# Data Mining: Learning from Large Data Sets - Fall Semester 2015

## Large Scale Image Classification

BY SUYIFAN,ZJIA,WEIXI@STUDENT.ETHZ.CH

November 8, 2015

In this project we built a distributed classifier for image classification. Using the feature vector of images, we applied the random Fourier features to apply an approximated version of Gaussian kernel in the context of stochastic gradient descent.

## 1 Stochastic Gradient Descent

We implemented the standard stochastic gradient descent ourselves, the update rule for weight vector $w$ is:

$$\begin{aligned} w_t' &= w_t + \eta_t \cdot yx \cdot \mathbb{1}_{(yw^Tx<1)} \\ w_{t+1} &= w_t' \cdot \min\left(1, \frac{1/\sqrt{\lambda}}{\|w_t'\|}\right) \end{aligned}$$

Where the step size $\eta_t = 1/\sqrt{t}$, and $\lambda$ is a regularization factor to tune.

Instead of implementing the stochastic gradient descent, we have also tried the `SGDClassifier` in scikit-learn. There is a parameter `alpha` to tune. In our experiments we found that the scikit version of SGD is slightly better than our implementation.

## 2 Feature Engineering

To make the sgd classifier perform better we tried to add more features to the origional vector. After normalizing the origional vector, we added some statics like the mean, median, std, etc. In our experient this can imporve a lot the performance.

## 3 Random Fourier Feature

In the lecture we have learn the generation of random Fourier features (*rff*) as an inverse kernel trick, in this way we can train an (approximately) equivalent svm with Gaussian kernel.

The transformation of the origional vector consist of sampling at random (according to Gaussian distribution) $m$ weight vectors and $m$ random translations, and compute the cosine of this weighted and translated vector.

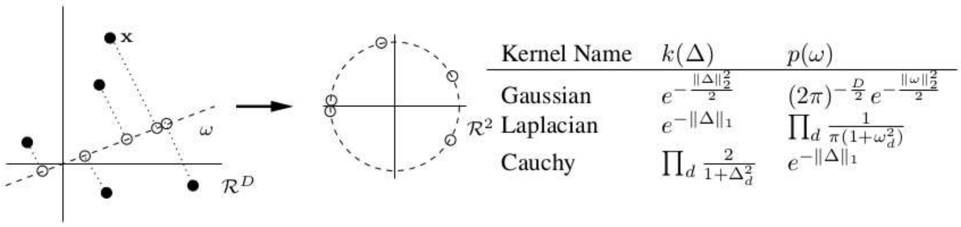
The process is as follows:

**Algorithm 1**

```
1. draw $w_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbb{R}^d)$, $b_i \overset{\text{iid}}{\sim} \text{Unif}(0, 2\pi)$, $i = 1...m$ ;
2. for each $x \in \mathbb{R}^d$, compute: $z = \sqrt{\frac{2}{m}}\big[\cos\big(w_i^T x \cdot \gamma + b_i\big)\big]_{i=1...m} \in \mathbb{R}^m$
3. use $z$ as feature vector to feed to the sgd classifier.
```

In the above algorithm, the $w_i$ are drawn from a normal distribution in dimension d, $\gamma$ is a parameter that controls the spread of the gaussian kernel.

Apart from the Gaussian distribution, we can choose other distributions as the course slide indicates. We have tried Laplace and Cauchy distribution, and with Cauchy distribution (which means we are using a Laplacien kernel), we get the best performance.



| Kernel Name | $k(\Delta)$ | $p(\omega)$ |
|---|---|---|
| Gaussian | $e^{-\frac{\lVert\Delta\rVert_2^2}{2}}$ | $(2\pi)^{-\frac{D}{2}} e^{-\frac{\lVert\omega\rVert_2^2}{2}}$ |
| Laplacian | $e^{-\lVert\Delta\rVert_1}$ | $\prod_d \frac{1}{\pi(1+\omega_d^2)}$ |
| Cauchy | $\prod_d \frac{2}{1+\Delta_d^2}$ | $e^{-\lVert\Delta\rVert_1}$ |

**Figure 1.** Random Fourier features for different kernels

# 4 Parameter selection

There are several parameters to tune in this project, including: the number of samples($m$), step size of sgd(`alpha`), the parameter for kernel ($\gamma$).

To select the parameter, we split the training set into some small pieces, and we used one piece of 10000 lines as training set and another 10000 lines as testing set.

We will run the algorithm on the training set and test it against the testing set, in this way we can choose the best parameters.

# 5 Team member contribution

- Yifan SU worked for the feature engineering and parameter selection and sgd;
- Zheng JIA worked for the parameter selection and sgd;
- Xing WEI worked for the sgd and random Fourier features.