Statistical Methods for NLP

Text Categorization, Linear Methods of Classification

Sameer Maskey

Week 2, Sept 12, 2012

Announcement

- Reading Assignments
  - Bishop Book – 5.1, 5.2, 5.3
  - J&M Book – 3.1, 3.8, 4.5.1
- Final Project
  - **Project Proposal Due: Sept 19th, 11:59pm, Wednesday**
  - Fill in the survey if you want us to match you with a student
- Survey
  - End of class, please fill in the survey
- Signup sheet for project partner

Project

- You should start thinking about the kind of project you want to do
  - This is a semester long project
  - Goal is for you to understand one NLP-ML problem of your interest in-depth
  - Depending on your expertise and passion you may try to get a research paper out of it
- Some Examples
  - Review to Ratings: Automatic Reviewer
  - Build a search engine writing your own ML algorithm
  - Information Extraction for Relations
  - Statistical Parser with your statistical algorithm for decoding

Student Projects from Last Class

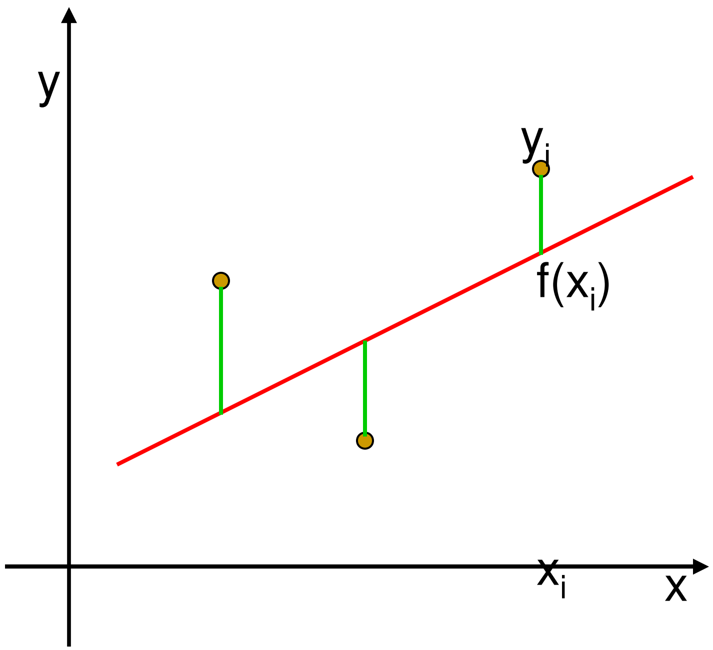
- Section Classification in Clinical Notes using Supervised HMM - Ying
- Automatic Summarization of Recipe Reviews - Benjamin
- Classifying Kid-submitted Comments using Machine Learning Techniques - Tony
- Towards An Effective Feature Selection Framework - Boyi
- Using Output Codes as a Boosting Mechanism - Green
- Enriching CATiB Treebank with Morphological Features - Sarah
- SuperWSD: SupervisedWord Sense Disambiguation by Cross-Lingual Lexical Substitution - Wenhan
- L1 regularization in log-linear Models - Tony
- A System for Routing Papers to Proper Reviewers - Zhihai

Topics for Today

- Text Categorization/Classification
- Bag of Words Model
- Linear Discriminant Functions
- Perceptron
- Naïve Bayes Classifier for Text

Review: Linear Regression

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\theta_0 + \theta_1 x_i))^2$$



$$\theta_1 = \frac{\sum_{i=1}^{N} x_i y_i - \frac{1}{N}\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{\sum_{i=1}^{N} x_i^2 - \frac{1}{N}\sum_{i=1}^{N} x_i \sum_{i=1}^{N} x_i}$$

$$\theta_0 = \frac{1}{N}\sum_{i=1}^{N} y_i - \frac{1}{N}\theta_1 \sum_{i=1}^{N} x_i$$

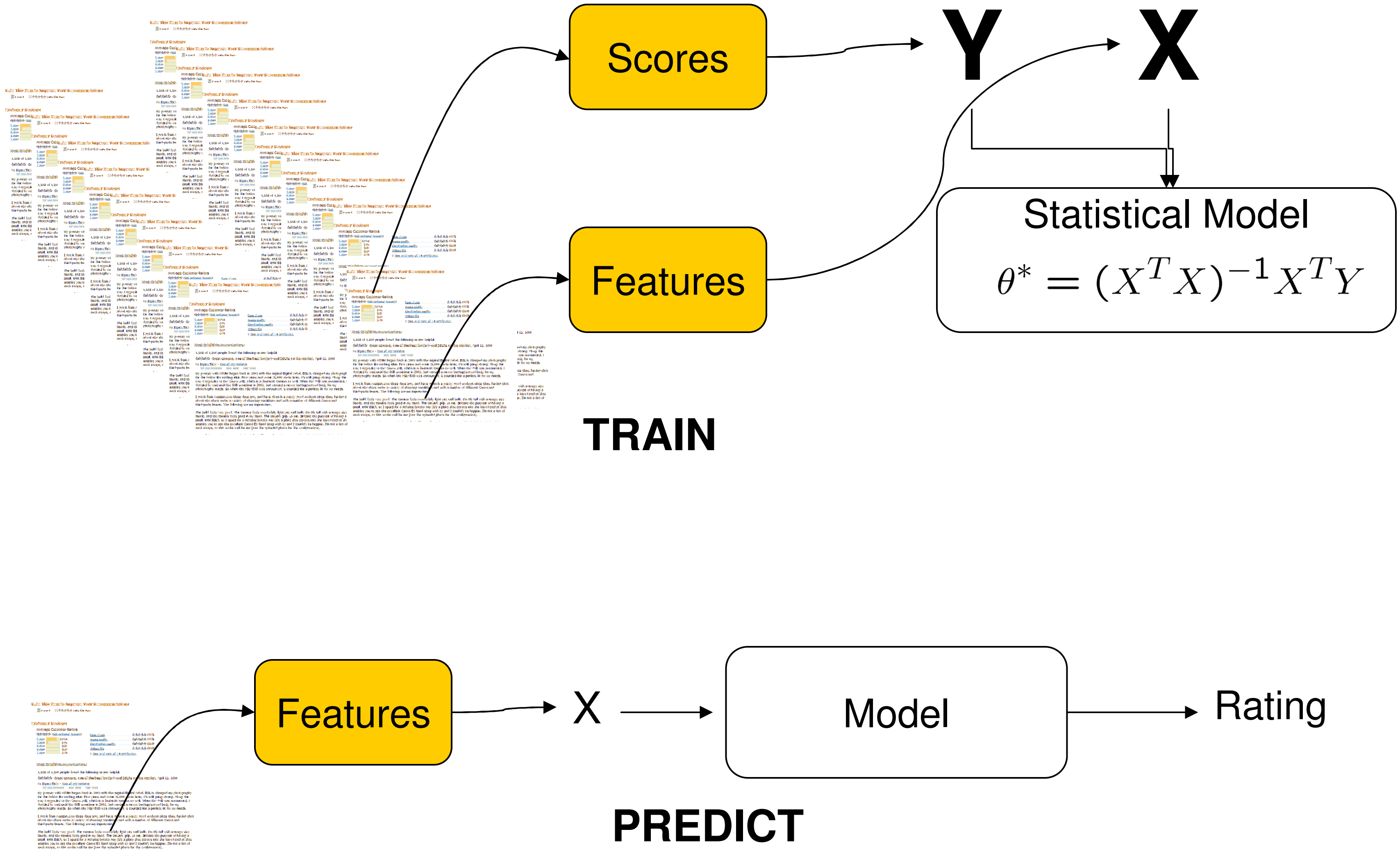
Review: Multiple Linear Regression

$$J(\theta) = \frac{1}{2N}\left\| \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ Y_N \end{bmatrix} - \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1K} \\ 1 & x_{21} & x_{22} & \ldots & x_{2K} \\ & & \ldots & & \\ & & \ldots & & \\ & & \ldots & & \\ 1 & x_{N1} & x_{N2} & \ldots & x_{NK} \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_K \end{bmatrix} \right\|^2$$

↑ **Y** ↑ **X** ↑ **θ**

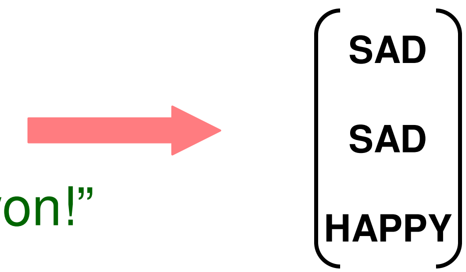
$$\theta^* = (X^T X)^{-1} X^T Y$$

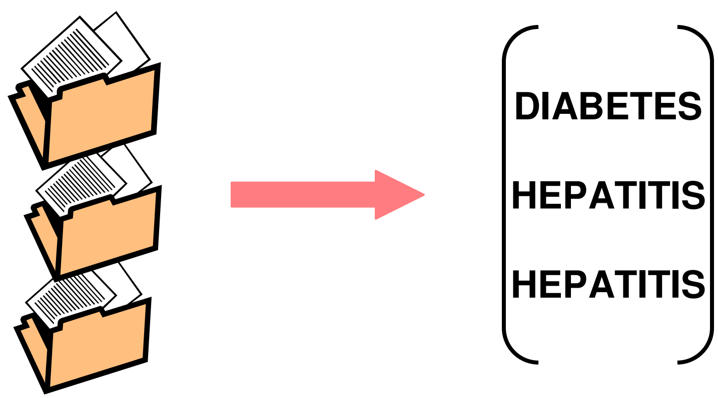
Reviews to Automatic Ratings



Predicted Output May Be Classes

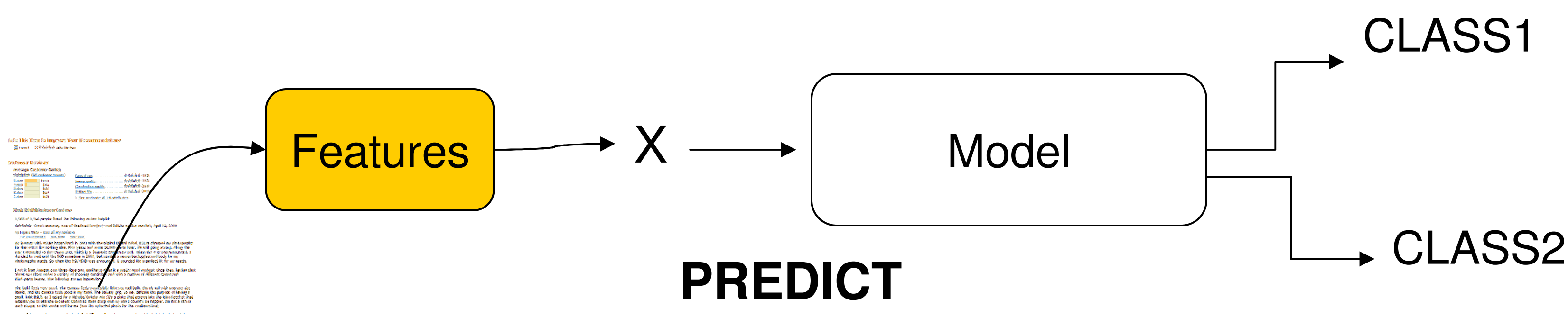
- "…is writing a paper"
- "… has flu ☹"
- "… is happy, yankees won!"





Text Categorization/Classification

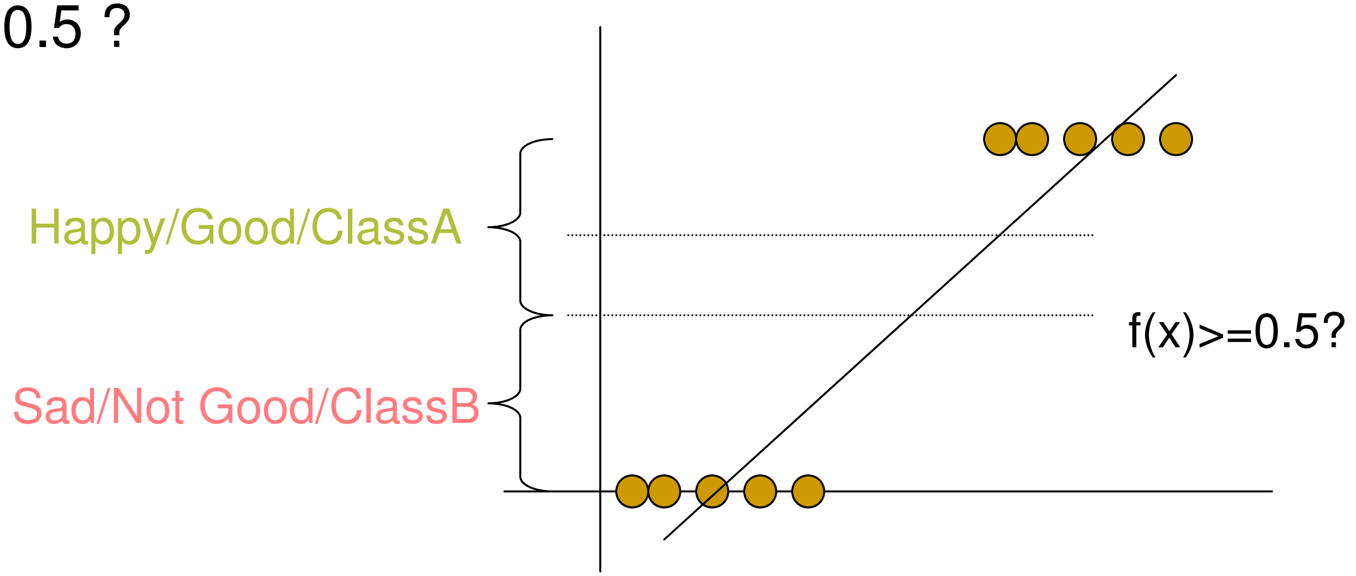
- Given any text (sentence, document, stories, etc), we want to classify it into some predefined class set



- Training Data consists of Y values that are 0 and 1
  - Review is good or bad
  - Status update is happy or sad

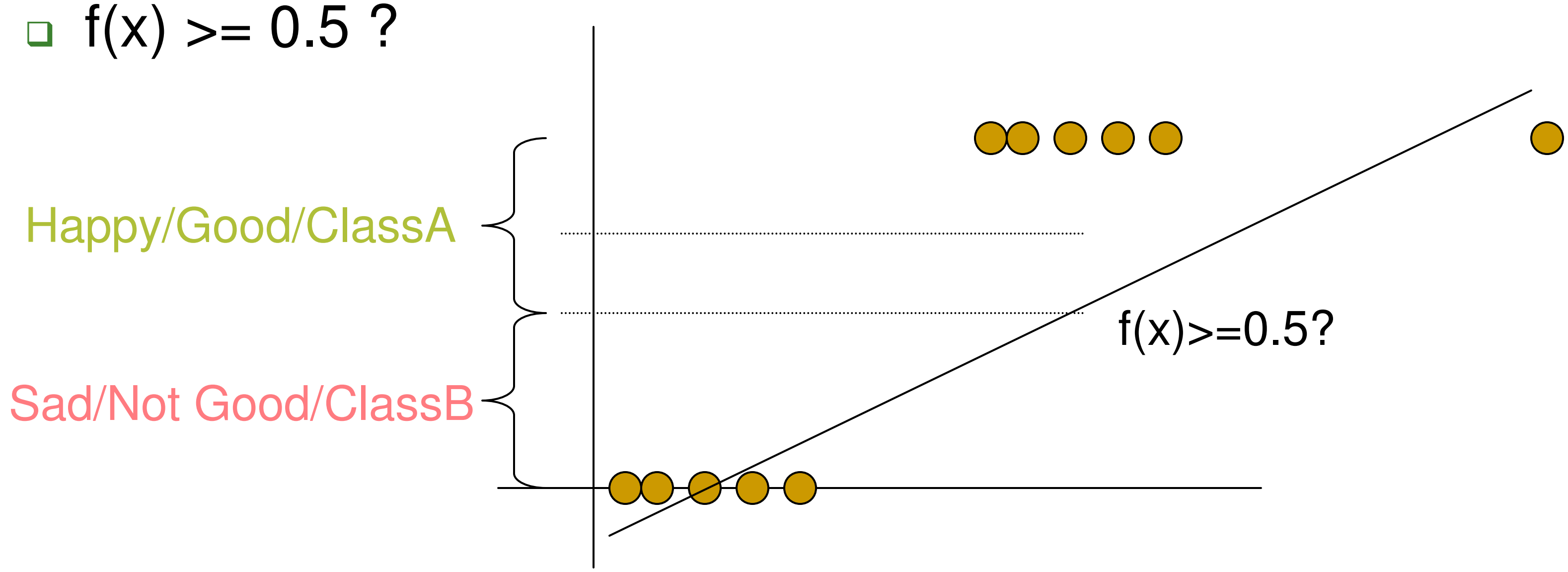
Regression to Classification

- Can we build a regression model to model such binary classes?
- Train Regression and threshold the output
  - If f(x) >= 0.7 CLASS1
  - If f(x) < 0.7 CLASS2
  - f(x) >= 0.5 ?



Regression to Classification

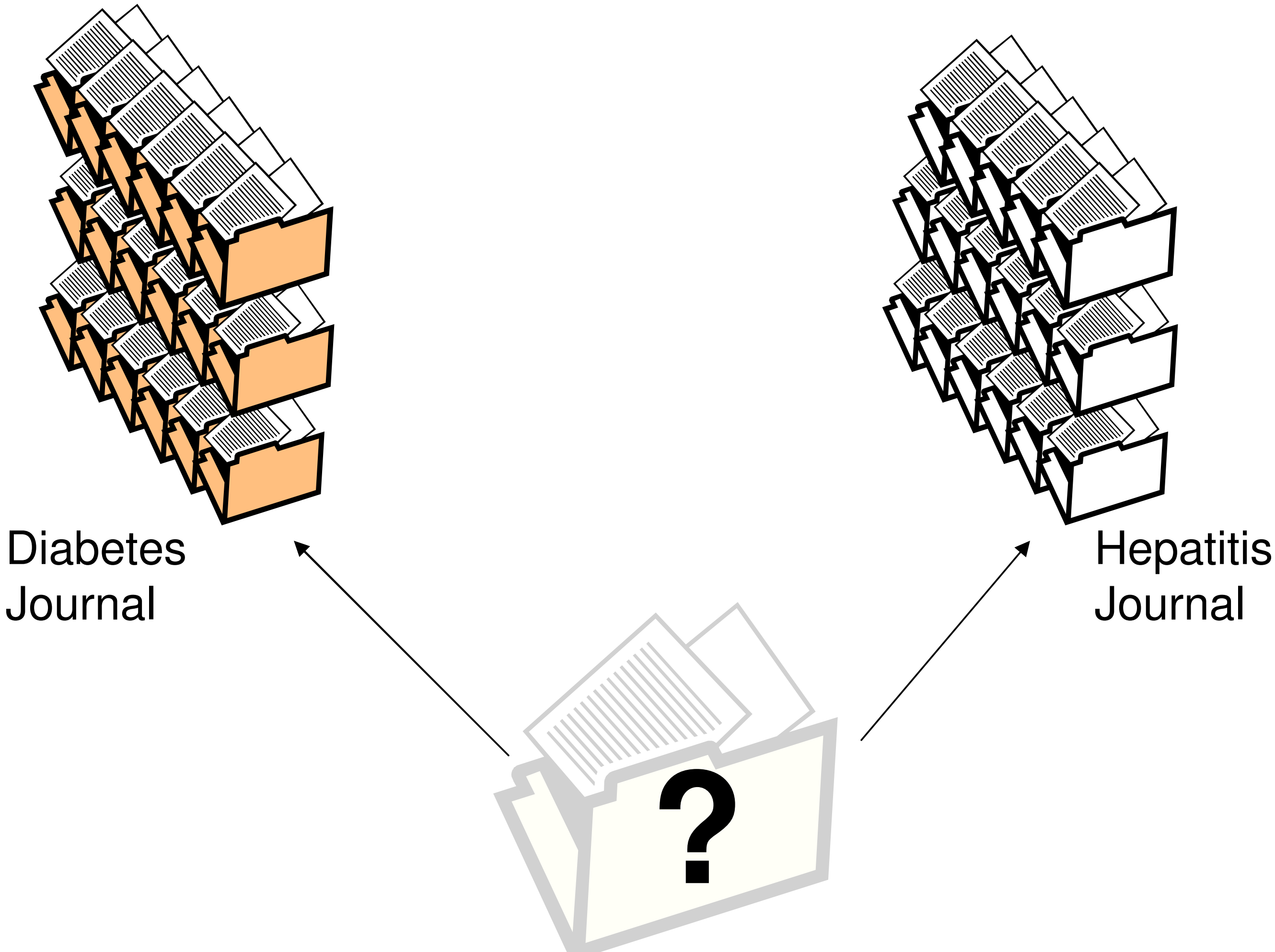
- Can we build a regression model to model such binary classes?
- Train Regression and threshold the output
  - If f(x) >= 0.7 CLASS1
  - If f(x) < 0.7 CLASS2
  - f(x) >= 0.5 ?



Regression to Classification

- Thresholding on regression function does not always work
- Gaussian assumption on noise
- When the output is binary class, we may want to dry a different technique of modeling than regression
- Many modeling techniques that will better produce class category values we want for Y
    - Linear Classifiers is one such method

Text Classification



Text Similarity

- To classify a new journal paper into either diabetes group of hepatitis group we could probably compute similarity of this document with the two groups

- How do we compute similarity between text or group of documents?

- First, we need representation of text that takes account of all the information in it?
  - Count of +ve words may not be enough

Text/Document Vectors

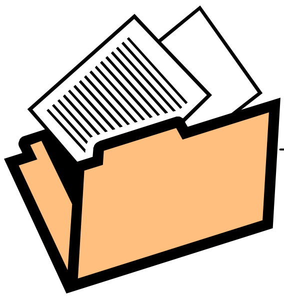
- Document Vectors
  - Documents can be represented in different types of vectors: binary vector, multinomial vector, feature vector

- Binary Vector: For each dimension, 1 if the word type is in the document and 0 otherwise

- Multinomial Vector: For each dimension, count # of times word type appears in the document

- Feature Vector: Extract various features from the document and represent them in a vector. Dimension equals the number of features

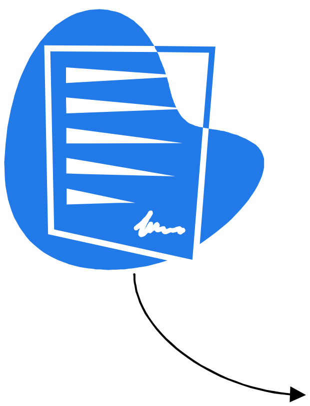
Example of a Multinomial Document Vector



Screening of the critically acclaimed film NumaFung Reserved tickets can be picked up on the day of the show at the box office at Arledge Cinema. Tickets will not be reserved if not paid for in advance.

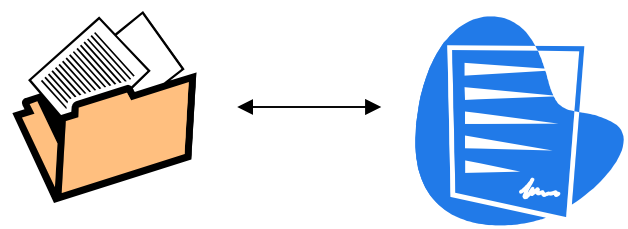
4 THE	4
2 TICKETS	2
2 RESERVED	2
2 OF	2
2 NOT	2
2 BE	2
2 AT	2
1 WILL	1
1 UP	1
1 SHOW	1
1 SCREENING	1
1 PICKED	1
1 PAID	1
1 ON	1
1 OFFICE	1
1 NUMAFUNG	1
1 IN	1
1 IF	1
1 FOR	1
1 FILM	1
1 DAY	1
1 CRITICALLY	1
1 CINEMA	1
1 CAN	1
1 BOX	1
1 ARLEDGE	1
1 ADVANCE	1
1 ACCLAIMED	1

Example of a Multinomial Document Vector



4	THE	4
2	SEATS	2
2	RESERVED	2
2	OF	2
2	NOT	2
2	BE	2
2	AT	2
1	WILL	1
1	UP	1
1	SHOW	1
1	SHOWING	1
1	PICKED	1
1	PAID	1
1	ON	1
1	OFFICE	1
1	VOLCANO	1
1	IN	1
1	IF	1
1	FOR	1
1	FILM	1
1	DAY	1
1	CRITICALLY	1
1	CINEMA	1
1	CAN	1

Find out how similar two text vectors are to find document similarity?



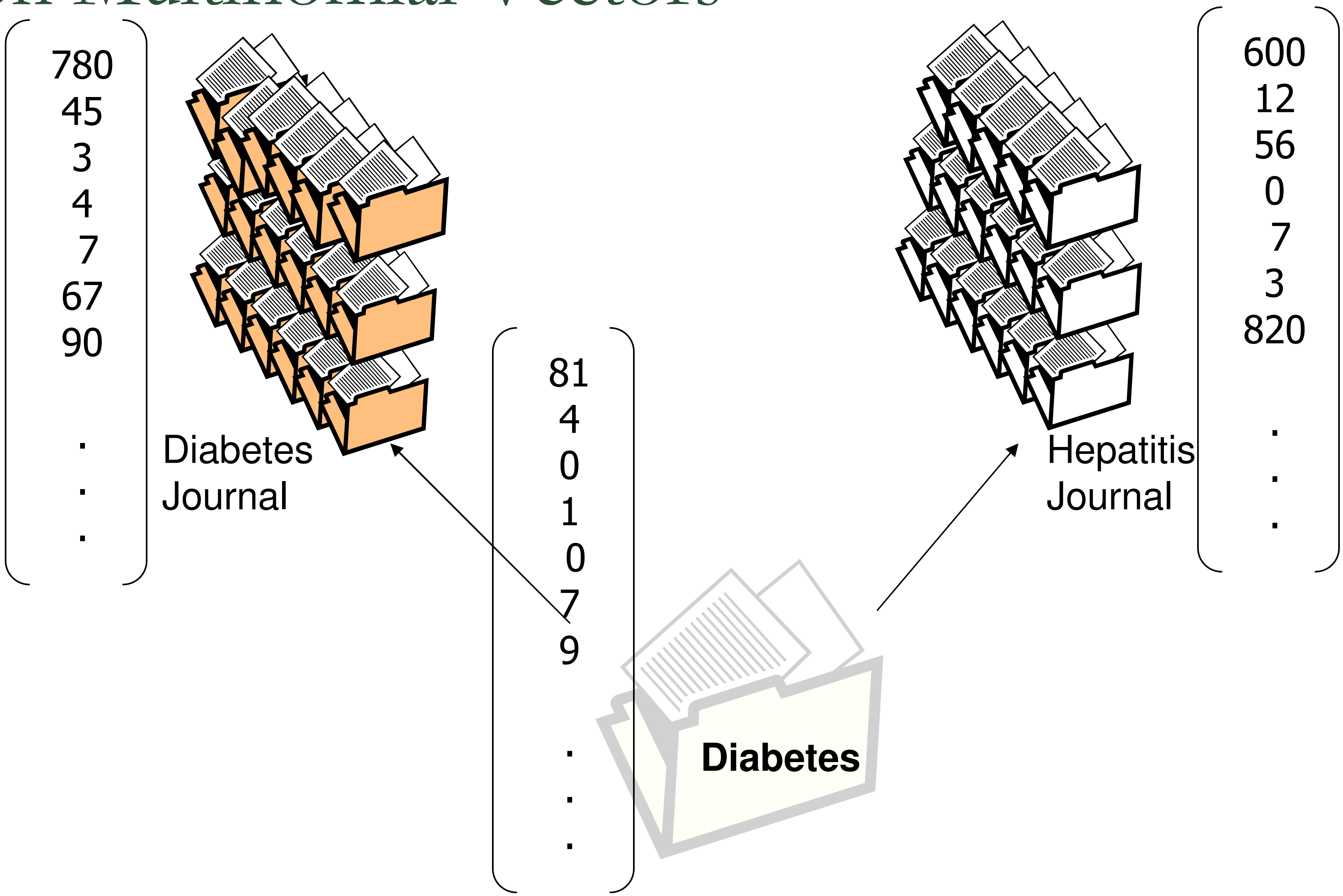
Text Similarity : Cosine of Text Vectors

- Given a set of vectors we can find the cosine similarity between them

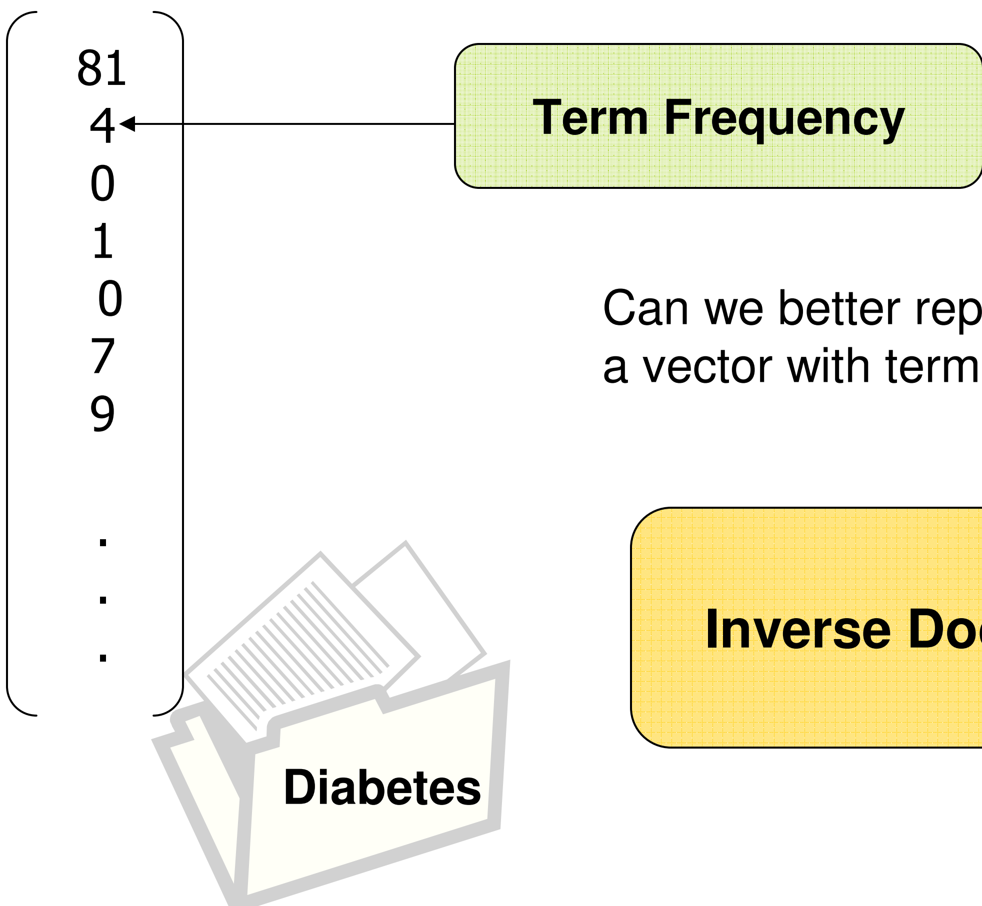
$$Cos\theta = \frac{v_1 . v_2}{\|v_1\| \|v_2\|}$$

- Cos 90 = 0, vectors are not similar
- Higher cosine value = higher similarity

Text Classification with Cosine Similarity on Multinomial Vectors



TF.IDF (Term Frequency and Inverse Document Frequency)



Can we better represent the text besides
a vector with term frequency for classification?

Inverse Document Frequency?

Inverse Document Frequency

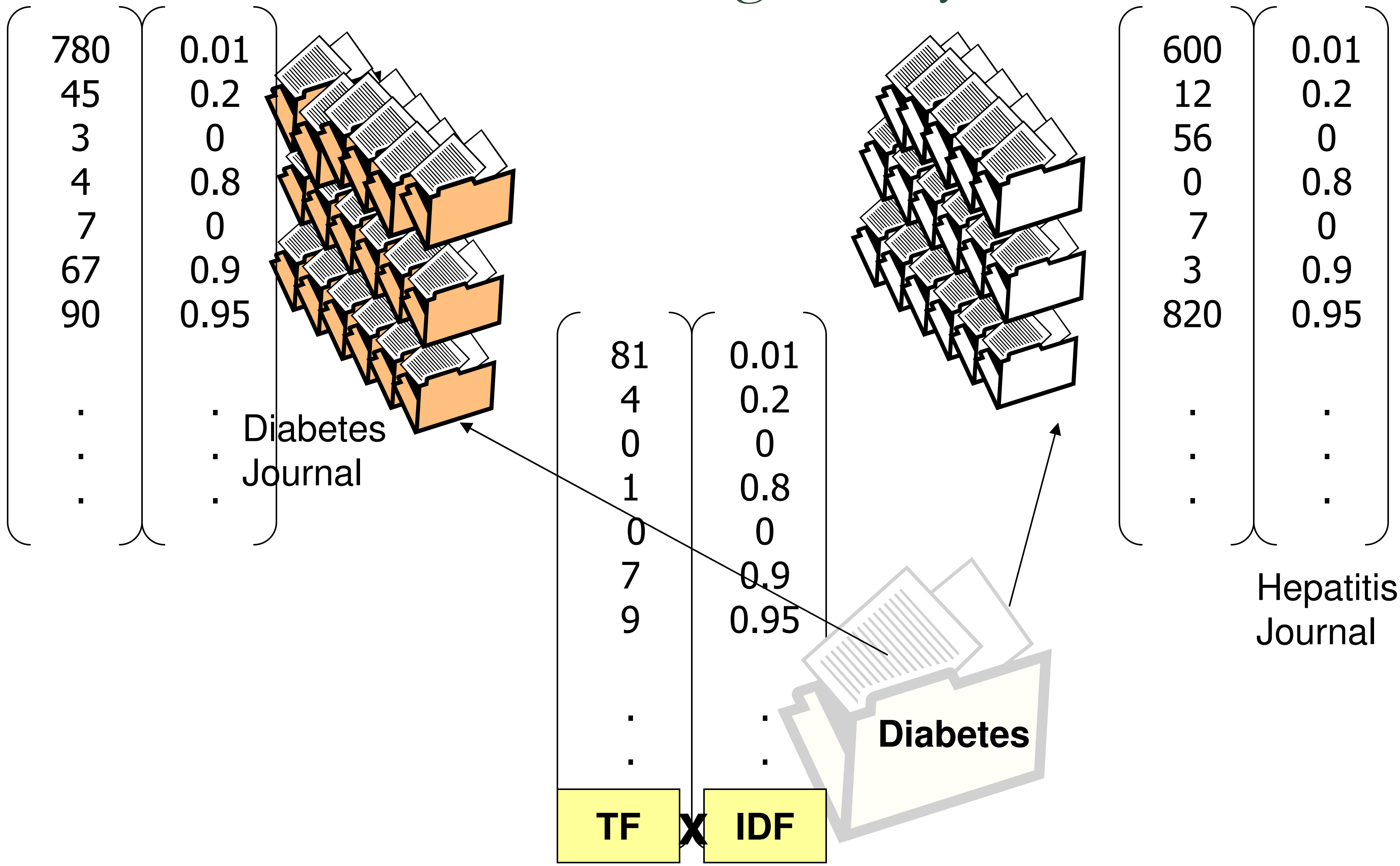
- Some words occur a lot no matter what the class of the document is
  - ‘the, of, an, …”

- Find words that occur in a fewer number of documents
  - These words could be more discriminating across documents

Inverse Document Frequency Weighting

- IDF = $\log(N/n_i)$
  - Where N is the total number documents
  - n_i is the total number of documents the word occur in
  - Fewer the documents word occur in higher the IDF value
- Words such as 'the, a, on, … ' will occur in many document so will have lower IDF value
- Multiply TF with IDF to get TF.IDF weighting of words in our multinomial vector

Text Classification with Cosine Similarity on Multinomial Vectors Weighted by TF.IDF



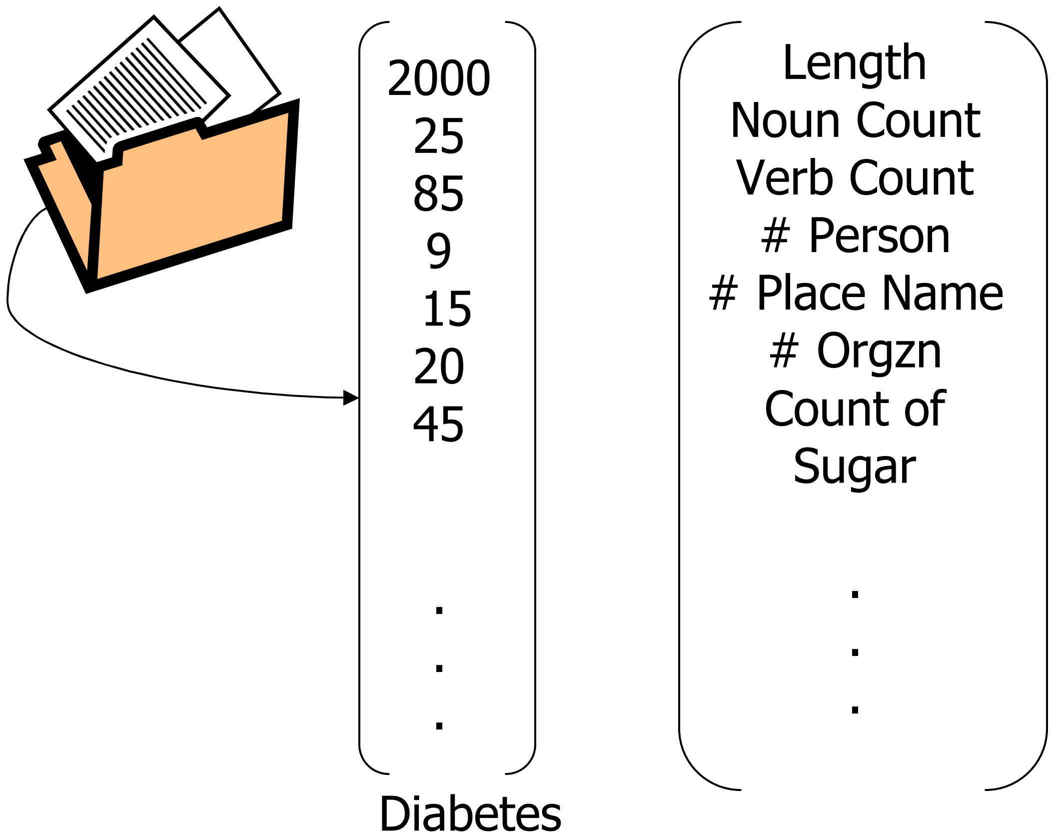
Feature Vectors

- Instead of just using words to represent documents, we can also extract features and use them to represent the document

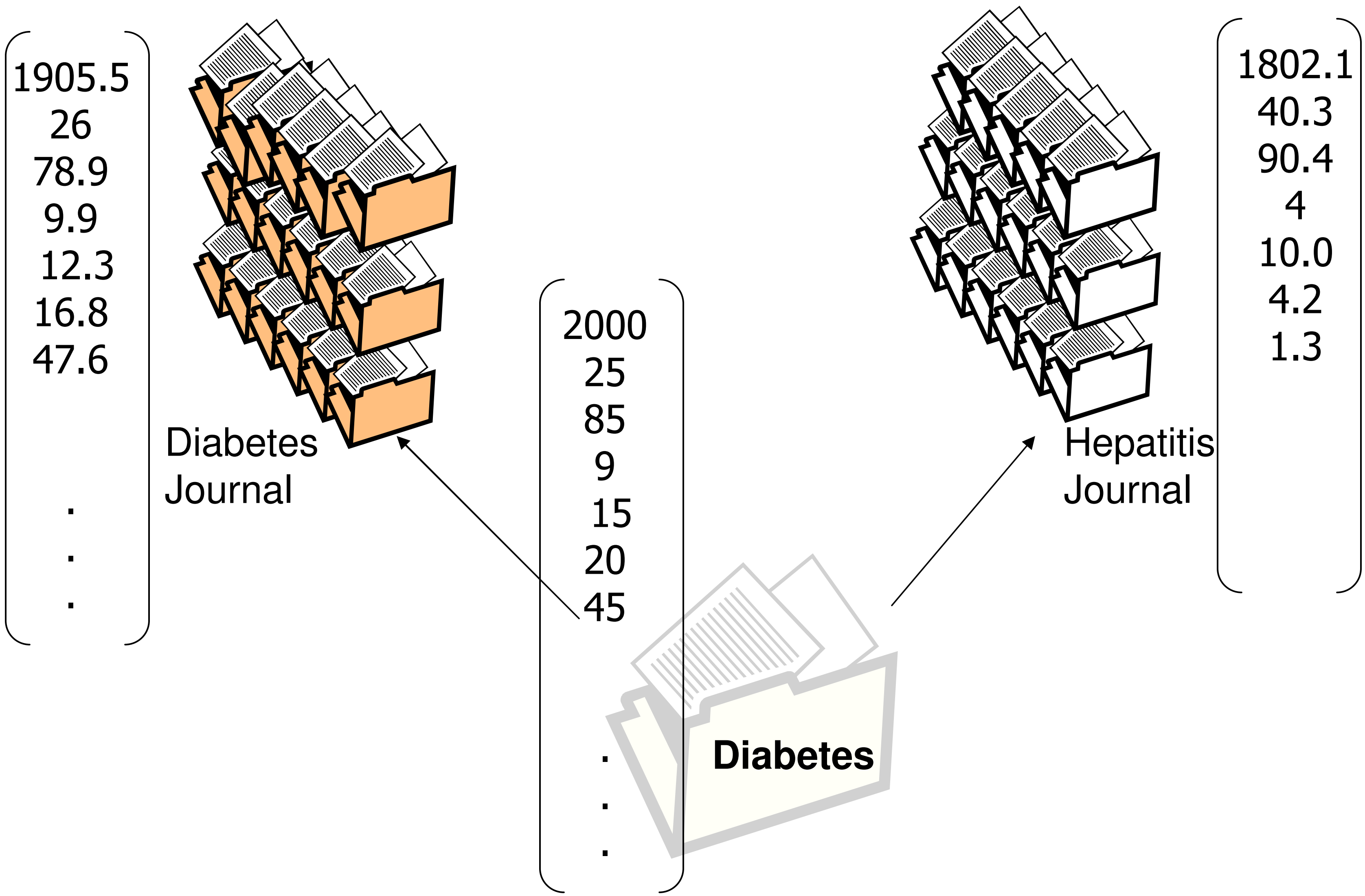
- We can extract features like document length (LN), number of nouns (NN), number of verbs (VB), number of person names (PN), number of place (CN) names, number of organization names (ON), number of sentences (NS), number of pronouns (PNN)

Feature Vectors

- Extracting such features you get a feature vector of length ‘K’ where ‘K’ is the number of dimensions (features) for each document



Text Classification with Cosine Similarity on Feature Vectors

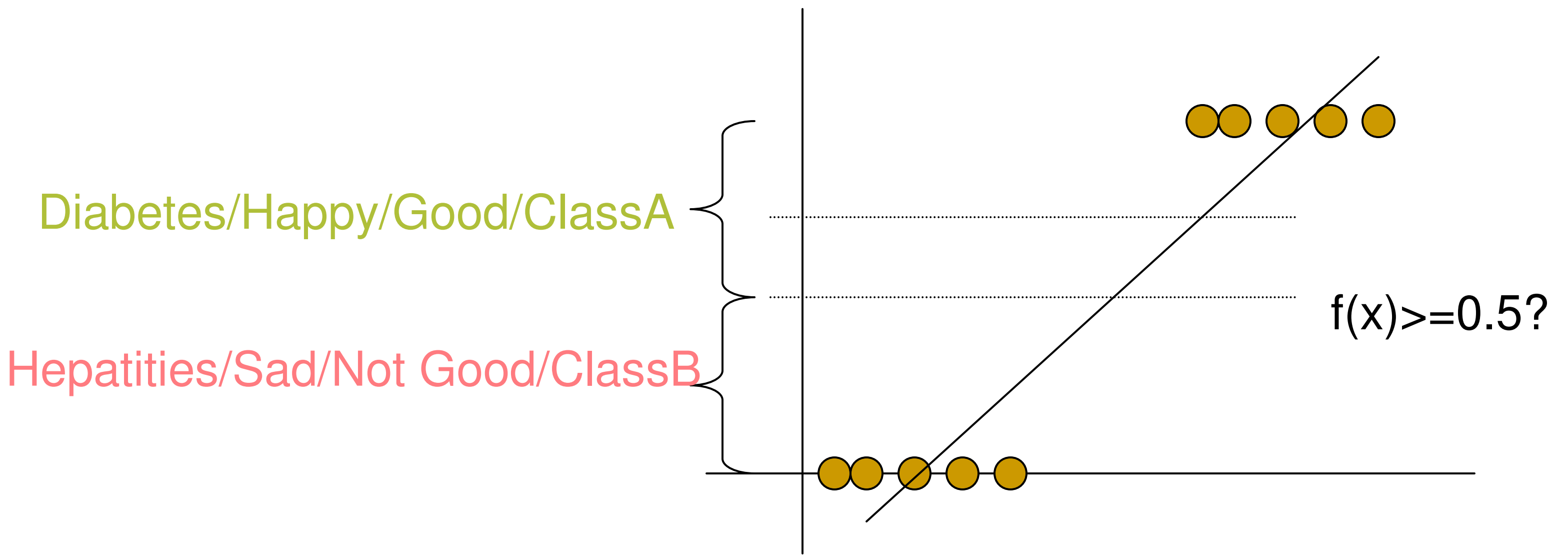


Cosine Similarity Based Text Classifier

- Build multinomial vectors or feature vectors for each document in the given class
  - Each dimension represent count of the given word or feature
  - Can take average of the vectors to represent a corpus
- ‘N’ averaged vectors would be the model for ‘N’ classes of the documents
- For any new document compute similarity of its multinomial vector to the ‘N’ class vectors
- Highest Cosine Similarity represents the class

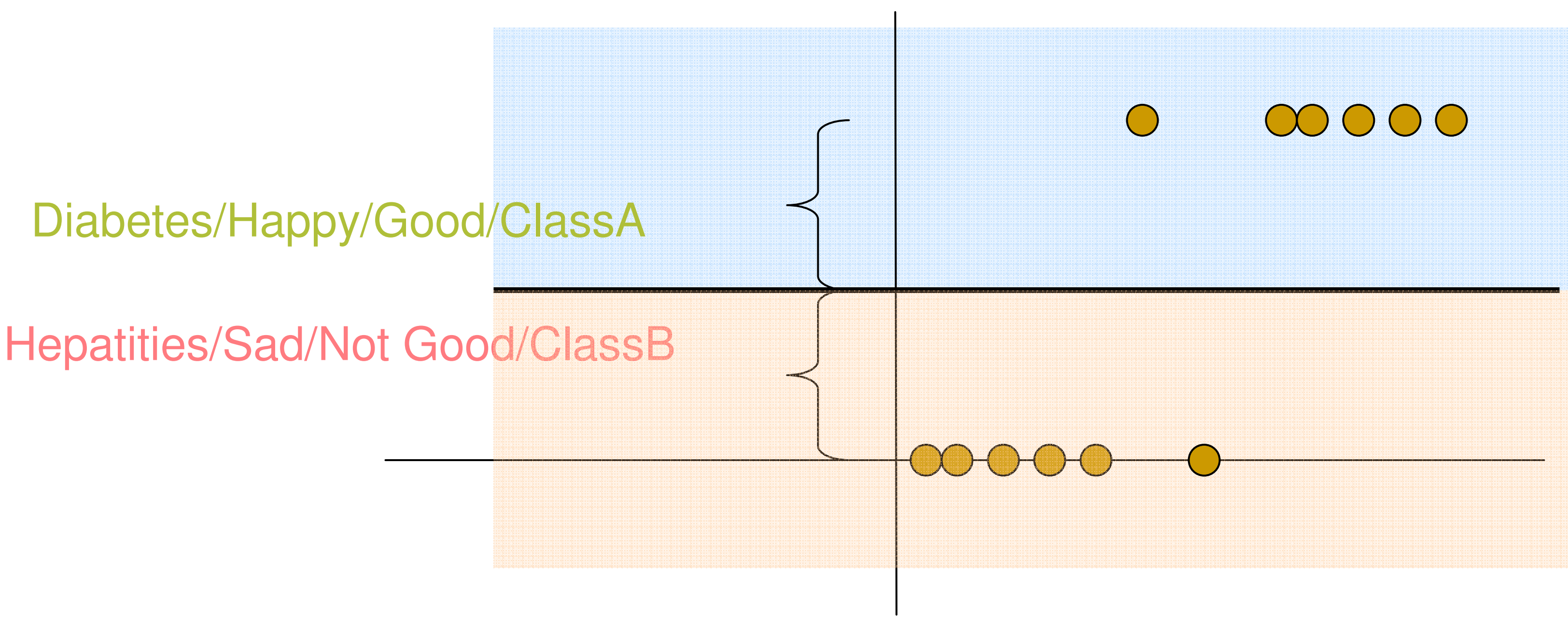
Text Classification using Regression?

- Regression for binary text classes not ideal
  - 'Diabetes vs. Hepatitis'
- We want better modeling technique



Half Plane and Half Spaces

- Half plane is a region on one side of an infinite long line, and does not contain any points from other side
- Half space n-dimensional space obtained by removing points on one side of hyperplane (n-1 dimension)
  - What would it look like for a 3 dimensional space



Linear Discriminant Functions

- A linear discriminant function is defined by

$$f(x) = w^T x + w_0$$

  - where ‘w’ is the weight vector and w_0 is bias

- For a binary classifier

$$\text{Decision Surface } f(x) = 0$$

$$\text{Class } C_0 \text{ if } f(x) > 0$$

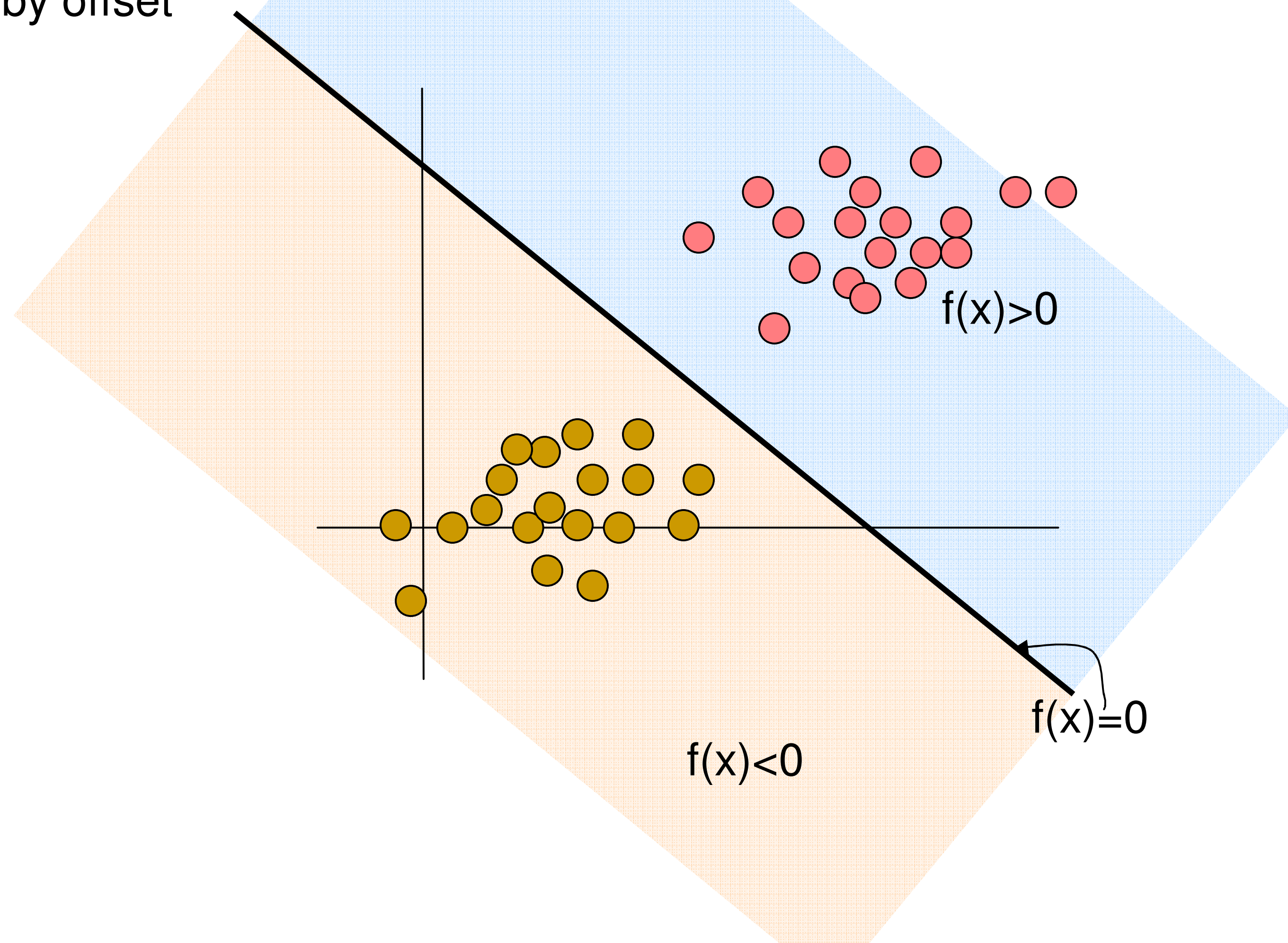
$$Class \mathrm{C}_1 \text{ if } f(x) < 0$$

Decision Surface

→ Linear Discriminant function divides the space features live in by a hyperplane decision surface

→Orientation is defined by normal vector (**w**) to the decision surface

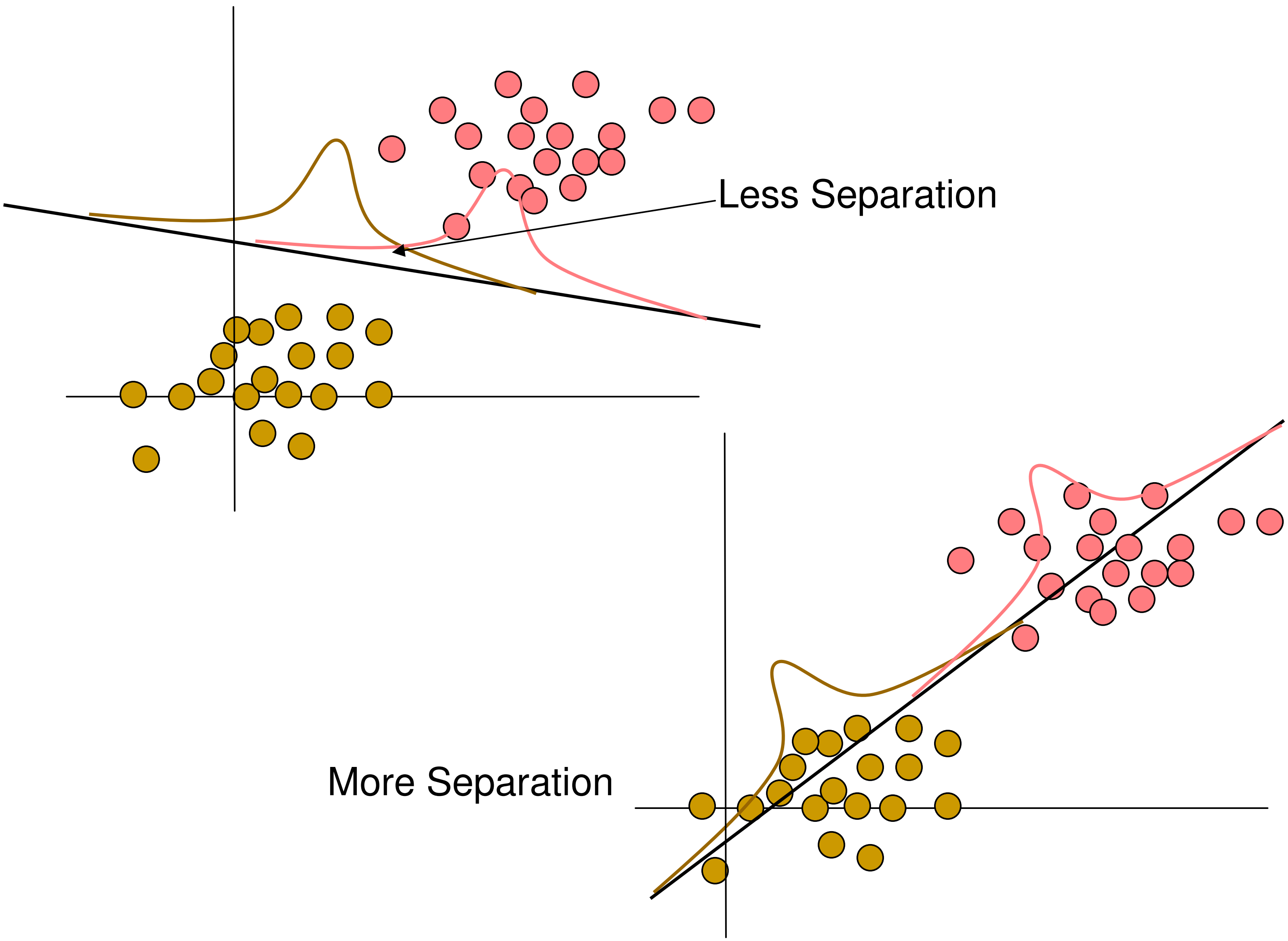
→Location defined by offset



How Do We Find the Decision Surface

- We want to find a decision surface that will produce least class error in the training data
  - Fisher's Linear Discriminant
    - Dimensionality reduction, project data on a line and classify
  - Linear Discrimination with a hyperplane in (d-1) dimension
    - Perceptrons

Varying Levels of Separation



Fisher's Linear Discriminant Function

- We want to find the direction (w) of the decision surface such that points are well separated
  - Project points to a line
  - Compute mean and variances for the classes

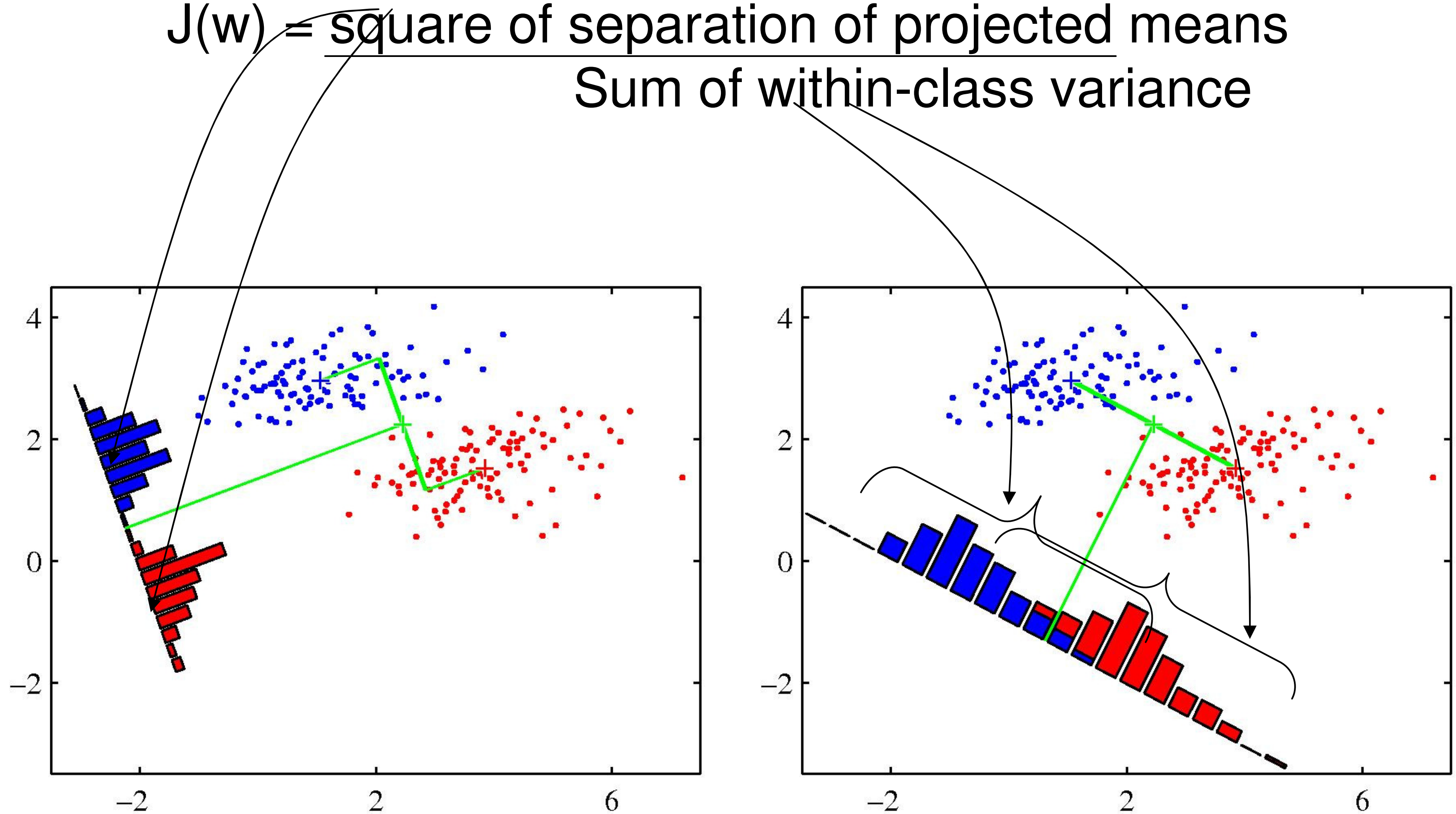
  - Maximize

$$J(w) = \frac{\text{square of separation of projected means}}{\text{Sum of within-class variance}}$$

Maximize a function that will produce large separation between class means (projected) and has smaller within-class variance

Why This Criteria Makes Sense?

$$J(w) = \frac{\text{square of separation of projected means}}{\text{Sum of within-class variance}}$$



Projecting points on a line [1]

Fisher's Linear Discriminant

Projected means

$$J(\mathbf{w}) = \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_0)^2}{\mathbf{w}^T(\sum_1 + \sum_0)\mathbf{w}}$$

$$\mathbf{w} = (\sum_1 + \sum_0)^{-1}(\mu_1 - \mu_0)$$

Higher the interclass difference in means and
lower the in-class variance better
is the model

Linear Discrimination with a Hyperplane

- Dimensionality reduction is one way of classification
- We can also try to find they discriminating hyperplane by reducing the total error in training
  - Perceptrons is one such algorithm

Perceptron

- We want to find a function that would produce least training error

$$R_n(w) = \frac{1}{n}\sum_{i=1}^{n} Loss(y_i, f(x_i; w))$$

Can't we find closed form
Solution by setting derivatives to zero?

Minimizing Training Error

$$\text{Given training data} < (x_i, y_i) >$$
$$\text{We want to find } w \text{ such that}$$
$$(w.x_i) > 0 \text{ if } y_i = -1 \text{ misclassified}$$
$$(w.x_i) < 0 \text{ if } y_i = 1 \text{ is misclassified}$$

- We can iterate over all points and adjust the parameters

$$w \leftarrow w + y_i x_i$$

$$\text{if } y_i \neq f(x_i; w)$$

- Parameters are updated only if the classifier makes a mistake

Perceptron Algorithm

We are given (x_i, y_i)

Initialize w

Do until converged

$\quad\quad$ if $\text{error}(\text{y}_i, sign(w.x_i)) == TRUE$

$\quad\quad\quad\quad w \leftarrow w + y_i x_i$

$\quad\quad$ end if

End do

If predicted class is wrong, subtract or add that point to weight vector

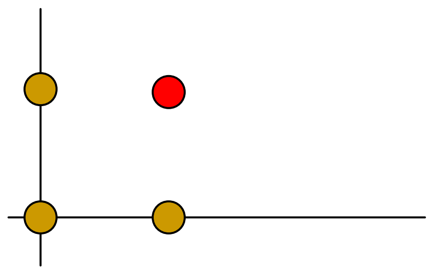
Another Version

Y is prediction based on weights and it's either 0 or 1 in this case

$$Y_j(t) = f[w(t).x_j]$$

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{i,j}$$

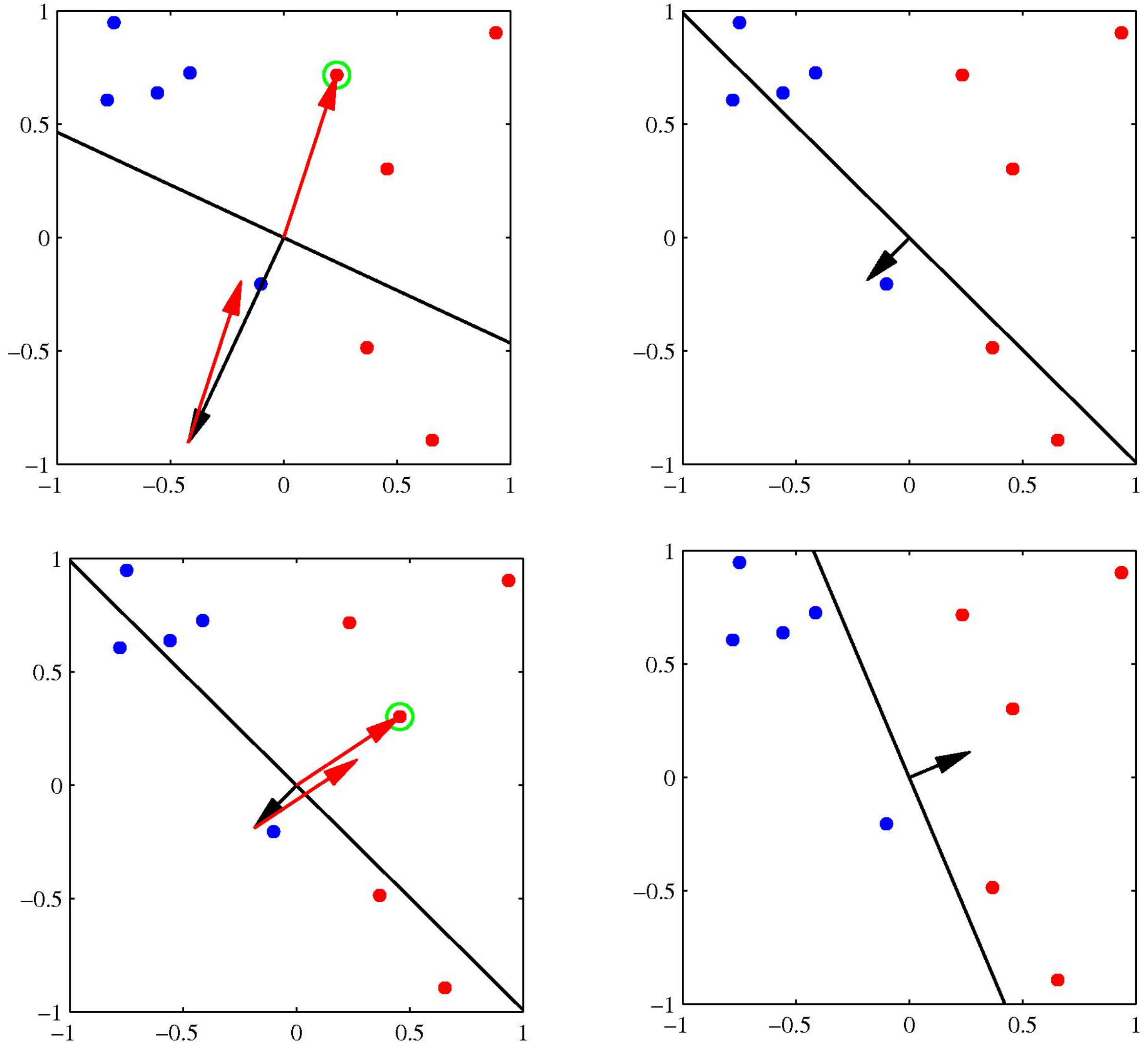
Error is either 1, 0 or -1



Input				Initial weights			Output					Error	Correction	Final weights		
Sensor values			Desired output				Per sensor			Sum	Network					
x_0	x_1	x_2	z	w_0	w_1	w_2	c_0	c_1	c_2	s	n	e	d	w_0	w_1	w_2
							$x_0 * w_0$	$x_1 * w_1$	$x_2 * w_2$	$c_0 + c_1 + c_2$	if $s > t$ then 1, else 0	$z - n$	$r * e$	$\Delta(x_0 * d)$	$\Delta(x_1 * d)$	$\Delta(x_2 * d)$
1	0	0	1	0.4	0	0.1	0.4	0	0	0.4	0	1	+0.1	0.5	0	0.1
1	0	1	1	0.5	0	0.1	0.5	0	0.1	0.6	1	0	0	0.5	0	0.1
1	1	0	1	0.5	0	0.1	0.5	0	0	0.5	0	1	+0.1	0.6	0.1	0.1
1	1	1	0	0.6	0.1	0.1	0.6	0.1	0.1	0.8	1	-1	-0.1	0.5	0	0

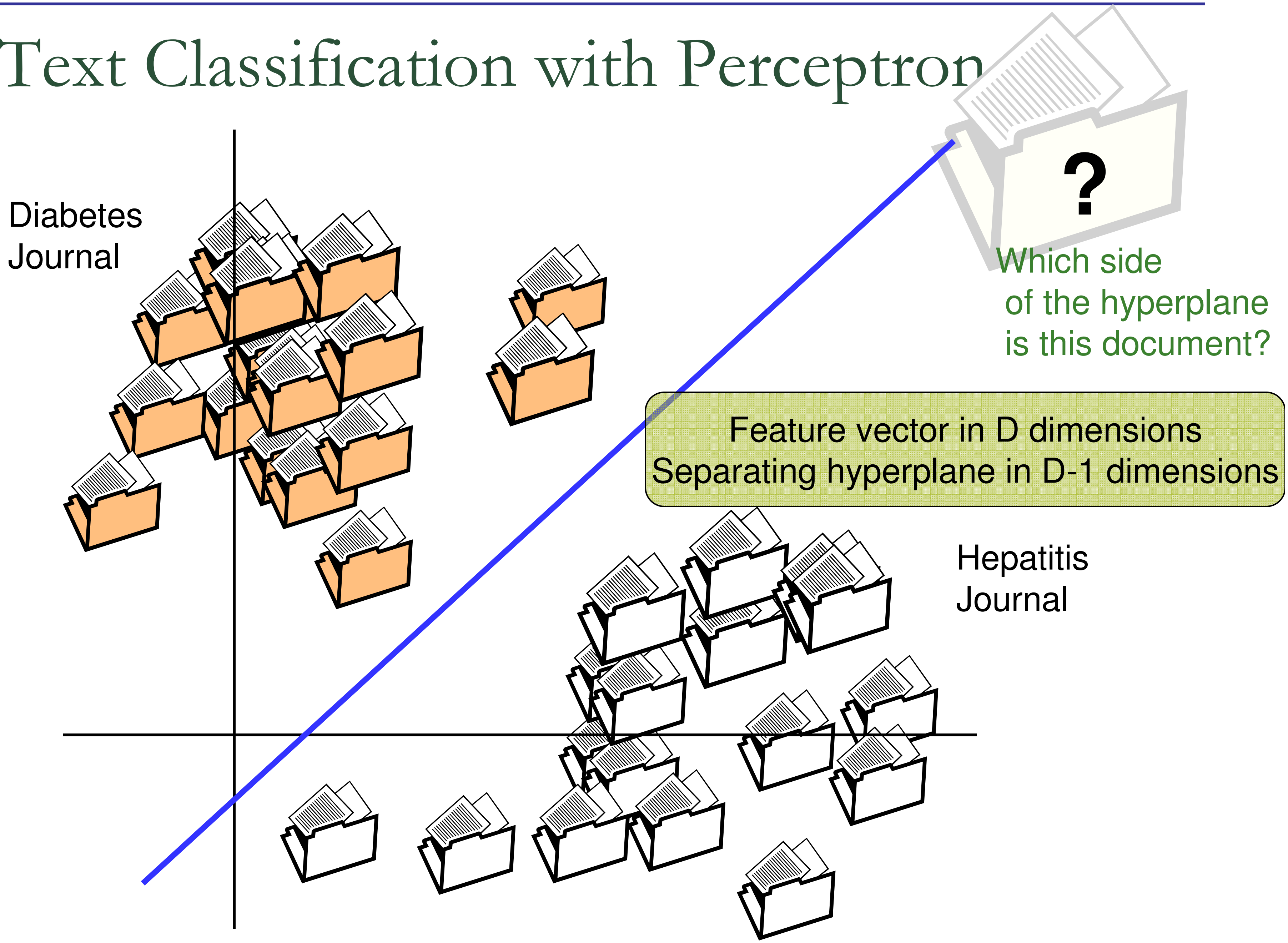
Example from Wikipedia

Why Algorithm Makes Sense?



Convergence illustration [1]

Text Classification with Perceptron



Text Classification with Perceptron

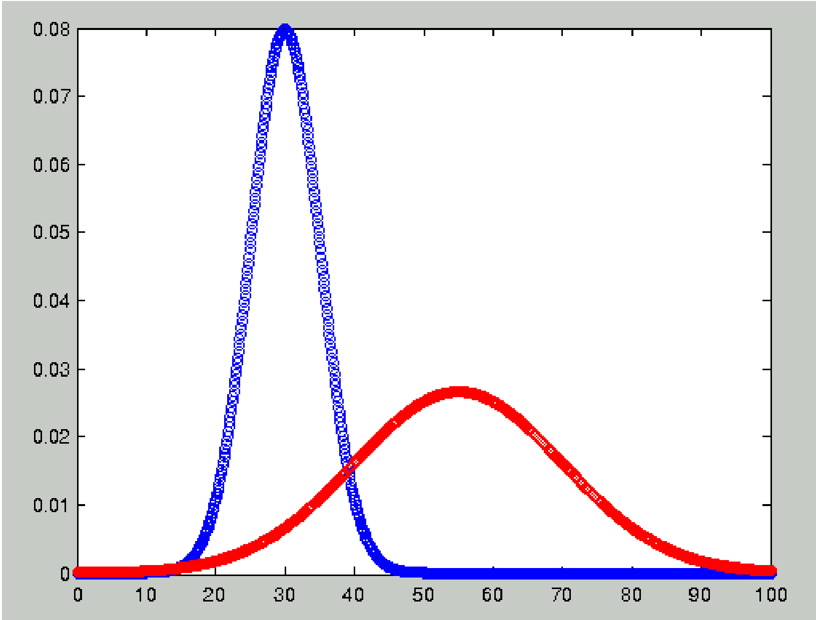
- Perceptron may not always converge
- Ok for two classes, not trivial to extend it to multiple classes
- Not the optimal hyperplane
  - Many hyperplanes that separates the class
  - Depends on random initialization

Generative vs. Discriminative

- Generative Classifier
  - Model joint probability p(x,y) where x are inputs and y are labels
  - Make prediction using Bayes rule to compute p(y|x)
- Discriminative Classifier
  - Try to predict output directly
  - Model p(y|x) directly

Generative Classifier

- We can model class conditional densities using Gaussian distributions
- If we know class conditional densities
  - p(x| y=C1)
  - p(x|y=C2)
- We can find a decision to classify the unseen example

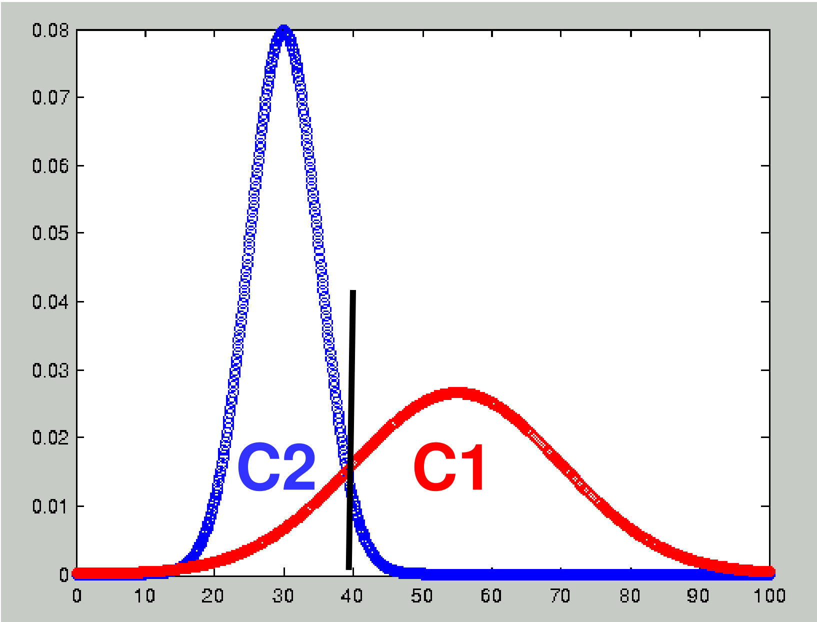


Bayes Rule

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

→So how would this rule help in classifying text in two different categories; Diabetes vs Hepatitis

→Think about distribution of count of the word diabetes for example



Generative Classifier

- If we have two classes C1 and C2
- We can estimate Gaussian distribution of the features for both classes
  - Let's say we have a feature x
    - x = length of a document
  - And class label (y)
    - y = 1 diabetes or 2 hepatitis

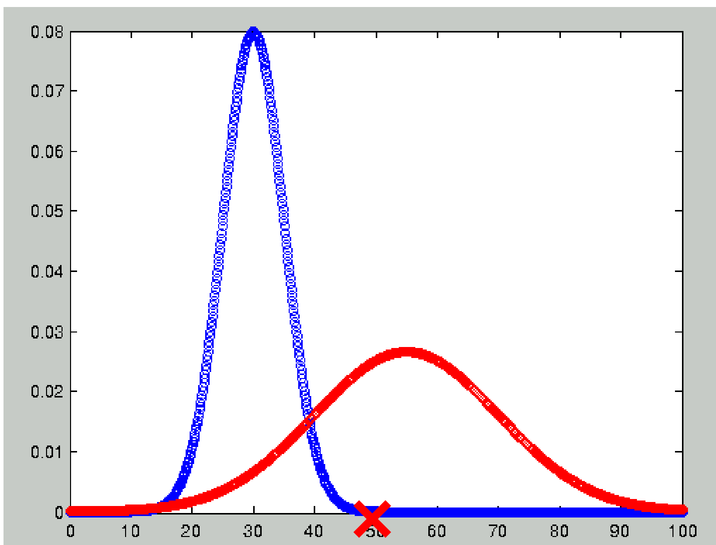
(30, 1)
(55, 2)
(24, 1)
(40, 1)
(35, 2)
…

$$N(\mu_1, \sum_1)$$
$$N(\mu_2, \sum_2)$$

Find out μ_i and $\sum_i$ from data for both classes

Generative Classifier

- Given a new data point find out posterior probability from each class and take a log ratio
- If higher posterior probability for C1, it means new x better explained by the Gaussian distribution of C1



$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

$$p(y=1|x) \propto p(x|\mu_1, \sum_1)p(y=1)$$

Naïve Bayes Classifier

- Naïve Bayes Classifier a type of Generative classifier
  - Compute class-conditional distribution but with conditional independence assumption
- Shown to be very useful for text categorization

Conditional Independence

- Given random variables X, Y,Z, X is conditionally independent of Y given Z if and only if

$$P(X,Y|Z) = p(X|Z)P(Y|Z)$$

$$\begin{aligned} P(X|Y) &= P(X_1, X_2|Y) \\ &= P(X_1|Y)P(X_2|Y) \end{aligned}$$

Conditional Independence

- For a feature vector with ‘n’ features we get

$$P(X_1, X_2, ..., X_N|Y) = \Pi_{i=1}^{N} P(X_i|Y)$$

N features are conditionally independent of one another given Y

Why would this assumption help?

Naïve Bayes Classifier for Text

$$P(Y_k, X_1, X_2, ..., X_N) = P(Y_k)\Pi_i P(X_i|Y_k)$$

Prior Probability of the Class

Conditional Probability of feature given the Class

Here N is the number of words, not to confuse with the total vocabulary size

Naïve Bayes Classifier for Text

$$P(Y = y_k | X_1, X_2, ..., X_N) = \frac{P(Y=y_k)P(X_1,X_2,..,X_N|Y=y_k)}{\sum_j P(Y=y_j)P(X_1,X_2,..,X_N|Y=y_j)}$$

$$= \frac{P(Y=y_k)\Pi_i P(X_i|Y=y_k)}{\sum_j P(Y=y_j)\Pi_i P(X_i|Y=y_j)}$$

$$Y \leftarrow argmax_{y_k} P(Y = y_k)\Pi_i P(X_i|Y = y_k)$$

Naïve Bayes Classifier for Text

- Given the training data what are the parameters to be estimated?

$$P(Y) \qquad P(X|Y_1) \qquad P(X|Y_2)$$

$P(Y)$	$P(X\|Y_1)$	$P(X\|Y_2)$
Diabetes : 0.8 Hepatitis : 0.2	the: 0.001 diabetic : 0.02 blood : 0.0015 sugar : 0.02 weight : 0.018 …	the: 0.001 diabetic : 0.0001 water : 0.0118 fever : 0.01 weight : 0.008 …

Equations to Implementation

$$P(X|Y_1) \qquad P(X|Y_1) = \Pi_i P(X = x_i | Y = y_1)$$

$$\theta_{i,j,k} \equiv P(X_i = x_{ij} | Y = y_k)$$

the: 0.001
diabetic : 0.02
blood : 0.0015
sugar : 0.02
weight : 0.018
…

MLE Estimation of the parameters

$$\hat{\theta}_{i,j,k} = \hat{P}(X_i = x_{ij} | Y = y_k)$$
$$= \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

#D{x} = number of elements in the set D that has property x

Equations to Implementation

- Count the number of documents ‘sugar’ occurs and Class=Hepatitis {C(s,H)}
- Count the number of documents ‘sugar’ occurs and Class=Diabetic {C(s,D))
- Count the number of Hepatitis Document {C(H)}
- Count the number of Diabetes Document {C(D)}
- Probability ‘sugar’ occurs for Class=Diabetes is C(s,D)/C(D)

References

- [1] Christopher M Bishop, “Pattern Recognition and Machine Learning” 2006