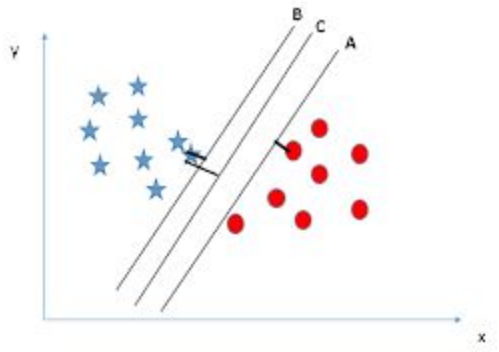
Report for Pegasos Project

Pegasos: primal estimated sub-gradient solver for SVM is a method to solve optimization problem of SVM.

Optimization Problem of SVM:
As shown in the adjacent figure, SVM optimization problem is to draw a line/hyperplane such that distance between the line and the nearest positive/negative point is maximized.



Consider,the dataset given as $S = \{(x_i, y_i)\}_{i=1 \text{ to } m}$, where $x_i \in R^n$ and $y_i \in \{+1, -1\}$. Optimization problem is given as shown in the figure below:

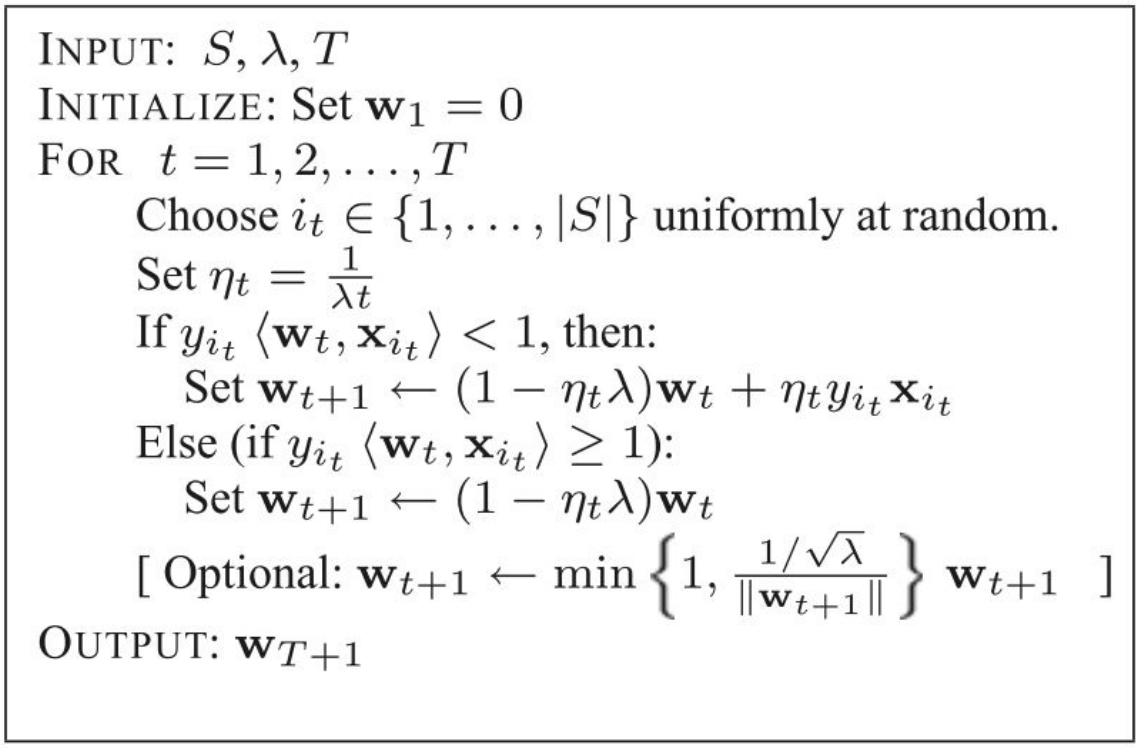
$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m} \sum_{(\mathbf{x},y)\in S} \ell(\mathbf{w}; (\mathbf{x}, y)),$$

where

$$\ell(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - y \langle \mathbf{w}, \mathbf{x} \rangle\},$$

Objective of this project is to implement Pegasos algorithm.

Non-kernelized version of Pegasos algorithm is as shown in the figure below:



During the implementation of this algorithm, first we need dataset S. Fashion-MNIST dataset is used for this implementation(As suggested in the project description). [https://www.kaggle.com/zalando-research/fashionmnist] This link provided the dataset in csv file format. Dataset is about multiclass classification of different fashion objects such as shoes, dress etc. Since this project is based on binary classification, I have represented and interested in only two classes namely, Tshirt/Top as -1 and Trousers as +1. Dataset parsing and suitable formatting of csv file is done by format_data function in the code.

Other parameters such as lambda and T are set as per the required learning rate. Learning rate is decided in such a way that it should not be too high such that the iterations oscillate or it should not be too low such that it takes too many iterations to converge. After every iteration learning rate decreases. For each iteration in the algorithm, we choose a uniformly random sample set(x,y). Using the numpy library in python, we calculate dot product of w and x which is then multiplied with corresponding y-value. This is nothing but indicator function. As shown in the figure below, when we consider sub-gradient of the objective if the value of indicator function is below 1 then there is a non-zero loss as shown in the algorithm figure.

$$f(\mathbf{w}; i_t) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \ell(\mathbf{w}; (\mathbf{x}_{i_t}, y_{i_t})).$$

We consider the sub-gradient of the above approximate objective, given by:

$$\nabla_t = \lambda \mathbf{w}_t - \mathbb{1}\left[y_{i_t} \left\langle \mathbf{w}_t, \mathbf{x}_{i_t} \right\rangle < 1\right] y_{i_t} \mathbf{x}_{i_t},$$

Pegasos is based on stochastic gradient descent algorithm. General outline of stochastic gradient descent algorithm is shown in the figure below.[https://svn.spraakdata.gu.se/repos/richard/pub/ml2016_web/a2_clarification.pdf]

Algorithm 2 Stochastic gradient descent with a fixed number of steps.

Inputs: a list of example feature vectors $\boldsymbol{X}$
a list of corresponding outputs Y
the number of steps T
$\boldsymbol{w} = (0,\ldots,0)$
for t in $[1,\ldots,T]$
select randomly a training instance $\boldsymbol{x}_i$, with a corresponding output label y_i
determine a step length η
compute the gradient $\nabla(f)$ of the objective function $f(\boldsymbol{w},\boldsymbol{x}_i,y_i)$
$\boldsymbol{w} = \boldsymbol{w} - \eta \cdot \nabla(f)$
the end result is $\boldsymbol{w}$

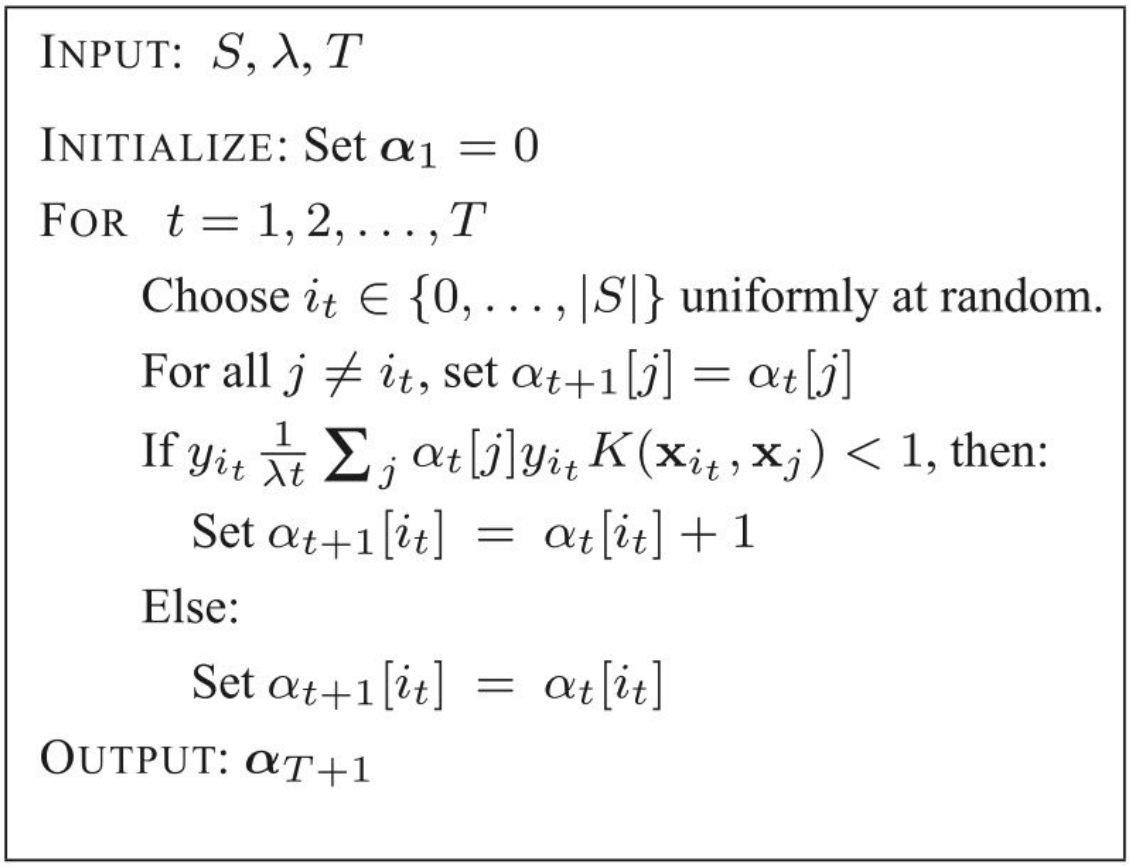
This is fundamental flow of Pegasos algorithm. Gradient $\nabla(f) = \lambda \cdot w + \nabla(\text{Loss})$ where Loss is hinge loss. Throughout, the implementation of this algorithm, we have used hinge loss. There exist other loss functions as well which are useful in other different scenarios such as logarithmic loss function. Hinge loss function gives us tightest convex upper bound on 0-1 loss. Hinge loss is represented as Max{0, 1 − y <w, x>}. Hence,

$\nabla(\text{Loss}) = -yi \cdot xi$ if $yi \cdot (w \cdot xi) < 1$
(0,. . ., 0) otherwise

[For further detailed information regarding loss functions and there subgradient please refer to the original paper of Pegasos].

The above algorithm was non-kernelized version of Pegasos algorithm. Kernelization support to the Pegasos algorithm allows us to execute Pegasos algorithm using kernels and without having direct access to the feature vectors x. This is because of Representer theorem[Kimeldorf, G., Wahba, G.: Some results on tchebycheffian spline functions. J. Math. Anal. Appl. 33, 82–95 (1971)]. These kernels allow us to make inseparable data separable by using mapped version of the actual instances. Few famous kernels are Gaussian kernel, polynomial kernel. Gaussian kernel is represented as follows : $e^{(-\text{Gamma}*||x-y||_2^2)}$. Polynomial kernel is represented as $(<x.y> + c)^d$ where d is degree of polynomial. If c is zero the polynomial kernel becomes homogeneous which has been successfully implemented in this

project. The kernelized pegasos algorithm is shown in the figure below:



This algorithm works in similar way where instead of W we have α (α value for a data point represents how many times it has been selected). It again uses modified version of indicator function which also involves Kernel(polynomial kernel in this implementation) to count the number of times a datapoint is selected. Once α_{T+1} is obtained, we can use that value to test on different dataset as follows: y' = sign (Sum $\alpha_i y_i x^T{}_i$ x) [MIT Lecture Notes https://people.csail.mit.edu/dsontag/courses/ml16/slides/lecture6_notes.pdf]. $x^T{}_i$ x can now be replaced with corresponding mapping as well. Here for simplicity we has used kernel on the given values.

SEE THE VIDEO LINK FOR IMPLEMENTATION:
[https://drive.google.com/file/d/1c4FLE9bZW13loGj1ish4M9xOMn4vK2iI/view?usp=sharing] 47MB [Contact on achintya.desai@research.iiit.ac.in if link is not opening]

For Non-kernelized Pegasos :

Training sample count	10	100	1000	1000
Time in seconds	0.002046	0.01922	0.18999	2.002760

For Kernelized Pegasos:

Training sample count	10	100	500
Time in seconds	8.2194	87.4864	446.5860