Final Project Report

Buzz Prediction in Online Social Media

Yan Jin 1520010

I. Preprocessing

A. Feature Selection

This is a binary classification problem. In the dataset, Twitter-Relative-Sigma-1000, (F. Kawala, 2013) the predicted attribute is if the event is buzz or no buzz. This attribute is Boolean: 1 meaning ‘buzz observed', 0 meaning ‘no buzz observed'. It is stored is the rightmost column.

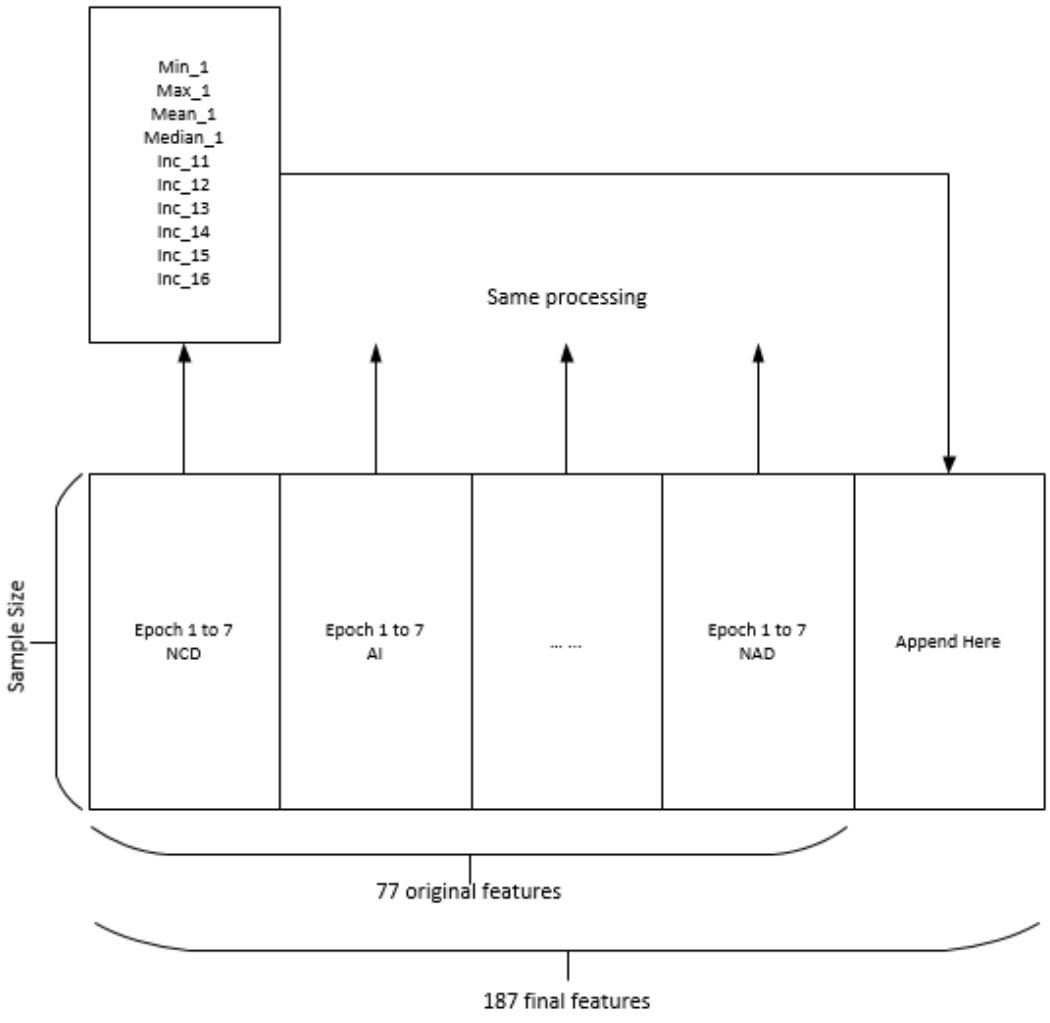

Figure 1: Preprocessing

And the predictors is organized as a time series format. Each instance is described by 77 feature, those describe the evolution of 11 'primary features' through time. Hence each feature name is post fixed with the relative time of observation.
Overall, the training dataset consists of 100,000 instances, 77 features and 1 response, while the testing dataset consists of 10,000 instances, 77 features and 1 response.
The consecutive values of a time series are usually not independent, but highly correlated. To alleviate the problem, time series can be transformed from the time domain into another domain in order to de-correlate the time features and reveal the hidden structure of the series. (Hauskrecht, 2009)
I generate five types of extra features for this classification, min, max, mean, median and increment between two epochs. Specifically, look at the Figure 1. For each 'primary feature' (11 features in total), I calculate the min, max, mean and median value of 7 epochs, and also calculate the increment between two adjacent epochs, such as$X[,i]-X[,i-1]$. These new developed features have meaningful explanation. In real online social media, we always pay close attention to the daily change of number of fans, posts or comments.

B. Training Set Reduction

On the other hand, the original training dataset consists of 100,000 data items so that it is too large to train a SVM model. We present a clustering algorithm that can be used to preprocess standard training data and show how SVM can be simply extended to deal with clustered data, which is effectively a set of weighted examples. (Theodoros Evgeniou)
The spatial complexity of SVM is N^2, independent from the dimensionality of the feature space. Then it allows us to extend the method in feature spaces of infinite dimension. But SVM presents severe limitation on the size of the training set. Hence we use clustering method to reduce the size of training set.
The idea consists of substituting the training set with a smaller set of new weighted points,

$$\{(t_1, y_1, n_1), \dots, (t_g, y_g, n_g)\}$$

So that the original SVM can be adjusted to separate the clustering set as

$$Min\left(\frac{1}{2}w^T w + C\sum n_i\varepsilon_i\right)$$
$$subject\ to\ y_i(w\varphi(t_i)+b) \geq 1-\varepsilon_i, i = 1,2,\dots,K$$
$$\varepsilon \geq 0$$

Where we have modified the second term in the objective function with a weighed sum to take in account the number of points represented by each cluster. According to this thinking, I make a k-means clustering, whose k is arbitrarily set to 10, and got 10 clusters, since it is a very imbalanced dataset, I keep all buzz data samples. Then I randomly choose a specific proportion (30%) of dataset to build up my new training set. Finally, the size of training set is about 30,000.

II. Predictors

I use two SVM models with different kernels as my training model. The first one is SVM-RBF kernel, which is SVM with Gaussian kernel, and the other one is SVM with polynomial kernel. I treat this problem as a non-linearly separable case so that C-SVM is used. The basic formulation of dual of SVM is as follows:

$$\max \sum_i \alpha_i - \frac{1}{2}\sum_{jk} \alpha_j \alpha_k y_j y_k k(x_j, x_k)$$

$$subject\ to\ 0 \le \alpha_i \le C\ for\ any\ i, and\ \sum_i \alpha_i y_i = 0$$

Then I use RBF kernel and polynomial kernel, they are

1. RBF (Gaussian) Kernel

   In machine learning, the Gaussian kernel is a popular kernel function used in support vector machine classification. The RBF kernel on two samples x, x', represented as feature vectors in some input space, is defined as

   $$k(x, x') = \exp\left(-\left\|x - x`\right\|^2 / 2\sigma^2\right)\ for\ \sigma > 0$$

   In RBF kernel, defining the transformed features in terms of the original features of a data point leads to an infinite-dimensional representation.

   To use the RBF kernel, there are two parameters to be predefined before the training procedure, σ and C.

2. Polynomial Kernel

   The polynomial kernel represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models. Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these.

   For degree-d polynomials, the polynomial kernel is defined as

   $$K(x, y) = (x^T y + C)^d$$

   To use the polynomial kernel, there are also two parameters to be predefined before the training procedure, d and C.

III. Training Strategy

In this part, I will talk about what tool I use to train the SVM model, how I choose the parameters to do the model selection, and a few things else.

A. SVM Model Training Tool

I tried several SVM packages, like SVM-torch, SVM-light and LibSVM. SVM-torch, (Ronan Collobert, 2001), is written in C++, and it has been specifically tailored for large-scale problems (such as more than 20,000 examples, even for input dimensions higher than 100). SVM light is an implementation of SVM in

C. LIBSVM is an integrated software for SVM. (Rong-En Fan, 2005). It has many other programming language interfaces and extensions to LIBSVM. In R, the package “e1071” is provided.

```
svm(x, y, type, kernel, gamma, coef0, cost, cross, probability)
```

The input of this method, x is the data matrix and y is a response vector with one label for each row/component of x. SVM can be used as a classification machine, as a regression machine, or for novelty detection, the valid types are c-classification, nu-classification, one-classification and others. Also we could specify the kernel in the function. And for different kernel, we could set different required parameters, which are from cross validation. And if we set the parameter, probability is equal to true, we could get the estimated results in the probability format rather than a classified class through the threshold method in common.

B. Description of Training Set

The original training set is imbalanced, which could be found in the following table.

	0	1
#	99144	856

The size of original training set is too large for R package “e1071” to run, hence I reduce the training set to a much smaller size set through clustering method. For the model selection part, I used about 10% of original dataset (10751) to do cross validation to get the optimized parameter from my specified range. For the afterward model training part, I used about 30% of original dataset (30325) to train the SVM model. In both procedures, I am trying to keep all the data, whose label is classified as 1 because I do not want to lose the imbalanced information.

In terms of features, I use all the generated features (110) and the original features (77), it is 187 in total. The features have 7 epochs.

C. Model Selection and Cross Validation

I also do the cross validation to find the optimized parameters. Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called over fitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test data. However, by partitioning the available data into two sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of sets. A solution to this problem is a procedure called cross validation. A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV.

In terms of RBF kernel,

```
tune.svm(x = x.cross, y = y.true, gamma = 10^(-2:-1), cost = 10^(1:2),
                    tunecontrol = tune.control(sampling = "cross", cross = 5))
```

It is a time consuming procedure, even I only use 10 percentage of training data, so that I only choose the potential gamma (0.1, 0.01), and the potential C is from (10,100). I use 5 folder cross validation to choose the good parameters.

D. Principle Component Analysis

Since the size of features is ove 100, and intuitively not all the features contribute to a good result. I am thinking using PCA to do the feature reduction. But it is not finished yet, so that I do not apply this procedure over the current model training.

Reducing dimension is an essential step before any analysis of the data can be performed. The general criterion for reducing the dimension is the desire to preserve most of the relevant information of the original data according to some optimality criteria. (Yijuan Lu, 2007)

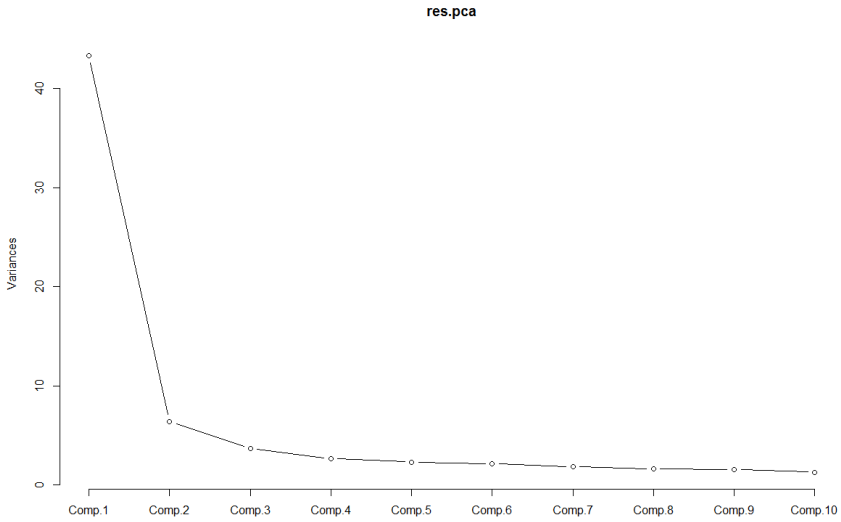

Figure 2: Variance VS. Component

From the results, I could find the cumulative proportion of variance of the first 10 components is 0.86804412. Hence theoretically we could map the original feature into the these 10 components, and then use these reduced components to train the SVM model. But I do not finish this part.

IV. Experimental results

In the experiment, I use 30% of original dataset as the final training set through clustering method, and choose randomly 50% of original dataset as the testing set to analyze the performance of predictors.

In the beginning, I use cross validation to select the parameters. $\sigma = 0.01$ and $C = 1$

If I choose the original 77 features to train the SVM model, the results without probability is as

follows,

	0	1
0	49572	151
1	11	266

The corresponding ROC curve is

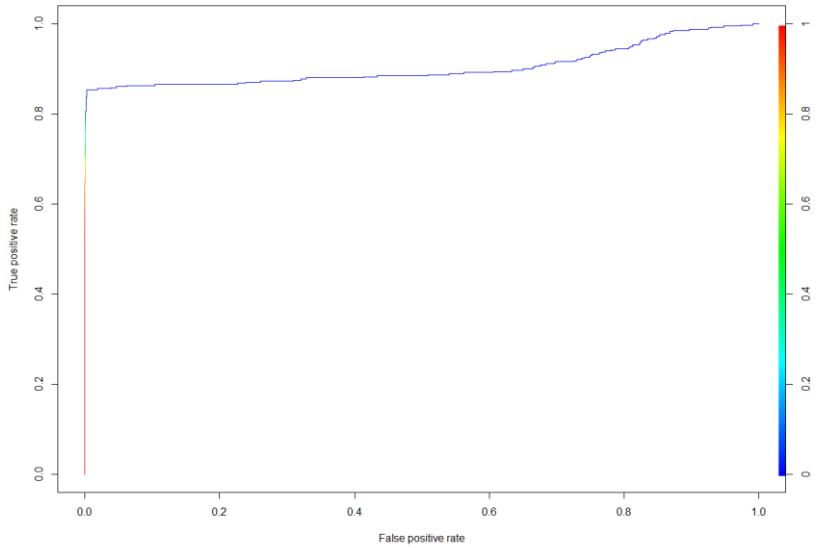

Figure 3: ROC, 77 features

The AUC of RBF kernel and 77 features could be 0.9038.
But if we use 187 features and still use RBF kernel, the results could be improved and the roc curve is as follows, and the AUC could be 0.952786.

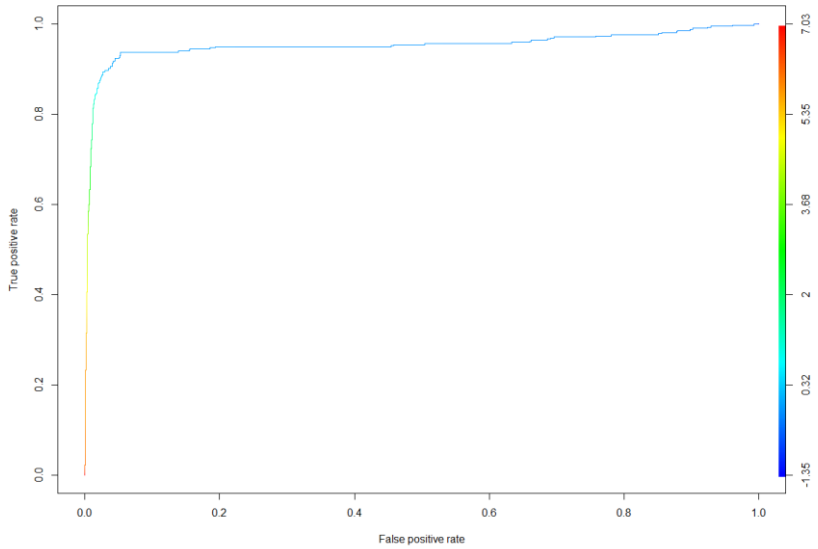

Figure 4: ROC, 187 features

And the AUC of polynomial kernel could be 0.9497012 with 187 features.

V. References

F. Kawala, A. D.-C. (2013). Predictions d'activite dans les reseaux sociaux en ligne. *Approches Mathematiques et Informatique (MARAMI)*, pp. 16.

Hauskrecht, I. B. (2009). A Supervised Time Series Feature Extraction Technique using DCT and DWT. *International Conference on Machine Learning and Applications.*

Lin, Y.-W. C.-J. (n.d.). Combining SVMs with Various Feature.

Ronan Collobert, S. B. (2001). SVMTorch: support vector machines for large-scale regression problems. *The Journal of Machine Learning Research*, 143-160.

Rong-En Fan, P.-H. C.-J. (2005). Working Set Selection Using Second Order Information. *Journal of Machine Learning Research*.

Theodoros Evgeniou, M. P. (n.d.). Support Vector Machines with Clustering for Training with Very Large Datasets.

Yijuan Lu, I. C. (2007). Feature Selection Using Principal Feature Analysis. *ACM Multimedia.* Augsburg, Germany.