CSE-481 Optimization Methods
Term Paper
Pegasos for SVM Training

Shyamgopal Karthik
Roll No. 20161221
IIIT Hyderabad

***Abstract*—This document summarizes the implementation of the ”Primal Estimated Sub-gradient Solver for SVM” i.e Pegasos[1]. As the name suggests, it uses the primal formulation of the SVM. We first show how to perform basic binary classification. Further, we extend the implementation to incorportate Mercer kernels and demonstrate their efficacy in non-linear classification. Finally, we also implement multi-class classification to show the performance on Fashion-MNIST[2] and compare it against standard benchmarks.**

***Index Terms*—SVM, Primal, Sub-gradient, Kernels**

I. Introduction

Pegasos uses the primal sub-gradient to iteratively update the weights W of the SVM. Using the sub-gradient makes sense since the most popular SVM loss function i.e hinge loss is not differentiable at all points and therefore resorting to sub-gradient based methods is inevitable.

II. Binary SVM Algorithm

We state the algorithm for training the binary SVM below: The SVM formulation essentially tries to maximize the ”margin” in order to ensure a good classification. This is illustrated in Fig 1

Algorithm 1: Pegasos for Binary SVM

```
Result: Weight vector $w$ after $T$ iterations
Given $X$, $\lambda$,$T$; $t = 0$ while $t < T$ do
    Choose $i_t$ randomly from $i_t \in \{1, ...|S|\}$ ;
    $\eta \leftarrow \frac{1}{\lambda_t}$ ;
    if $y_{i_t}(w_t \cdot x_{i_t} < 1$ then
        $w_{t+1} \leftarrow (1 - \eta_t\lambda)w_t + \eta_t y_{i_t} X_{i_t}$ ;
    else
        $w_{t+1} \leftarrow (1 - \eta_t\lambda)w_t$ ;
    end
end
```

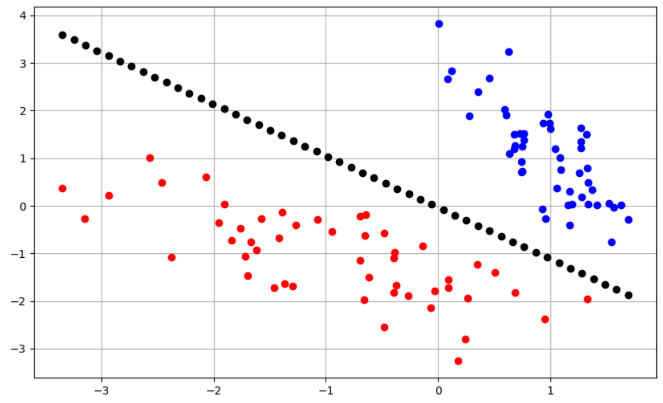

Fig. 1. Among all possible linear seperating boundaries, the SVM optimization function picks the one that maximizes the margin. This can be seen here

III. Kernelized SVM

The basic idea behind kernelization in the context of SVMs is to project a sample x to different(higher dimensional) space. This can be done using a mapping function $\phi(x)$. However, $\phi(x)$ could even be infinite dimensions in the case of the rbf kernel. Therefore, we can't store the kernel explicitly. We get around this by using the kernel trick. Given n samples, we define $K(i, j) = \phi(x_i)^T \phi(x_j)$. Evaluating this kernel matrix is easy and can be done beforehand. Also, the SVM formulation has dot products between x_i and x_j everywhere, which are now just replaced with $K(i, j)$. The only condition on K is that it should be positive semi-definite. This comes from Mercer's theorem and therefore are called Mercer kernels. The algorithm for training a kernelized SVM is done using the dual variabes and is given below: Once we have α, we compute w

Algorithm 2: Pegasos for Kernelized Binary SVM

```
Result: Dual Variables $\alpha$ after $T$ iterations
Given $X$,$\lambda$ $\alpha_i = 0$; $t = 0$ while $t < T$ do
    Choose $i_t$ randomly from $i_t \in \{1, ...|S|\}$ ;
    if $y_{i_t} \frac{1}{(\lambda t)} \sum_j \alpha_j^t y_j K(i, j) < 1$ then
        $\alpha_i^{t+1} \leftarrow \alpha_i^t + 1$ ;
    else
        $\alpha_i^{t+1} \leftarrow \alpha_i^t$ ;
    end
end
```

as

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

The benefits of using a kernelized SVM is shown in Fig 2 and in Fig. 3. The linear SVM which is capable of only linearly seperating the data fails to classify accurately while the rbf kernel based SVM is capable of accurately classifying the data.

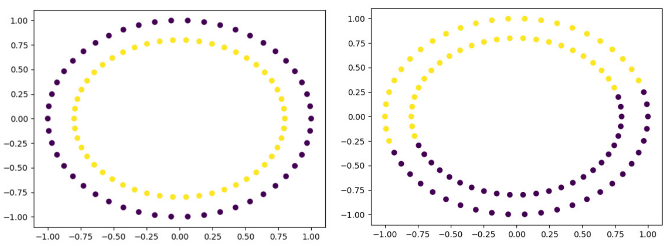

Fig. 2. Result of the rbf kernel SVM on the left and the linear SVM on the right

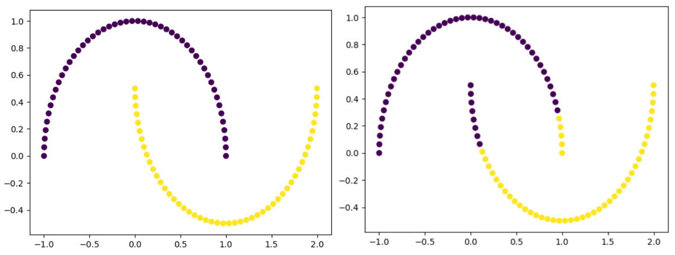

Fig. 3. Result of the rbf kernel on the left while the linear kernel is shown on the right.

IV. Multi-class Classification

To extend the SVM formulation to multi-class classification, we use the one-vs-one approach. To solve a k class classification problem, we train ${}^{k}C_2$ pairwise classifiers. When we need to

V. Results on Fashion-MNIST

The Fasion MNIST[2] dataset contains 10 classes and is similar to the original MNIST dataset, except it is harder thereby giving a more challenging test to classifiers. We use the one-vs-one strategy to benchmark our SVM on the dataset and compare it to the baseline linear SVM provided by scikit-learn. We could not use the kernelized SVM as training it on 40,000 images was too time consuming.
Our results are given below: Our confusion matrix is shown in Fig 4

Method	Accuracy
Pegasos Linear SVM	80.29
Scikit-learn Linear SVM	72.9

As we can see, our Pegasos based SVM benefits from the fact that large number of iterations can be run and therefore we

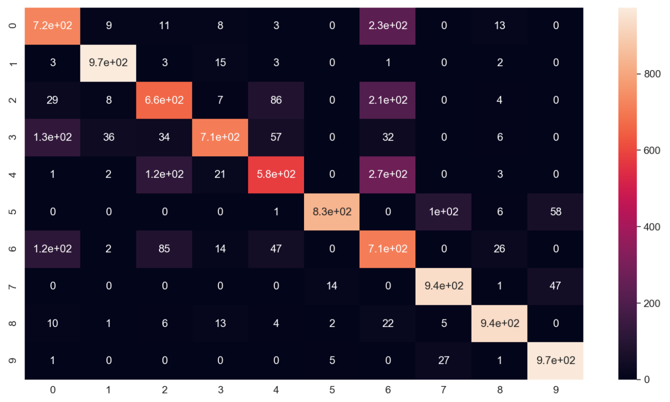

Fig. 4. Confusion matrix for the fashion MNIST dataset

are able to perform better than the baseline scikit-learn Linear SVM.

References

[1] Shai Shalev-Shwartz et al. “Pegasos: Primal estimated sub-gradient solver for svm”. In: *Mathematical programming* 127.1 (2011), pp. 3–30.

[2] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].