Practical Project Report

Yanlin DU Yu LI Ling JIN

2 avril 2018

For the practical project, we have chosen the second paper "Random Features for Large-Scale Kernel Machines" written by Ali Rahimi and Ben Recht. The practical report consists of five parts. We will first introduce our specific task, then some datasets we have tried the method on will be presented. The third part will be the description of our implementation. The analysis of the results and the conclusion will be the last two parts.

1. Task introduction

Kernel machines such as the Support Vector Machine are widely used in solving machine learning problem, since they can approximate any function or decision boundary arbitrary well with enough training data. However, those methods applied on the kernel matrix (Gram matrix) of the data scale poorly with the size of the training dataset. The kernel trick may become intractable to compute as the computation and storage requirements for the kernel trick are exponentially proportional to the number of samples in the dataset. It takes a long time to train a model when training examples have big volume. For some specialized algorithms for linear Support Vector Machines, they operate much more quickly when the dimensionality of data is small because they operate on the covariance matrix rather than the kernel matrix of the training data.

This paper we've chosen proposes a way to combine the advantages of the linear and nonlinear approaches. This method transformed the training and evaluation of any kernel machine by mapping the input data to a randomized low-dimensional feature space in order to create corresponding operations of a linear machine. Those randomized features are designed to ensure that the inner products of the transformed data are nearly equal to those in the feature space of a user specific shift-invariant kernel. This method gives competitive results with state-of-the-art kernel-based classification and regression algorithms. What's more, random features fix the problem of large scale of training data when computing the kernel matrix. The results have similar or even better testing error.

2. Description of task and dataset

In order to implement this method, we choose several datasets respectively for classification and regression. For classification problem, we have chosen two datasets : Adult and KDDCUP99, which are also chosen in the paper. For regression problem, we choose dataset : Condition Based Maintenance of Naval Propulsion Plants Data Set. The description of each dataset is presented below.

2.1 Adult

The adult data represents different effects that relate to incomes of adults. The original data is downloaded from `https://archive.ics.uci.edu/ml/machine-learning-databases/adult/`. This data was extracted from the census bureau database. The original data is split into two thirds as

training data and one third as test data. Total data contains 48842 instances which include 32561 training instances and 16281 test instances. The prediction is to determine whether a person makes over 50K a year. There are 14 features which describe personal information like age, native country, marital status, relationship with other family members, sex, race, education, and also professional information like occupation, working sector (Machine-op-inspct), gain and loss (capital-gain and capital-loss), work time (hours-per-week). A special feature fnlwgt (final weight) refers to population totals derived from CPS by creating "weighted tallies" of any specified socio-economic characteristics of the population. The original data contains string type of data and numerical type of data. All string type of data are transformed to 0 and 1 by one-hot encoding so that the number of features is 108 in the end.

2.2 KDDCUP99

This dataset is downloaded from : `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

There are 41 features in this dataset. Some of the main features are described as below :

1. **duration** : length (number of seconds) of the connection
2. **protocol_type** : type of the protocol, e.g. tcp, udp, etc.
3. **service** : network service on the destination, e.g., http, telnet, etc.
4. **num_failed_logins** : number of failed login attempts
5. **count** : number of connections to the same host as the current connection in the past two seconds
6. **same_srv_rate** : % of connections to the same service

In the original dataset, each connection is labeled as either normal, or as an specific attack type. While the paper has make this problem much simpler as a binary classification problem, which means there are only two labels : normal and abnormal.

There are no missing data in this dataset, so we don't do any none value processing. While in the 41 features, there are three categorical features. We choose to do one-hot encoding on these three

features to transform them into numerical variables.

2.3 Condition Based Maintenance of Naval Propulsion Plants Data Set

Download from : `https://archive.ics.uci.edu/ml/datasets/Condition+Based+Maintenance+of+Naval+Propulsion+Plants#`

In fact, in order to reproduce the performance of the proposed method in algorithm, we planned to use the same dataset for regression task, for example CPU, Census. However, what we can find online doesn't have enough number of instances like mentionned in this paper. Therefore, we choose another dataset for reproducing the performance of regression task.

This dataset is generated from a sophisticated simulator of a Gas Turbines (GT), mounted on a Frigate characterized by a COmbined Diesel eLectric And Gas (CODLAG) propulsion plant type. This dataset has 11,934 instances and each sample has 18 features. What's more, we choose the feature **Gas Generator rate of revolutions** which has the largest number of unique values to be the label in such a regression task.

And we don't need to do more preprocessing of this dataset because its data type is all real and has no NaN values.

When we implement on this dataset, the first thing is to separate the train and test set. Here for this dataset, the train set makes up 67% of the whole data. Then transforming the data uses the proposed method to realize the data transformation. And choosing the linear model **Ridge** fits the transformed training data and predicts the test part.

3. Description of implementation

There are two kinds of random features introduced in this paper : random Fourier features and random binning features. For random Fourier features, we have used the implementation code of the library sklearn. The **sklearn.kernel_approximation.RBFSampler** constructs an approximate mapping for the radial basis function kernel. For random binning features, we use the implemention written by Matej Balog. The github website is `https://github.com/matejbalog/mondrian-kernel/blob/master/random_binning.py`.

In this set of experiments, we will use least squares regression on random features to approximate the training on supervised kernel machines. First, we obtain the random features of the training data, then we train regressors and classifiers by solving the least squares with ridge regularization problem $\min_{\mathbf{w}} ||\mathbf{Z}'\mathbf{w} - \mathbf{y}||_2^2 + \lambda||\mathbf{w}||_2^2$, where $\mathbf{y}$ denotes the vector of target, and $\mathbf{Z}$ denotes the matrix of random features. To evaluate the resulting machine on a test data point $\mathbf{x}$, we can simply compute $\mathbf{w}'\mathbf{z}(\mathbf{x})$.

We also encounter several problems during the implementations. The main reason that causes problems are that the paper don't mention exactly and clearly how they realize experiments. The first

problem is the difficulty to find right datasets used in the paper. In the original paper, 5 datasets are used. In order to compare randomized feature with other existing kernel methods like exact SVM and CVM, authors replicate results in "Core Vector Machines : Fast SVM training on very large data sets". Authors indicate that datasets for experiments are the same as those in the referred paper. However, CPU and Census datasets for regression don't exist in the referred paper. What's more, the dimensions of CPU and Census datasets that we can find in the UCI archive are much smaller than that described in our paper. So we have to find another dataset with nearly the same scale for regression. And the possible links of data have been already expired. Secondly, the numbers of features mentioned in the paper are numbers after feature engineering. As authors don't mention how they implement exactly feature engineering, it is hard to achieve the datasets with the same dimensions. This will cause a problem of the difference of performance between experiments in the paper and our experiments. Besides, for KDDCUP99 dataset, the original target has more than 20 classes. When we did multiple classification, the performance of the algorithm was very poor. So we changes to binary classification and get reasonable results.

4. Analysis of results

To evaluate the performance of the algorithm both in classification and regression task, this paper uses the error percent to represent. For the classification task, this is just the percent of testing points incorrectly predicted. And for the regression task, it's the RMS error normalized by the norm of the ground truth, which is represented like

$$error\_rate = \frac{||y\_pred - y\_true||_2}{||y\_true||_2}$$

Table 1 shows the comparison of testing error and training time between ridge regression with random features and SVM.

TABLE 1 – Comparison of testing error and training time

Dataset	Fourier + LS	Binning +LS	Exact SVM
Adult	23.62%	23.60%	23.97%
classification	1.43 secs	12.99 secs	9.144 mins
32,000 instances	D=500	P=30	SVC
KDDCUP99	11%		
classification	27.5 secs	Memory error trained on GPU	
4,900,000 instances	D=50		
Condition based maintenance	0.225%	7.58%	9.456%
regression	0.697 secs	1.35 secs	11.38 secs
12,000 instances	D=500	P=30	SVR

From the table 1 above, it's obvious that random features combined with linear method ridge regression have a better performance in both computation time and accuracy. For example, for Condition based maintenance dataset, testing error rate of Fourier random features combined with least square regression is 0.225%, while for SVR, the test error rate is much higher : 9.456%. And the computation time of Fourier random features with least square is 0.697 secs, which is 16 times less then that of SVR. While for KDDCUP99, the largest dataset among these three datasets, the computation time of random Fourier features with ridge regression is 27.5 secs. We trained it in Azure with a GPU, SVC takes more than 10 hours so we didn't finish the training.

4.1 Variation of test errors as D changes for random Fourier features

And in order to evaluate the effect of parameter **D** respectively on the training and testing time and testing error rate, we use the random Fourier features and get the following figures.

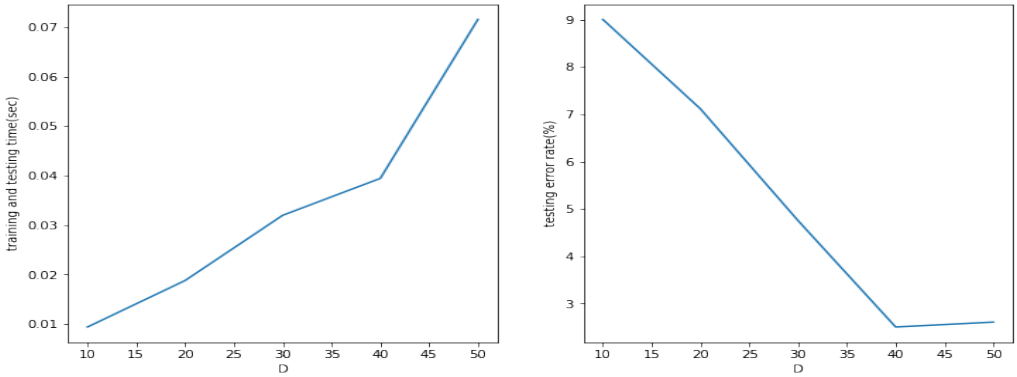

FIGURE 1 – The effect of D on time and error rate in Condition Based Maintenance of Naval Propulsion Plants Data Set

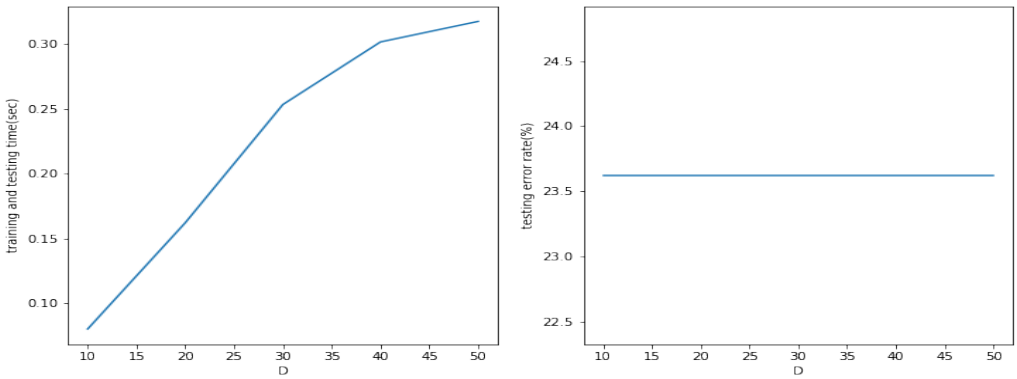

FIGURE 2 – The effect of D on time and error rate in Adult Data Set

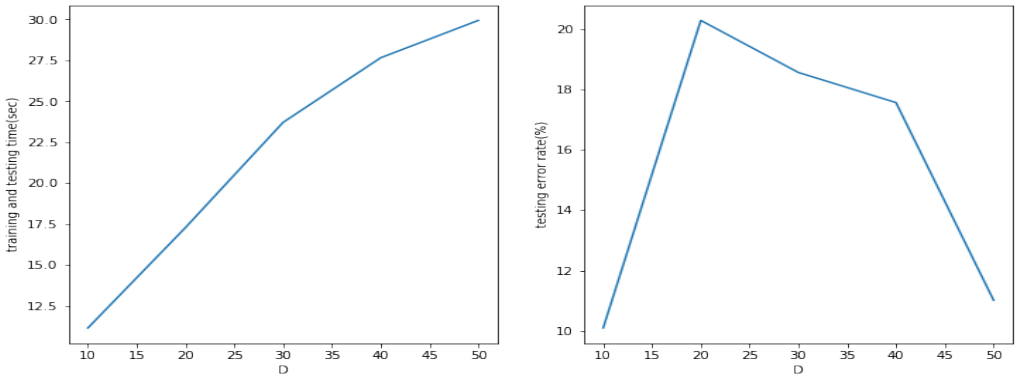

FIGURE 3 – The effect of D on time and error rate in KDDCUP Data Set

We observe that the larger D is, the more computation time is needed. For Condition Based Maintenance of Naval Propulsion Plants Data Set, larger D brings less test error. While for Adult dataset, the variation of D doesn't change test error. This may be caused by very sparse features. For KDDCUP99 dataset, at first test error goes up with larger D, but from $D = 20$, it decreases as D is larger.

4.2 Variation of test errors as P changes for random binning features

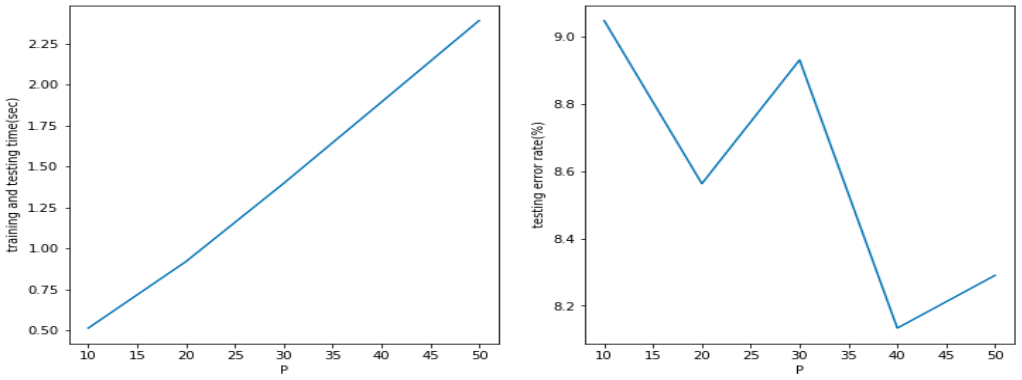

FIGURE 4 – The effect of P on time and error rate in Condition Based Maintenance of Naval Propulsion Plants Data Set

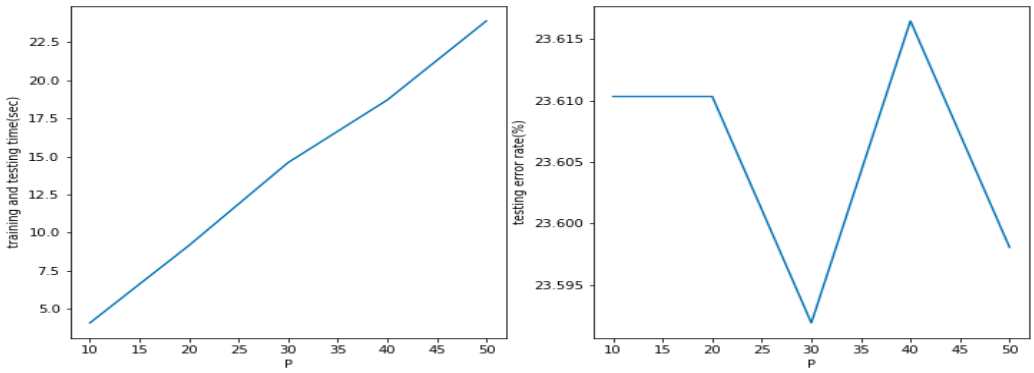

FIGURE 5 – The effect of P on time and error rate in Adult Data Set

As we cannot achieve the results of KDDCUP99 for the computation capacity problem, we only have plots for random binning features on two datasets. Similarly, the larger D is, the more computation time is needed for random binning features. When P increases, the test error of regression fluctuates to decreases. While for adult dataset, it fluctuates without obvious decreasing tends. But the fluctuation is very small.

5. Conclusion

According to the several parts we've written, the conclusion of this implementation is as follows :

5.1 Critical view of the method

Pros :

1. A significant computational speed-up over kernel machines can be achieved by first computing random features and then applying the associated linear technique.
2. For the regression dataset we've chosen to implement, the error rate of the proposed method is obviously lower compared to SVM.

Cons :

1. The proposed method cannot generalize in all kinds of datasets.

5.2 Critical view of the Library used on the task

Pros :

1. **sklearn.kernel_approximation.RBFSampler** realizes well the expected performance of the method proposed in this paper, getting higher accuracy with less time.

Cons :

1. We cannot change the kernel because this library has the deterministic one.