LIBGUNDAM Tutorial

Ming-Hen Tsai
Alumni of Computer Science Department, National Taiwan University
Taipei 106, Taiwan

1. Introduction

The document introduces the functions of LIBGUNDAM[^1], which is a tool for learning good feature representations fast in an unsupervised (without label information) manner.

In many tasks, good feature representation exempt us from building large learning systems, which significantly reduces our developing time and training time. In addition, the needs for learning large-scale and sparse data fast is a vital issue in many applications.

LIBGUNDAM is such a package to made to meet the needs above written mainl y in C++. The learned representation is in \( \text{SVM}^{\text{light}} \) format that can directly be used by classifiers like LIBSVM[^2], LIBLINEAR[^3] and FEST[^4]. If you are not familiar with any of the tools above, please check the websites for detail.

2. Algorithms

We let \( X = [x_1 x_2 \ldots x_l]^T \), where \( x_i \in \mathbb{R}^{n \times 1} \) means the i-th training instance, to denote the set of training data. In the settings of this paper, in each dimension, non-numerical features are converted to numerical form by mapping them to values sequentially from one to number of categorical variables.

2.1. Categorical Feature Expansion

In this section, we try to explain some implementation details and give the users intuitions on using the techniques.

2.1.1. What is Categorical Feature

Categorical features are features that represents a category rather than a magnitude. For example, let each row be features of a student and column \( j \) of \( X \) contains each student’s id, say \( \{1, 2, 3, \ldots, n\} \). The student id is a categorical feature.

[^1]: www.csie.ntu.edu.tw/~b95028/software/lib-gundam
[^2]: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[^3]: http://www.csie.ntu.edu.tw/~cjlin/liblinear/
[^4]: http://www.cs.cornell.edu/~nk/fest/
2.1.2. Expansion of Categorical Feature

There is a potential problem for representing categorical features in the form above. For illustration, we consider a problem to predict if a student will fail by its id with his/her past records. For each instance, we have indicators \( y = \{0, 1\} \) indicating if a student passed some course and \( X = [x_1, x_2, \ldots, x_{n-1}, x_l]^T \) indicating the student’s id on the corresponding course.

If we are using a linear discriminant classifier, say support vector machines, which judge a instance’s label by its relation to the hyperplane, we may finally have a plane, say \( f(x) = wx + b \), which means As a student’s number is less than \( \frac{b}{w} \), he/she is more unlikely to succeed. It is unreasonable as we know commonly student id is not correlated to the pass rate of a student.

Thus, we need to transform the features to make the linear discriminant work. Assume there are \( n \) students. We expand it as a \( \bar{X} \in \mathbb{R}^{l \times n} \), where

\[
\bar{X}_{ij} = [X_i = j], \text{where } [] \text{ is } 1 \text{ is the statement inside is true and is } 0 \text{ otherwise.}
\]

The new feature matrix enables our linear discriminant to deal with the difference of each individual. The method is used in Lo et al.

2.2. Two Degree Polynomial Feature Expansion

Chang et al. (2010) gave a thorough comparison on accuracy and speed by using explicit two-degree polynomial expansion on data. They also proposed a fast algorithm to do expansion on sparse data. Here, we not only used some implementation techniques to make the computation faster but extend it to a more general setting.

2.2.1. Online Two Degree Polynomial Expansion and Condensing

2.2.2. Expansion on Correlated Subspaces

2.3. Scaling on a Dimension

2.4. Principal Component Analysis

2.4.1. Subspace

2.4.2. Sparse Principal Component Analysis

3. Usages

The codes are not as updated as the algorithms introduced above because writing packages and doing individual tests may take some time.

3.1. Installation

After downloading the software, you must go to the software home directory and type ”make” to install the software. You might also want to install LIBLINEAR by typing ”make liblinear-1.7″.
There are two commands you can use, "feature-mine" and "feature-extract". The former takes a file in SVM\textit{light} format as input and output a profile that contains information to transform the file. The latter takes a profile and a data file and outputs the preprocessed new data file.

Some python scripts to convert the data format are in the "./format-converter" directory.

### 3.2. Auto Configuration

Use "feature-mine" to do auto configuration.

#### 3.2.1. Categorical Feature Expansion

The option `-c` means to expand categorical features. A feature with no more than k distinct values are treated as categorical variables. k is at default specified as $\frac{\text{number of instances}}{10}$. You can specify k by adding a value after `-c`. Examples follow.

```bash
bash> ./feature-mine -c heart_scale heart_scale.extract
bash> ./feature-mine -c 10 heart_scale heart_scale.extract
```

In the example, the first line does categorical feature expansion using default parameters. The second line treats dimensions with unique values less than 10 and does categorical feature expansion.

#### 3.2.2. Two Degree Polynomial Feature Expansion

The option `-p` provides us to do explicit polynomial feature expansion. By specifying the option `-c` it expands dimensions that take only two values. By specifying the option `-a` it expands all dimensions. By default, it uses option `-a`.

The examples of using the option are shown below.

```bash
bash> ./feature-mine -p a heart_scale heart_scale.extract
bash> ./feature-mine -p c heart_scale heart_scale.extract
```

#### 3.2.3. Scaling on A Dimension

\textsc{Libgundam} provides basic arithmetic operations on scaling a given feature column. By specifying the option `-s`, you can use the functions. It supports to 'scale a column to [0,1]', 'scale a column to [-1,1]', and 'normalize a column' with additional parameter 'l','m','n' respectively. The examples of using the option are shown below.

```bash
bash> ./feature-mine -s l heart_scale heart_scale.extract
bash> ./feature-mine -s m heart_scale heart_scale.extract
bash> ./feature-mine -s n heart_scale heart_scale.extract
```

#### 3.2.4. Composite Parameters

If there are more than one parameters, they will be applied sequentially.
3.3. The Configuration File

3.3.1. Condensing

3.3.2. Principal Component Analysis

4. Applications

4.1. Case Study

4.1.1. Heart data

Heart is a heart disease database similar to a database already present in the repository (Heart Disease databases) but in a slightly different form.

You can see the feature details in UCI Repository. The data in SVMlight format can be found in. You can of course try to find good feature representation, but we would like to try using LIBGUNDAM here.

The data is put in the "data" directory. We did categorical data expansion and scaling to \([0, 1]\) by the following commands.

```bash
bash> ./feature-mine -c -s data/heart_scale data/heart_scale.extract
bash> ./feature-extract data/heart_scale.extract data/heart_scale\ data/heart_scale.extracted
```

You can try using LIBLINEAR with 5-fold cross-validation to check the accuracy difference.

```bash
bash> ./liblinear -1.7/train -v 5 data/heart_scale
bash> ./liblinear -1.7/train -v 5 data/heart_scale.extracted
```

5. Acknowledgement

The author appreciates Prof. Chih-Jen Lin for giving him many useful advices. He thanks Dr. Isabelle Guyon for giving him intention to start the project, and thanks Dr. Leon Bottou for his lectures in Princeton University that inspire the author a lot. He also thanks Chia-Hua Ho, his long-term cooperator in National Taiwan University, for giving him a lot of ideas on algorithms and implementation. He especially thanks all users who have provided very useful advices.

References


---

5. [http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29](http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29)