M2 Data Science

Structured Data: Learning, Prediction, Dependency, Testing

Random Features for Large-Scale Kernel Machines

April 3, 2018

Abstract

A. Rahimi and B. Recht proposed in 2007 a very innovative approach for accelerating the training of kernel machines: they map the input data to a randomized low dimensional feature space so that large scale dataset can be solved using linear methods. To do so, they approximate the kernel function using Random Fourier Features or Random Binning Features. The aim of this report is to assess the empirical efficiency of such an approach.

Léa BRESSON
Eya KALBOUSSI
Benoît ROBAGLIA

Lecturer: Zoltan Szabo

Table of contents

1 Introduction

Kernel machines, such as Support Vector Machines (SVMs), are very efficient for handling dataset with a large number of features. Unfortunately, these methods are not appropriate when the number of instances is very high (explosion of the computational cost).

1.1 A quick theoretical background

The paper *"Random Features for Large-Scale Kernel Machines"* (2007) by A. Rahimi and B. Recht [1] is the first to propose an efficient method for accelerating the training of kernel machines with large dataset.

The idea behind is to embed the data into a finite dimensional space using a randomized feature map $z : \mathbb{R}^d \to \mathbb{R}^D$ so that the inner product between a pair of transformed points approximate the kernel:

$$k(x, y) \approx z(x)'z(y).$$

The approximation is good because Bochner's theorem assures that the Fourier transform of the kernel k is a probability density function. A simple linear algorithm is then applied on the transformed feature space. Thus, this approach combines both accuracy of kernel methods and speed of linear methods.

The algorithm proposed by A. Rahimi and B. Rech is the following :

- Compute the Fourier transform p of the shift-invariant kernel k.
- Draw D iid samples $\omega_1, ..., \omega_D \in \mathbb{R}^D$ from the probability distribution p and D iid samples $b_1, ..., b_D \in \mathbb{R}$ uniformly on $[0, 2\pi]$.
- Compute $z : \mathbb{R}^d \to \mathbb{R}^D$ such that $z(x) = \sqrt{\frac{2}{D}}[\cos(\omega_1'x + b_1), ..., \cos(\omega_D'x + b_D)]$.

The aim of this report is to assess the empirical efficiency of such an approach. Please refer to our second Paper Analysis for more theoretical details.

1.2 Task to be solved

The paper [1] proposes two random feature sets to approximate kernel matrices: Random Fourier Features and Random Binning Features. In this work, we focus on Random Fourier Features. More specifically, we are interested in the **Radial Basis Function kernel** (Gaussian) that is a shift invariant kernel:

$$k(z) = e^{-\gamma||z||_2^2} \to p(w) = \frac{1}{\sqrt{4\pi\lambda}} e^{-\frac{w^2}{4\lambda}}$$

with p the Fourier transform. The feature map of the Radial Basis Function (RBF) kernel is approximated by Monte Carlo approximation of its Fourier transform.

In this work, we compare the performance (in terms of both timings and accuracy) of *i)* a linear SVM using the approximate mappings and *ii)* a kernelized (Gaussian) SVM. We perform simulations on two datasets: the *USPS* dataset (9298 instances, 256 attributes) and the *Gisette* dataset (7000 instances, 5000 attributes).

The present report is organized as follows. In **Section 2** we will present the dataset and models used. The implementation details (library, code etc.) will be introduced in **Section 3**. Results will be show in **Section 4**. Finally, we will be discussing the method introduced by A. Rahimi and B. Recht in **Section 5**.

2 Dataset and model selection

To test the Random Fourier Features method, we chose image datasets of hand recognition digits in order to have high dimensional feature space. The first one is U.S. Postal Service dataset which contains 256 features. The second one is the Gisette data base which has 5000 features.

2.1 USPS dataset

The USPS data set refers to numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized.

The data base contains 9298 samples and 256 features (to represent the 16 x 16 image's pixels), a sample standing for a 8-bit image of a hand-written digit between 0 and 9 (cf. appendix - Figure 6).

The machine learning task is to recognize the digit represented in the picture. Therefore we have to solve a 10-class classification problem where each class is a number between 0 and 9. As this is a toy dataset of Scikit-learn, it is quite clean and not much preprocessing was necessary (we only normalized the data as mentioned before).

We then randomly splitted our data into a train and a test set respectively containing 80% and 20% of the original data.

2.2 Gisette dataset

Gisette is a handwritten digit recognition problem inspired by the MINST dataset. The problem is to separate the highly confusable digits '4' and '9'. This is a binary classification problem. The images are size-normalized in a 28 x 28 window (cf. appendix - Figure 7). The data base contains 5000 features, 6000 samples in the training/testing set and 1000 samples

in the validation set.

The dataset was constructed as follow :

- A random subset of the "four" and "nine" patterns is selected from the training and test sets of the MNIST dataset.
- The data is normalized so that the pixel values are in $[0, 1]$.
- In addition to the original variables (normalized pixels), the researchers added a randomly selected subset of products of pairs of variables. These pairs use pixels drawn randomly in a region of the image where the "four"/"nine" separation is more likely to fall in. 2500 features were creating with this procedure.
- Another 2500 pairs were used to construct "probes".

Eventually, we did a simple preprocessing of normalization to have the data centered and reduced.

2.3 Model selection

We then implement a linear SVM using an approximate mapping for the RBF kernel. The baseline for our model is the exact SVM with RBF kernel.

USPS dataset:
As it is a toy dataset, we used the default parameters for each model and didn't do any exhaustive grid search. We fixed the γ parameter for our feature map to $1/200$ and the regularization parameter of the SVM to 5.

Gisette dataset:
Concerning the linear SVM, the regularization parameter, C, is chosen with a 5-fold cross validation. The only free parameter we need to optimize for the RFF feature map is γ. We proceeded also by 5-fold cross validation to find the optimal gamma. For the exact SVM with RBF kernel, the regularization parameter is set to C = 6 thanks to 5-fold cross validation). It is also set to C = 6 for the RFF-SVM with the same arguments.

3 Implementation overview

3.1 RBF kernel

In order to construct an approximate mapping for the RBF kernel, we implement a Python Class based on the algorithm 1 of the paper "*Random Features for Large-Scale Kernel Machines*" by A. Rahimi and B. Recht. We use the code (with minor modifications) of the algorithm *RBFSampler* from Scikit-learn (module *sklearn.kernel_approximation*[1]). We de-

[1]The source code is available at the following link: `https://github.com/scikit-learn/scikit-learn/blob/a24c8b464d094d2c468a16ea9f8bf8d42d949f84/sklearn/kernel_approximation.py`

cide to implement our own Python Class for a higher readability and a better understanding. Moreover, we add the method *Compute_kernel* (described below) which was not implemented in Scikit-learn.

The Class, which relies on a Monte Carlo approximation to the kernel values, takes as input arguments:

- Gamma (float): parameter of RBF Kernel,
- N_components (int): number of Monte Carlo samples per original feature (target dimensionality of computed feature space).

In addition the Class has 3 methods:

- The *Fit method*: generates the Monte Carlo random samples,
- The *Transform method*: computes the new features, i.e., transformation of X (n_samples, n_features) to $Z(X)$ (n_samples, n_components),
- The *Compute_kernel method*: computes the approximated kernel matrix (additional code we had to write), i.e. the dot product between Z and Z^T.

This feature space is then used for the task of classification with an SVM.

3.2 Structured prediction task

The Scikit-learn package provides enough tools to implement an SVM: it wraps both *Liblinear* and *Libsvm* libraries. The performance is even greater since the wrapper is fined-tuned to minimize the memory allocations (faster and less memory usage on large datasets[2]). Moreover, Scikit-learn has many additional utilities that we will use such as PCA transformers, performance evaluation, GridSearchCV (exhaustive search over specified parameter values for an estimator), etc.

In particular, we use the *Sklearn.Svm* module that includes Support Vector Machine algorithms. We focus on the Classes **SVC** (using Libsvm) and **LinearSVC** (using Liblinear). Both take as input an array X of size [n_samples, n_features] holding the training samples, and an array y of class labels. After being fitted, the model can make predictions.

Different kernels can be specified at initialization and it is possible to specify the kernel used in the algorithm. In our case, we use *i)* the RBF feature map given by our personalized Class described above with the LinearSVC (linear kernel). We compare the results to *ii)* the SVC class with RBF kernel.

[2]See the following link for more detailed information `http://fa.bianp.net/talks/fosdem-skl/`.

4 Results

4.1 Checking the computation of the kernel

As said above, our personalized Python Class computes a low-dimensional kernel that could approximates the real one with lower computational and storage time.

In order to make sure that the kernel computed by the *Compute_kernel* method does verify:

$$K^d(x, y) \approx K^D(x, y) \text{ , where } d > D$$

We ran a test where we compute the cosine similarity between the true kernel and the approximation for a random dataset containing 200 features. When changing the size D (parameter *n_components* of the Python Class) of the low dimensional space, we obtain the following graph:

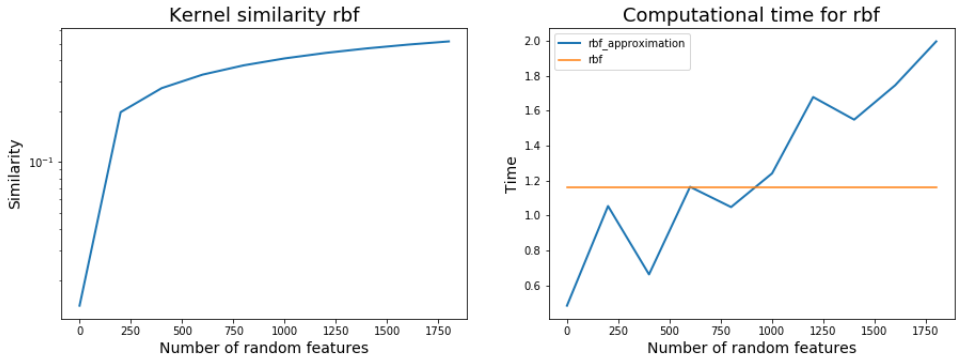

Figure 1: Evaluation of the approximated feature map

Both graphs show that the similarity between the two kernels grows as the size of the low dimensional space grows, to reach almost 1 when D is very large. On the other hand, the computational time is not always better when using the kernel approximation method. When D exceeds $\approx$ 750, using the real kernel is faster than computing its approximation. We have to find the right D in order to have a faster algorithm that approximates quite well the kernel, depending on the task needed.

4.2 USPS dataset

In order to test the efficiency of kernels in classification tasks, we start by running our algorithm on the USPS algorithm, which contains 256 features.

We compute the approximated kernel for several values of D (size of the low dimensional space), and pick the best D such that we obtain the lowest computational time possible

corresponding to the highest score. When plotting results, we obtain the following graphs.

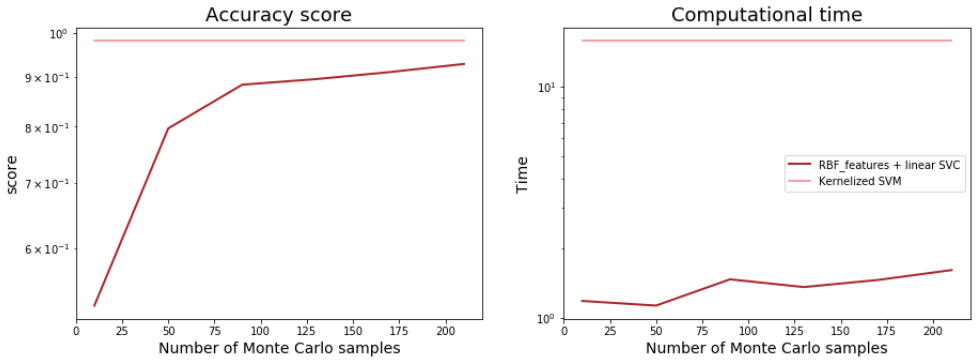

Figure 2: Performances of the Exact SVM and the RFF + linear SVM - USPS dataset

We can see that no matter the size of the low dimensional space chosen such that D<210, the computational time of the approximated kernel + a linear method remains smaller than the computational time of a SVM. Thus, we can choose D based on the score.

The best score obtained with the RBF kernel + linear classifier (D=210) is **0.92** reached in **1.61 s** whereas the score obtained with the SVM algorithm is **0.98** reached in **15.7s**. The results out of both algorithms are quite comparable in terms of accuracy. However, the approximation kernel method is much faster.

Here is the normalized confusion matrix for the USPS dataset:

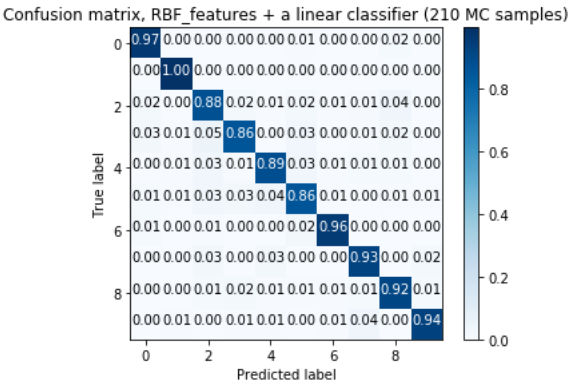

Figure 3: Confusion Matrix - RFF + linear SVM

4.3 Gisette dataset

In order to prove further the efficiency of the kernel approximation trick, we choose the Gisette dataset, which, unlike the USPS, contains 5000 features. As done before, we choose

several values of the size D of the low-dimensional space varying between 10 and 3000, and choose the one value that fits best our task. When plotting results, we obtain the following graphs:

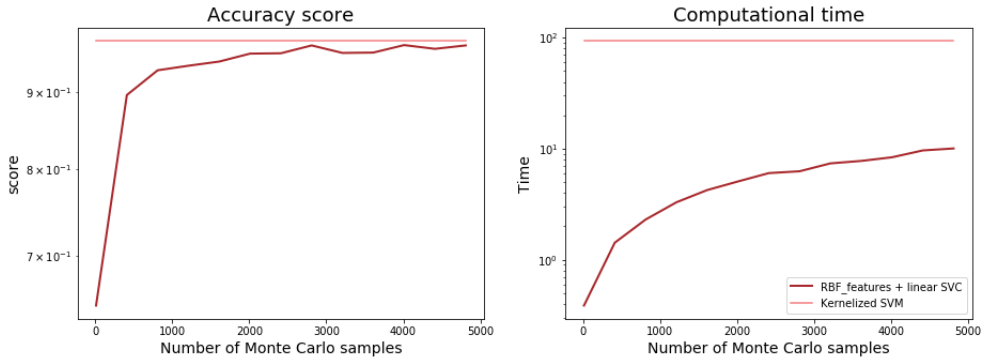

Figure 4: Performances of the Exact SVM and the RFF + linear SVM - Gisette dataset

As seen earlier with the USPS dataset, the computational time of the RFF + linear classifier is smaller than the non-linear classifier one for all chosen sizes of the low-dimensional space as long as it remains smaller than the size of the input feature space. the best score obtained with the RBF kernel + linear classifier is **0.96** reached in **10.1 s** whereas the score obtained with the SVM algorithm is **0.97** reached in **93.4 s**. The results out of both algorithm are quite comparable in terms of accuracy, even though the exact SVM classifier remains better than the method using kernel approximation.

Here is the normalized confusion matrix for the Gisette data set:

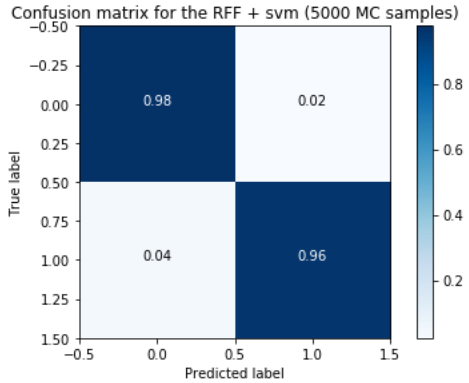

Figure 5: Confusion Matrix - RFF + linear SVM

5 Conclusion

The Random Fourier Features method appears to be much more efficient when handling very large data sets. As presented above, using a RFF with a linear method costs much less in terms of computational time and storage. Thus, the combination of kernel map approximations with a linear classifier can perform better than a non linear classifier (such as SVM), depending on the task to be solved.

Although this method seems to outperform non linear kernel based methods, one still needs to find the right γ and D in order to obtain acceptable results. Unlike non linear kernel based methods, we noticed that the combination of a RFF space and a linear method is much more sensitive to the choice of γ than a standard SVM in terms of accuracy. No matter how good D is, when γ is too large, the approximation error can be very large. In order to overcome this problem, we can use a Memory Efficient Kernel Approximation (MEKA) framework to approximate the kernel matrix as proposed in *"Memory Efficient Kernel Approximation"* by Si Si, Cho-Jui Hsieh and Inderjit S. Dhillon [2].

The choice of the size of the the low-dimensional space is also very important, since choosing D too small can have extremely large approximation error and choosing D too large may be too heavy in terms of computational time and storage. Choosing the right parameters is crucial for the method to work properly.

The code implemented in Scikit-learn reproduces well the theoretical baseline introduced by the paper. However, its fit method is independent of the data set chosen as well as the task to be done.

References

[1] Ali Rahimi and Ben Recht. "Random features for large-scale kernel machines". In: *In Neural Information Processing Systems* (2007).

[2] Inderjit S. Dhillon Si Si Cho-Jui Hsieh. "Memory Efficient Kernel Approximation". In: (2017).

Appendix

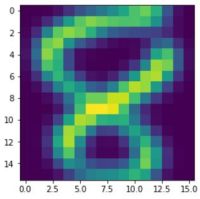

Figure 6: USPS Dataset

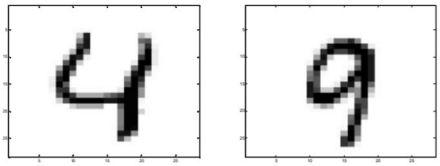

Figure 7: Gisette Dataset