2
trec_715	720	none	-	10.0	0.0	95.5	44.4	78.2	5.0	282.5	3.1
trec_716	720	none	-	10.7	0.5	143.2	80.7	44.0	5.0	360.0	0.0
trec_717	720	none	-	10.0	0.0	80.6	35.6	82.0	5.2	290.4	3.6

Table 7.14: TREC9 results for 1-norm MI-SVM with feature selection. Note that, without feature selection, 1-norm MI-SVM reports an out-of-memory error.

7.9.3 LNPBoost

parameter	slack_balance	run_conv_tol	run_step_tol	num_round_cglp	use_fs	fs_max_iter	fs_min_iter	fs_init	fs_num_add	fs_add_tol
value	0.5	0.0001	1e-05	10	1	50	5	5	2	-0.5

Table 7.15: LNPBoost parameters for Corel and MUSK data sets

run_exp_tag	slack_penalty	feature_map	sigma	max_cosine_tol	fs_num_iter_avg	fs_num_iter_std	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std	PRDP_accuracy_avg	PRDP_accuracy_std	PRDP_score_avg	PRDP_score_std
tiger_110	360	none	-	0.60	12.2	1.5	8.2	5.3	70.5	6.9	9.0	1.1	75.5	8.6	86.0	8.0
tiger_111	360	none	-	0.64	11.4	0.5	8.2	3.1	67.5	10.3	10.9	0.8	80.0	9.1	94.0	6.7
tiger_112	360	none	-	0.68	12.3	1.6	20.6	11.4	62.0	11.6	13.9	1.5	76.0	13.5	97.6	5.1
tiger_130	360	none	-	0.70	12.9	2.4	16.7	24.8	64.0	8.1	14.6	2.1	74.0	7.4	92.8	9.7
tiger_131	360	none		0.74	11.6	0.7	16.6	6.3	71.0	11.7	17.5	1.5	79.5	8.3	97.0	8.1
tiger_113	180	none	-	0.60	12.8	2.0	23.9	11.6	66.5	13.1	13.7	1.6	75.0	10.3	47.7	2.2
tiger_114	180	none	-	0.64	12.9	1.4	35.7	9.7	74.5	7.2	14.5	1.1	77.5	10.6	47.3	2.9
tiger_120	360	rbf2	8	0.60	14.0	2.9	7.1	1.4	54.5	6.9	36.8	13.4	73.0	10.6	146.2	12.8
tiger_121	360	rbf2	8	0.64	12.6	1.3	8.2	1.9	57.0	10.9	48.7	5.5	70.0	13.7	149.6	15.1
tiger_122	360	rbf2	8	0.68	12.6	1.5	10.4	1.9	65.5	15.5	52.9	2.6	74.0	10.7	138.9	11.0
tiger_150	360	rbf2	8	0.70	13.5	2.8	13.6	5.6	57.5	11.8	53.0	2.0	74.0	12.4	137.8	11.7
tiger_151	360	rbf2	8	0.74	15.1	2.3	20.0	8.8	65.0	12.9	53.3	3.2	75.0	12.7	139.0	12.5
tiger_152	360	rbf2	8	0.78	13.2	2.0	22.7	10.8	68.5	14.0	54.9	2.9	76.5	9.4	137.7	13.0
tiger_191	360	rbf2	8	0.80	14.6	3.3	54.1	23.4	61.0	9.7	56.3	3.7	75.0	13.3	140.1	7.6
tiger_123	180	rbf2	8	0.60	15.2	4.0	49.4	20.4	68.0	14.4	35.1	3.2	72.0	9.2	77.1	4.6
tiger_124	180	rbf2	8	0.64	18.3	5.3	71.6	32.1	74.5	6.4	34.3	2.0	75.5	8.6	75.4	4.9
tiger_125	180	rbf2	8	0.68	14.9	2.1	56.8	15.4	70.0	10.8	35.6	2.3	76.0	8.8	75.4	5.4
tiger_160	180	rbf2	8	0.70	17.2	4.7	73.1	33.4	77.5	7.9	35.2	1.8	77.0	11.6	75.5	5.2
elephant_110	360	none	-	0.60	12.3	1.3	5.6	1.9	65.5	17.9	8.4	0.7	82.5	12.3	78.3	8.1
elephant_111	360	none	-	0.64	12.2	0.9	9.4	4.3	66.0	15.4	9.4	0.6	82.5	10.6	74.1	8.4
elephant_112	360	none	-	0.68	12.6	1.4	15.6	7.3	67.0	10.1	11.2	1.1	83.0	11.1	74.1	6.1
elephant_130	360	none	-	0.70	13.1	1.7	29.8	26.4	70.5	9.6	11.9	1.1	83.0	9.8	74.0	7.6
elephant_131	360	none	-	0.74	13.1	1.7	47.4	43.3	65.5	11.9	14.0	0.9	85.0	11.1	74.9	4.3
elephant_113	180	none	-	0.60	14.3	2.5	57.8	42.8	73.5	6.7	10.4	1.0	83.5	8.5	38.8	2.3
elephant_120	360	rbf2	8	0.60	12.9	2.1	6.4	1.8	52.5	13.6	39.7	7.9	71.5	15.6	157.5	8.5
elephant_121	360	rbf2	8	0.64	12.5	1.5	7.8	2.1	61.5	14.0	49.7	6.6	72.5	13.6	150.3	9.7
elephant_122	360	rbf2	8	0.68	12.3	1.6	10.2	2.1	62.0	10.3	52.8	3.7	75.5	11.7	150.2	7.0
elephant_150	360	rbf2	8	0.70	12.7	2.2	12.7	4.2	63.0	11.6	55.2	1.8	73.5	11.3	147.8	9.1
elephant_151	360	rbf2	8	0.74	13.7	2.8	17.4	4.3	56.5	11.6	55.3	2.8	77.5	13.2	149.5	5.8
elephant_123	180	rbf2	8	0.60	17.0	4.6	66.0	55.6	66.0	12.9	33.0	1.7	72.0	11.8	78.2	2.9
elephant_124	180	rbf2	8	0.64	14.9	4.3	53.0	21.8	71.5	14.0	33.8	1.2	74.5	13.4	76.9	2.7
elephant_125	180	rbf2	8	0.68	14.5	3.0	80.6	91.6	67.0	15.3	34.1	1.8	73.0	13.8	77.4	1.8
elephant_160	180	rbf2	8	0.70	14.9	3.0	63.9	17.7	68.0	11.4	33.9	1.2	71.0	12.9	77.3	2.9
fox_110	360	none	-	0.60	12.4	1.7	7.0	2.7	53.5	15.5	10.1	1.1	53.5	9.7	136.6	7.2
fox_111	360	none	-	0.64	12.9	1.7	10.7	5.6	55.5	14.0	12.1	1.2	59.0	13.5	140.8	3.8
fox_112	360	none	-	0.68	12.7	1.1	15.4	4.0	50.0	7.5	15.0	1.4	54.0	9.1	143.3	4.8
fox_130	360	none	-	0.70	12.1	1.5	20.0	12.9	54.0	9.1	17.7	1.5	56.0	9.9	141.7	5.1
fox_131	360	none	-	0.74	13.0	1.5	44.1	38.3	54.0	10.5	19.9	2.2	57.5	8.6	139.5	3.8
fox_113	180	none	-	0.60	14.4	3.0	46.1	29.0	54.5	9.6	15.2	1.8	56.0	7.0	72.6	2.0
fox_120	360	rbf2	8	0.60	12.6	1.1	4.9	0.8	56.0	10.5	44.5	3.6	54.0	10.5	169.7	3.4
fox_121	360	rbf2	8	0.64	14.1	2.3	10.6	3.3	50.5	8.3	46.6	6.0	53.0	7.5	168.1	5.0
fox_122	360	rbf2	8	0.68	12.8	2.3	11.1	4.3	50.5	13.4	51.2	3.8	60.5	13.8	168.9	4.0
fox_150	360	rbf2	8	0.70	13.2	2.2	14.1	5.5	54.5	9.6	53.5	3.5	55.0	11.3	168.8	4.0
fox_151	360	rbf2	8	0.74	13.7	2.8	18.8	6.0	51.0	8.1	55.7	2.6	59.5	6.4	169.9	4.2
fox_123	180	rbf2	8	0.60	16.7	2.8	61.7	17.8	52.5	14.0	33.6	2.5	54.0	14.3	87.4	6.0
fox_124	180	rbf2	8	0.64	16.5	5.2	64.7	27.9	57.0	12.1	34.1	2.8	57.5	7.9	86.3	1.5
fox_125	180	rbf2	8	0.68	16.8	3.6	81.3	27.4	55.0	13.9	34.0	2.4	55.0	10.3	88.9	6.9
fox_160	180	rbf2	8	0.70	17.2	2.9	86.8	21.5	55.0	13.7	33.9	1.4	53.0	13.4	86.5	1.6

Table 7.16: Corel results for LNPBoost

run_exp_tag	slack_penalty	slack_balance	feature_map	sigma	max_cosine_tol	fs_num_iter_avg	fs_num_iter_std	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std	PRDP_accuracy_avg	PRDP_accuracy_std	PRDP_score_avg	PRDP_score_std
musk1_500	150	0.5	none	-	0.70	17.5	6.1	12.9	9.7	54.4	19.2	5.7	1.0	78.9	8.2	31.8	2.0
musk1_501	150	0.5	none	-	0.74	14.7	3.3	15.0	6.4	68.9	18.0	7.4	1.4	78.9	8.2	31.6	2.4
musk1_502	150	0.5	none	-	0.78	18.7	5.0	39.5	23.1	60.0	18.3	7.1	1.3	72.2	9.4	31.8	3.0
musk1_503	150	0.5	none	-	0.82	18.1	3.8	120.8	35.7	56.7	17.7	8.6	1.7	80.0	7.0	31.8	2.6
musk1_520	75	0.5	none	-	0.70	15.4	3.6	18.5	7.9	54.4	20.6	6.2	1.1	75.6	11.5	17.9	0.8
musk1_521	75	0.5	none	-	0.74	21.6	7.8	49.1	30.5	68.9	12.6	5.8	1.3	74.4	11.8	18.0	0.7
musk1_522	75	0.5	none	-	0.78	17.9	4.9	48.1	24.0	72.2	13.1	6.9	1.1	81.1	5.4	18.6	1.2
musk1_523	75	0.5	none	-	0.82	21.9	4.5	279.6	141.1	65.6	11.0	6.7	0.8	81.1	11.8	18.5	0.8
musk1_600	150	0.5	rbf2	9	0.70	13.3	2.5	6.5	2.2	60.0	15.0	39.6	2.5	83.3	9.4	58.8	2.1
musk1_601	150	0.5	rbf2	9	0.74	14.9	3.4	8.5	3.3	66.7	19.6	39.1	2.5	80.0	7.0	58.9	2.6
musk1_602	150	0.5	rbf2	9	0.78	14.5	2.5	12.1	3.6	71.1	14.1	40.7	2.0	84.4	13.0	58.4	1.4
musk1_603	150	0.5	rbf2	9	0.82	14.9	2.1	15.8	7.1	68.9	11.5	40.4	1.5	78.9	11.0	58.5	1.7
musk1_604	150	0.5	rbf2	9	0.86	12.8	2.4	13.4	8.0	68.9	11.5	41.3	2.1	84.4	9.4	58.3	1.8
musk1_605	150	0.5	rbf2	9	0.90	13.7	2.7	23.4	10.5	65.6	20.6	42.6	2.2	83.3	9.4	58.6	2.2
musk1_620	75	0.5	rbf2	9	0.70	14.2	2.0	8.3	2.1	73.3	15.0	24.4	0.4	77.8	15.7	31.4	0.6
musk1_621	75	0.5	rbf2	9	0.74	14.8	2.4	8.3	2.8	83.3	15.9	24.4	0.5	80.0	13.7	31.4	0.6
musk1_622	75	0.5	rbf2	9	0.78	13.7	2.2	7.5	2.2	76.7	15.2	24.4	0.5	76.7	12.2	31.6	0.5
musk1_623	75	0.5	rbf2	9	0.82	13.7	1.6	9.1	3.4	78.9	13.3	24.4	0.6	80.0	11.5	31.3	0.6
musk1_624	75	0.5	rbf2	9	0.86	13.5	2.6	11.2	7.0	77.8	12.8	24.5	0.7	82.2	14.1	31.3	0.6
musk1_625	75	0.5	rbf2	9	0.90	12.4	2.1	10.3	3.9	72.2	15.9	24.9	0.7	77.8	14.8	31.4	0.6
musk1_626	75	0.5	rbf2	9	0.96	12.9	2.6	61.5	58.5	83.3	9.4	28.4	0.8	81.1	9.1	31.1	0.7
musk2_530	150	0.5	none	-	0.60	14.9	5.1	25.4	18.9	60.0	11.5	5.9	1.2	69.0	9.9	42.3	3.3
musk2_521	75	0.5	none	-	0.60	22.6	7.8	454.4	307.6	62.0	9.2	5.0	1.2	71.0	12.0	22.9	1.1
musk2_104	150	0.5	rbf2	9	0.60	14.7	2.8	35.1	10.3	64.0	13.5	37.7	2.3	84.0	11.7	68.2	2.0
musk2_102	150	0.5	rbf2	9	0.68	16.0	3.7	24.8	9.4	65.0	13.5	38.8	1.5	80.0	12.5	68.0	1.3
musk2_100	150	0.5	rbf2	9	0.74	15.1	4.4	23.7	11.7	67.0	16.4	40.4	1.7	84.0	10.7	69.2	1.0
musk2_105	150	0.5	rbf2	9	0.78	14.1	3.8	55.2	35.2	67.0	8.2	42.3	2.2	83.0	10.6	68.0	1.3
musk2_106	150	0.5	rbf2	9	0.82	13.7	2.3	27.6	8.2	66.0	12.6	42.9	2.5	85.0	10.8	68.9	1.0
musk2_531	75	0.5	rbf2	9	0.60	13.3	1.9	15.6	5.2	67.0	12.5	25.9	1.0	82.0	10.3	36.0	0.5
musk2_532	75	0.5	rbf2	9	0.68	15.2	4.6	22.5	11.9	73.0	24.1	25.9	1.0	83.0	12.5	35.9	0.6
musk2_121	75	0.5	rbf2	9	0.74	13.6	2.2	41.9	22.3	76.0	15.1	26.3	0.8	81.0	13.7	35.9	0.4
musk2_122	75	0.5	rbf2	9	0.78	14.8	3.2	22.4	10.3	79.0	8.8	26.2	0.8	80.0	9.4	35.9	0.5
musk2_534	75	0.5	rbf2	9	0.86	12.7	1.9	37.3	19.2	73.0	12.5	27.3	1.2	80.0	16.3	36.0	0.5
musk2_536	75	0.5	rbf2	9	0.90	13.7	3.6	42.8	24.4	77.0	9.5	27.2	1.4	80.0	16.3	35.9	0.5

Table 7.17: MUSK results for LNPBoost

parameter	slack_balance	run_conv_tol	run_step_tol	num_round_cglp	use_fs	fs_max_iter	fs_min_iter	fs_init	fs_num_add	fs_add_tol
value	0.5	0.01	1e-4	1	1	50	5	5	2	-0.5

Table 7.18: LNPBoost parameters for TREC9 data sets

run_exp_tag	slack_penalty	feature_map	sigma	max_cosine_tol	fs_num_iter_avg	fs_num_iter_std	time_avg	time_std	accuracy_avg	accuracy_std	score_avg	score_std	PRDP_accuracy_avg	PRDP_accuracy_std	PRDP_score_avg	PRDP_score_std
trec_11	720	none	-	0.90	11.0	0.0	143.4	36.5	94.2	4.1	225.2	4.5	94.8	4.2	242.5	3.4
trec_12	720	none	-	0.90	11.1	0.3	74.2	3.5	86.0	3.6	236.7	4.6	86.0	3.6	260.8	2.4
trec_13	720	none	-	0.90	11.1	0.3	79.7	5.0	89.0	6.5	246.4	4.5	90.2	5.5	282.4	3.2
trec_14	720	none	-	0.90	11.8	1.4	84.7	13.6	82.5	8.7	239.8	3.6	82.5	8.4	269.4	3.0
trec_15	720	none	-	0.90	11.0	0.0	73.9	2.1	78.2	4.7	242.6	5.0	78.5	5.0	281.3	3.1
trec_16	720	none	-	0.90	14.0	2.7	169.2	52.8	58.0	5.9	294.8	6.2	60.0	6.7	336.6	7.3
trec_17	720	none	-	0.90	11.5	1.1	80.7	10.3	83.2	5.5	249.3	4.6	82.2	6.5	285.9	3.7

Table 7.19: TREC9 results for LNPBoost

Chapter 8

Summary and conclusions

This thesis is concerned with learning from examples that are ambiguous, as observed in several semi-supervised learning problems. Specifically the problems are multiple-instance learning, transductive inference and semi-supervised inductive inference. Multiple-instance learning addresses the problem of learning from inputs that are ambiguous due to polymorphism. Transductive learning and semi-supervised inductive inference deal with the problem of learning from a small collection of examples (instance-label pairs) that is augmented by a large collection of unlabeled inputs. Restricting to hyperplanes that pass through the origin, transductive learning and semi-supervised inductive inference can both be reduced to multi-instance learning. The reduction follows by converting the unlabeled and ambiguous inputs from these problems, into polymorphic inputs (multi-instances) with specific labels. This reduction to multi-instance learning aligns all sources of ambiguity under one framework. The key question that is addressed in this thesis is the following. Is it possible to recover the implicit categories underlying a collection of ambiguous examples?

This thesis proposes a framework for learning from ambiguous examples. Learning from ambiguous examples is performed through the systematic treatment of ambiguity. Information from ambiguous examples is derived via *disambiguation*, where inferences involving the interpretations of ambiguous examples are made. The key intuition is that *the true interpretation is self-reinforcing*. To formalize this idea, this thesis proposes a risk minimization principle for learning from ambiguous examples. As such this thesis presents a principled framework for making specific and plausible inferences about ambiguous inputs in a pattern classification system.

Specifically, disambiguation is encoded implicitly by selecting the "most positive" instance in each multi-instance. In this way, the principle of structural risk minimization drives the selection of the disambiguation to obtain the best bound on the generalization error of the underlying classifier. Practically speaking, this formulation leads to a non-linear optimization problem. Subsequently, a significant portion of this thesis concerns the derivation of specialized optimization techniques for finding the maximum-margin hyperplane, or a close approximation to it.

To summarize, the contributions of this thesis are:

1. Two data sets

2. A structural risk minimization / maximum-margin framework for learning from ambiguous examples

3. A pictorial framework for conveying principles of risk minimization and convex programming

4. One algorithm for learning from ambiguous examples using a 2-norm formulation of the maximum-margin problem

   (a) *Multi-instance support vector machine* (MI-SVM)

5. Three algorithms for learning from ambiguous examples using a 1-norm formulation of the maximum-margin problem

   (a) *Disjunctive programming boosting* (DPBoost)

   (b) *1-norm multi-instance support vector machine* (1-norm MI-SVM)

   (c) *Lift and project boosting* (LNPBoost)

6. Empirical evaluations on multi-instance learning and transductive learning domains

The first algorithm, multi-instance support vector machines (MI-SVM), uses the 2-norm to bound the weight vector. This results in an optimization with a quadratic objective function and non-linear constraints. Integer variables are introduced to represent the selection of an interpretation. This results in a challenging mixed-integer quadratic optimization problem, and an efficient greedy local search algorithm is derived. Despite the lack of guaranteed global convergence, the algorithm significantly outperforms a baseline method on several benchmarks. Because the approach is general, it works equally well for document categorization and automatic image annotation. The main short-coming of this technique is that, due to its susceptibility to local minima and despite its impressive empirical performance, it is uncertain whether the approximate solution will generalize as expected by the SRM principle for ambiguous examples.

The second algorithm, disjunctive programming boosting (DPBoost), uses the 1-norm to bound the weight vector. The resulting optimization problem has a linear objective function and non-linear constraints. The problem is approximated by replacing the non-linear feasible region with a larger convex set that is derived using lifting techniques, specifically hull-relaxations and parallel reductions, from disjunctive programming. This yields a large linear program; the number of rows and columns is $\mathcal{O}\left(D^2\right)$ where D the maximum of the number of features and the number of examples in the training data set. The benefit of this approach is convexity, implying that a unique globally optimal solution can be found for the approximation. In this way, local minima are avoided at the expense of solving an approximation to the original problem. From the perspective of combinatorial optimization, the solution is useful in that it provides a lower bound that could be used in a branch-and-bound scheme to find the global minimum of the original problem. Unfortunately, the algorithm had limited applicability due to its excessive memory requirements and its complexity.

The third algorithm called lift-and-project boosting (LNPBoost), also uses the 1-norm bound the weight vector. This approach was motivated by the need to limit the size and management difficulties of the linear program in DPBoost. In order to address this problem, a compact sequential approximation of the convex hull is computed by way of a cutting-plane technique adapted from combinatorial optimization. A key component of this technique is a novel formulation of lift-and-project cutting-planes for the disjunctive sets that arise from multi-instance margin constraints; in other words, a formulation of lift-and-project cutting-planes for arbitrary unions of halfspaces. A proof of the asymptotic convergence of the proposed sequential convexification algorithm is provided. The technique also manages the set of features used in the classifier with a boosting-like feature selection algorithm so that it can be applied to large, high-dimensional data sets.

LNPBoost computes a lower bound on the maximum-margin hyperplane problem for ambiguous examples. Analysis shows that the lower bound increases as the quality of the approximation increases. A projection from this solution onto the feasible region provides an upper bound. Analysis also shows an increasing trend in performance of the projected solution, as the quality of the compact sequential convexification increases. This trend is reflected by the accuracy of the corresponding classifier as the minimum-angle tolerance is decreased. Furthermore, evidence is provided to show that the algorithm is able to select amongst feasible points whose objective values are indistinguishable, by increasing the quality of the sequential approximation. This shows that the theoretical asymptotic convergence of the algorithm is realized in practice.

Results suggest that LNPBoost outperforms the 2-norm MI-SVM algorithm, although tests of significance were not included. LNPBoost outperforms EM-DD. For this result, statistical significance is shown.

A 1-norm version of the MI-SVM algorithm is introduced to complete the analysis of the proposed framework for learning from ambiguous examples. As an optimization strategy, 1-norm MI-SVM uses the 2-stage greedy local search heuristic like the 2-norm version. The 1-norm version is similarly fast and susceptible to local minima. Results suggest that, on the one hand the 1-norm version of MI-SVM performs better than the 2-norm version on the high-dimensional TREC9 data sets, but on the other hand it is worse on the Corel and MUSK data sets. The conclusion from this result is that neither of the two formulations (1-norm / 2-norm) is clearly superior.

The 1-norm version of the MI-SVM algorithm provides a suitable benchmark for comparison with the LNPBoost algorithm, because it optimizes the same objective function and uses the same feature selection technique. This comparison reveals that the classifier returned by the LNPBoost algorithm is slightly more accurate than that obtained by the 1-norm MI-SVM algorithm with feature selection. Therefore, the conclusion is the LNPBoost provides the best performance out of algorithms studied.

Finally, while the experiments on the Corel, MUSK and TREC9 data sets do not explicitly verify whether or not the true disambiguation is found, the transductive inference experiment on the USPS data set shows that this is indeed the case.

Therefore, this thesis concludes that the burden of collecting a large number of instance-label pairs may be supplanted with algorithms that learn from readily available ambiguous examples.

Bibliography

[1] H. Ben Amor, J. Desrosiers, and A. Frangioni. Stabilization in column generation. Technical report, GERAD, 2004.

[2] S. Andrews and T. Hofmann. Disjunctive programming boosting. In *Advances in Neural Information Processing Systems*, volume 16, 2004.

[3] S. Andrews and T. Hofmann. A cutting-plane algorithm for learning from ambiguous examples. Technical report, Brown U., 2006.

[4] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

[5] P. Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. *Machine Learning*, pages 21–29, 1997.

[6] E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

[7] E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *Society for Industrial and Applied Mathematics*, 6(3):466–486, July 1985.

[8] E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89:1–44, 1998.

[9] E. Balas. Projection and lifting in combinatorial optimization. *Computational Combinatorial Optimization*, 2001.

[10] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[11] E. Balas and M. Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Mathematical Programming*, 2003.

[12] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. Blei, and M. I. Jordan. Matching words and pictures. *JMLR*, 2003.

[13] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 1998.

[14] K. Bennett and A. Demiriz. Semisupervised support vector machines. In *Advances in Neural Information Processing Systems (NIPS*12)*, 1999.

[15] K. P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In *ICML*, 2000.

[16] J. Bi, J. Z. Wang, and Y. Chen. A sparse support vector machine approach to region-based image categorization. In *Computer Vision and Pattern Recognition*, 2005.

[17] T. De Bie and N. Cristianini. Convex methods for transduction. In *Advances in Neural Information Processing Systems*, 2003.

[18] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice - closing the gap. Technical report, ILOG, xxxx.

[19] A. Blum and A. Kalai. A note on learning from multiple-instance examples. *Machine Learning*, 1998.

[20] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proceedings Third International Conference on Visual Information Systems*. Springer, 1999.

[21] Y. Chen and J. Z. Wang. Image categorization by learning and reasoning with regions. *Machine Learning Research*, 5:913–939, 2004.

[22] Y. Chevaleyre, N. Bredeche, and J. D. Zucker. Learning rules from multiple instance data : Issues and algorithms. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2002.

[23] Y. Chevaleyre and J. D. Zucker. Noise-tolerant rule induction from multi-instance data. In *ICML*, 2000.

[24] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, 2000.

[25] A. Demiriz and K. Bennett. Optimization approaches to semisupervised learning. In M. Ferris, O. Mangasarian, and J. Pang, editors, *Applications and Algorithms of Complementarity*. Kluwer Academic Publishers, Boston, 2000.

[26] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

[27] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[28] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, New York, 2001.

[29] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. Conference on Computer Vision and Pattern Recognition*, 2003.

[30] E. Frank and X. Xu. Applying propositional learning algorithms to multi-instance data. Technical report, University of Waikato, 2003.

[31] E. Frank and X. Xu. Logistic regression and boosting for labeled bags of instances. In *Pacific Asia Knowledge Discovery and Data Mining*, 2004.

[32] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. Multi-instance kernels. In *Proc. 19th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2002.

[33] T. Gartner, J. W. Lloyd, and P. Flach. Kernels and distances for structured data. *Machine Learning*, 2004.

[34] S. A. Goldman and S. D. Scott. Multiple-instance learning of real-valued geometric patterns. *Mathematics and Artificial Intelligence*, 2003.

[35] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[36] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: Global versus component-based approach, 2001.

[37] ILOG. *ILOG CPLEX 6.5 Reference Manual*. ILOGCPLEX Division, Incline Village, NV, 1999.

[38] T. Joachims. *Advances in Kernel Methods - Support Vector Learning*, chapter Making Large-Scale SVM Learning Practical, pages 41–56. MIT Press, 1999.

[39] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann, San Francisco, CA, 1999.

[40] J. D. Keeler, D. E. Rumelhart, and W.-K. Leow. Integrated segmentation and recognition of hand-printed numerals. In *Neural Information Processing Systems 3*, 1990.

[41] J. E. Kelley. The cutting-plane method for solving convex programs. *Industrial and Applied Mathematics*, 1960.

[42] H. Kück, P. Carbonetto, and N. de Freitas. A constrained semi-supervised learning approach to data association. In *European Conference on Computer Vision*. Springer, 2004.

[43] S. Lee and I. E. Grossmann. New algorithms for nonlinear generalized disjunctive programming. *Computers and Chemical Engineering Journal*, 24(9-10):2125–2141, October 2000.

[44] Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 2002.

[45] P.M. Long and L. Tan. PAC learning axis aligned rectangles with respect to product distributions from multiple-instance examples. In *Proc. Comp. Learning Theory*, 1996.

[46] O. Mangasarian. Arbitrary-norm separating hyperplane. Technical report, University of San Diego, 1997.

[47] O. Maron. *Learning from Ambiguity*. PhD thesis, Massachusetts Institute of Technology, 1998.

[48] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

[49] O. Maron and A. L. Ratan. Multiple-instance learning for natural scene classification. In *Proc. 15th International Conf. on Machine Learning*, pages 341–349. Morgan Kaufmann, San Francisco, CA, 1998.

[50] A. McGovern and D. Jensen. Identifying predictive structures in relational data using multiple instance learning. In *ICML*, 2003.

[51] K. P. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *Advances in Neural Information Processing Systems*, volume 16, 2004.

[52] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, 1988.

[53] J. H. Owen and S. Mehrotra. A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming*, (89):437–448, 2001.

[54] D. Page and M. Craven. Biological applications of multi-relational data mining. In *Knowledge Discovery and Data Mining*, 2003.

[55] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, Englewood Cliffs, N.J., 1982.

[56] C. Perlich and F. Provost. Aggregation-based feature invention and relational concept classes. In *Knowledge Discovery and Data Mining*, 2003.

[57] L. De Raedt. Attribute-value learning versus inductive logic programming (the missing links). In *Inductive Logic Programming*, 1998.

[58] J. Ramon and L. De Raedt. Multi instance neural networks. In *Proceedings of ICML-2000, Workshop on Attribute-Value and Relational Learning*, 2000.

[59] G. Rätsch, S. Mika, B. Scholköpf, and K. R. Müller. Constructing boosting algorithms from SVMs: an application to one-class classification. 24(9-10):1184–1199, 2002.

[60] S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *International Conference on Machine Learning*, 2005.

[61] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[62] G. Ruffo. *Learning single and multiple decision trees for security applications*. PhD thesis, University of Turin, Italy, 2000.

[63] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[64] S. Scott, J. Zhang, and J. Brown. On generalized multiple-instance learning. Technical report, University of Nebraska, 2003.

[65] A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans. *Advances in Large Margin Classifiers*. MIT Press, 1999.

[66] Q. Tao, S. Scott, N. V. Vinodchandran Thomas, and T. Osugi. Svm-based generalized multiple-instance learning via approximate box counting. In *International Conference on Machine Learning*, 2004.

[67] Q. Tao and S. D. Scott. A faster algorithm for generalized multiple-instance learning. In *Artificial Intelligence*, 2004.

[68] V.N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[69] P. Viola. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[70] G. Wahba, C. Gu, Y. Wang, and R. Chappell. Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance. In D.H. Wolpert, editor, *The Mathematics of Generalization*. Addison-Wesley, 1995.

[71] J. Wang and J. D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *ICML*, 2000.

[72] N. Weidmann, E. Frank, and B. Pfahringer. A two-level learning method for generalized multi-instance problems. In *European Conference on Machine Learning*, 2003.

[73] J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider, 2004.

[74] Q. Zhang, S. A. Goldman, W. Yu, and J. E. Fritts. Content-based image retrieval using multiple-instance learning. In *ICML*, 2002.

[75] Qi Zhang and Sally A. Goldman. EM-DD: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[76] Z. H. Zhou and M. L. Zhang. Neural networks for multi-instance learning. In *Proceedings of the International Conference on Intelligent Information Technology*, 2002.

[77] Z. H. Zhou and M. L. Zhang. Ensembles of multi-instance learners. In *European Conf on Machine Learning*, 2003.

[78] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Neural Information Processing Systems*, 2004.

[79] J. D. Zucker and Y. Chevaleyre. Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. application to the mutagenesis problem. In *Canadian Conference on Artificial Intelligence*, 2001.