Assignment 4

CMSC 678 — Introduction to Machine Learning

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

Question 1---

1.a

P(x_1 , x_2, …., x_n)=P(x_1) * P(x_2) * ……. * P(x_n)   , independent

= $\prod_{i=1}^{n} \boldsymbol{P}(x_i)$

Log-likelihood,

L = log($\prod_{i=1}^{n} \boldsymbol{P}(x_i)$)

= log(P(x_1) * P(x_2) *……. * P(x_n))

= log P(x_1) + log P(x_2) +……. +log P(x_n)

= $\sum_{i=1}^{n} \boldsymbol{log\ P}(x_i)$

= $\sum_{i=1}^{n} \boldsymbol{log\ \sum_{k=1}^{K} \pi_k N(x_i; \mu_k, \sigma^2 I_D)}$

1.b

The marginal likelihood can be computed as, P(x_i) = $\sum_{k=1}^{K} \boldsymbol{\pi_k N(x_i; \mu_k, \sigma^2 I_D)}$.
That is probability of x_i across all the clusters k
where π_k is mixing distribution $\sum_k \pi_k$=1 and I_D is DxD Identity matrix

1.c

Given posterior probability of cluster k for a point x_i as $\gamma_{i,k} = \frac{\pi_k N(x_i;\mu_k,\sigma^2 I_D)}{P(x_i)}$

To derive the gradients we consider the E-M Step which is expectation maximization.
For the π_k, we need to consider this term, we can include this term with the help of Lagrange Multipliers, $\sum_k \pi_k$=1.

$\lambda(\sum_k \pi_k - 1)=0$

For μ_k the above lagrange multiplier will become 0, because of the constant term

$max_{\mu_k} E_{xi\sim\gamma_{i,k}}$ log(P(x_i))

L=$\sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{i,k}$ * log(P(x_i))

$\frac{\partial L}{\partial(\mu_k)} = \sum_{i=1}^{n} \sum_{k=1}^{K}. \frac{\partial(\gamma_{i,k} * \log(P(x_i)))}{\partial(\mu_k)}$

$\sum_{i=1}^{n} \sum_{k=1}^{K}. \frac{\partial(\gamma_{i,k} * \log(P(x_i)))}{\partial(\mu_k)} = 0$

Now considering for π_k ,

$max_{\pi_k} E_{xi\sim\gamma_{i,k}}$ log(P(x_i)) + $\lambda(\sum_k \pi_k - 1)$

L=$\sum_{i=1}^{n} \sum_{k=1}^{K}(\gamma_{i,k}$ * log(P(x_i))) + $\lambda(\sum_k \pi_k - 1)$

$\frac{\partial L}{\partial(\pi_k)} = \sum_{i=1}^{n} \sum_{k=1}^{K}. \frac{\partial(\gamma_{i,k} * \log(P(x_i)))}{\partial(\pi_k)} + \frac{\partial(\lambda(\sum_k \pi_k - 1))}{\partial(\pi_k)}$

$$\sum_{i=1}^{n} \sum_{k=1}^{K} \cdot \frac{\partial(\gamma_{i,k} * \log(P(x_i)))}{\partial(\pi_k)} + \frac{\partial(\lambda(\sum_k \pi_k - 1))}{\partial(\pi_k)} = 0$$

1.d

The gradients derived above for Eq1 w.r.t μ_k and π_k are used to optimize the Eq1 by updating the parameters μ_k and π_k from the gradients in the Eq1 to maximize the log-likelihood of the Eq1. We calculate the gradients using E-M steps to maximize log-likelihood. The gradients are calculated and they are solved for 0. After getting the values, the values are updated in the Eq1 and the parameter values are again calculated until convergence.

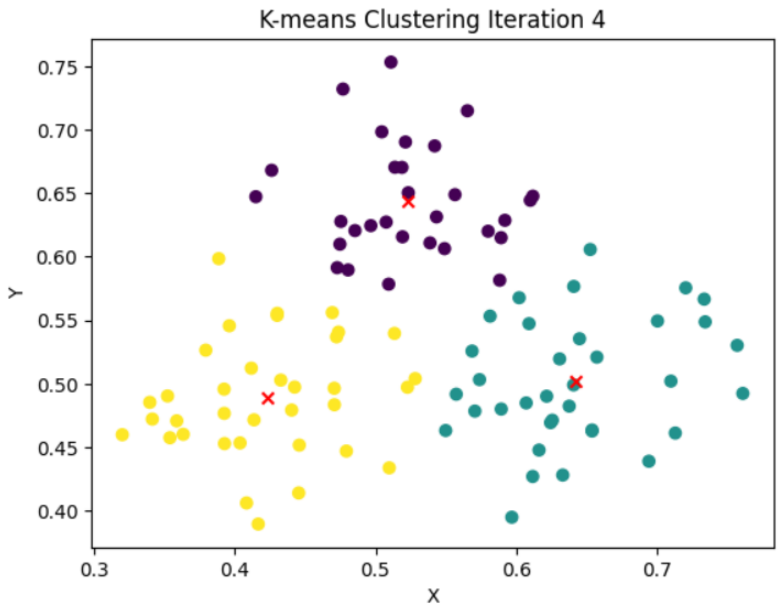
1.e

As the σ -> 0, the Gaussian Mixture model becomes more peaky and have data distributed and centered around the mean. The data will be surrounded by the mean and the distribution becomes peaky. The K-Means also have the data surrounded to their respective means. So when σ -> 0, The Gaussian Mixture Model is similar to the K-Means.

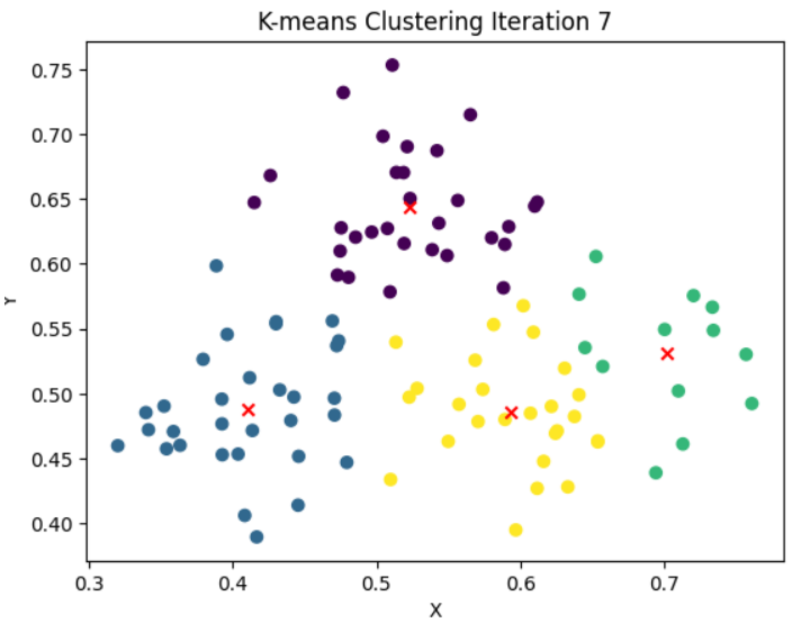
Question 2--

2.

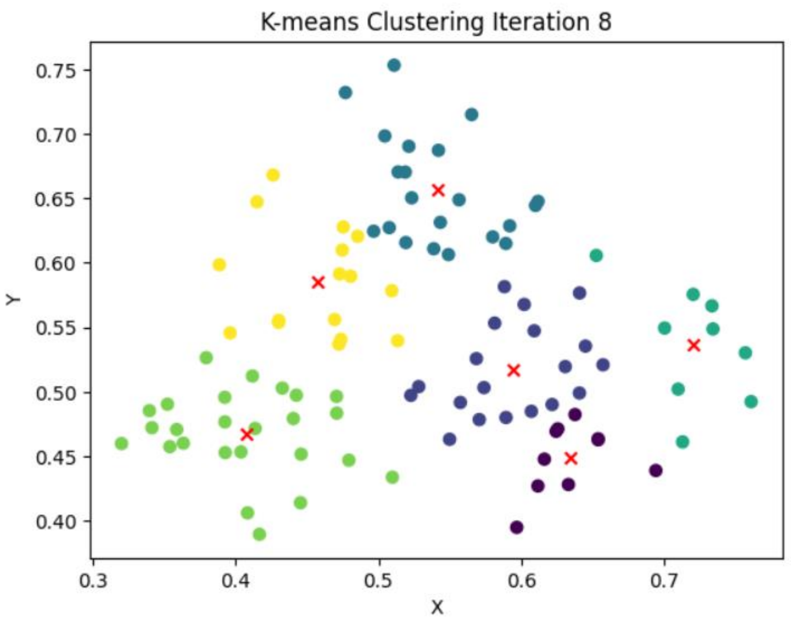
Implemented K-Means clustering algorithm according to the Llyods algorithm where we first initialize random cluster points as mean, next considering all the points which has minimum distance to the clusters we recalculate the mean. We run the algorithm till the previous mean is not changed. If the initial clusters are not chosen right then the convergence criteria might take a longer time, so I have taken convergence steps to iterate over 100 steps if the stopping criteria is met less than the steps it will terminate and if not, program will stop running after 100 iterations.
For the given data set, it will converge easily. But for the more data it will take time.
For the given data considered,
Initial cluster points as random, Number of clusters=3, and plotted according to the objective Eq3.
The algorithm converged in 4 steps. The maximal steps I have observed were 9 steps.



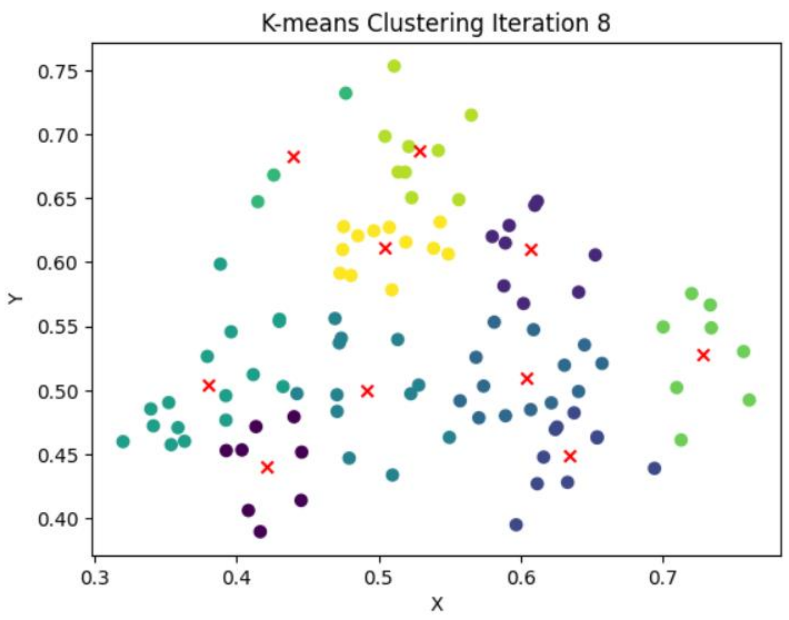
Initial cluster points as random, Number of clusters=4, and plotted according to the objective Eq3. The algorithm converged in 7 steps



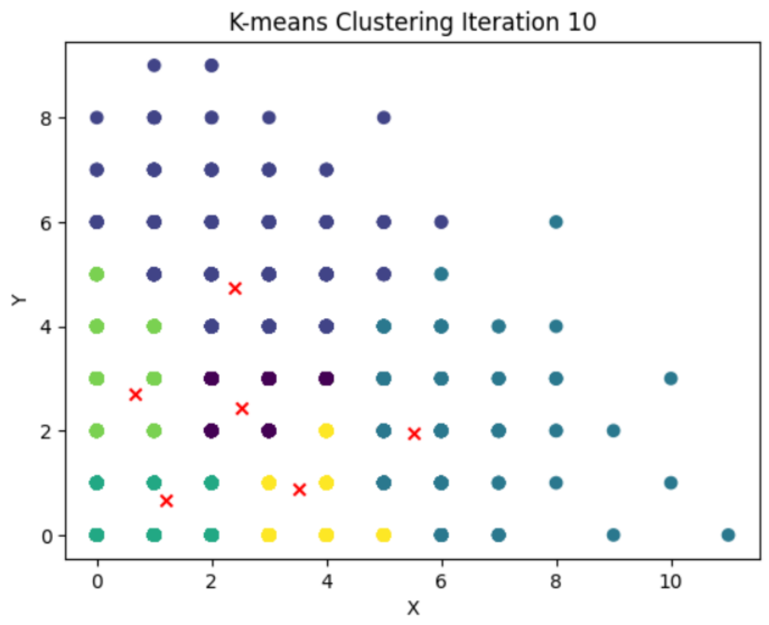
Initial cluster points as random, Number of clusters=6, and plotted according to the objective Eq3. The algorithm converged in 8 steps. For this all the initial cluster points are selected closer to each other, it converged by clustering the points.



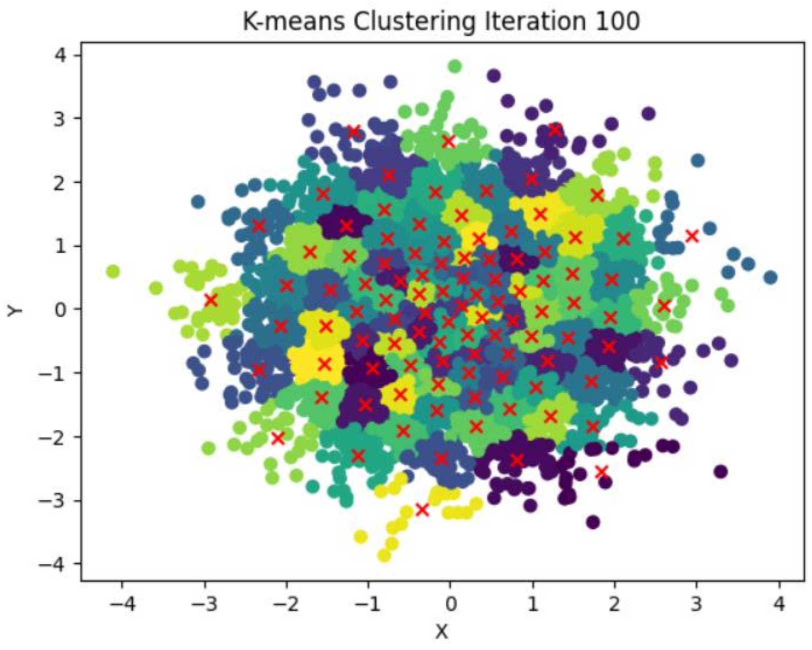
Initial cluster points as random, Number of clusters=10, and plotted according to the objective Eq3. The algorithm converged in 8 steps



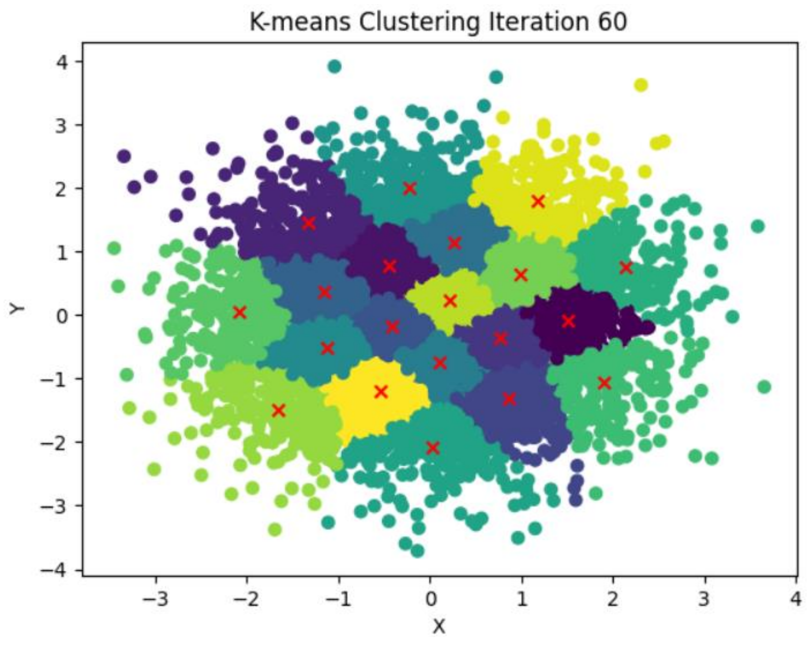
Now generated data using poisons distribution from numpy, Considered 6 clusters for 10000 data, initial cluster points at random, the algorithm didn't converge, so it stopped at 100th step. Ran again by randomly initializing the cluster points with same 6 clusters as before, converged in 10 steps.



Now generated data using Gaussian Mixture distribution from numpy and inbuilt Guassian Model, considered 100 clusters for 10000 data, initial cluster points at random, the algorithm took long time to converge, so stopped at 100th step.



considered 20 clusters for 10000 data, initial cluster points at random, the algorithm took long time to converge but it converged in 60 steps.



The convergence of algorithm in minimum steps depends upon the right initialization of the points as clusters. If its not like that then it may take longer time to complete or some times it may not terminate. In order to terminate we can define the max steps in the algorithm to stop.

Question 3---

3.

(a) The first paragraph of the introduction uses the terms "unconstrained," "empirical loss minimization," and "penalty term for the norm of the classifier." Identify the mathematical instantiations of these terms in equations 1 and 2 (i.e., what parts of equations 1 and 2 correspond to "unconstrained," "empirical loss minimization," and "penalty term?").

A. We have this equation,

$min_w \frac{\lambda}{2} |w|^2 + \frac{1}{m}\sum_{(x,y)\in S} l(w; (x,y))$ --------------- (1)

where $l(w; (x, y)) = \max\{0, 1 - y <w, x>\}$, ---------------(2)

The unconstrained above refers to the learning parameter has no bounds on it. We will not consider any constraint for the learning parameter w for optimization

The Empirical loss minimization term in Eq(1) is $\frac{1}{m}\sum_{(x,y)\in S} l(w; (x,y))$

$\frac{1}{m}\sum_{(x,y)\in S} \max\{0, 1 - y < w, x >\})$

The penalty term in Eq(1) is $\frac{\lambda}{2} |w|^2$

(b) What is the main point of the discussion of the methods on pages 3-4?

A. The main point of the discussion of the methods on pages 3-4 is to study about the recent work done on SVM and to see how these analysed methods connect with the model that the authors will be working. With the analysis of these they draw conclusions for existing analysed methods that effects accuracy by examining large datasets, regularization term, stepsize

(c) Describe, in written English, the mini-batch Pegasos algorithm, including why this algorithm works on "mini-batches" and the purpose of the projection step (eqn 6).

A. The mini-batch Pegasos algorithm is the extension of the basic pegasos algorithm, there it worked on the single instance, here the batches k is considered. The k batches consider all the different criteria of the instances which the earlier basic pegasos algorithm with one single instance can't see. Due to the batches, the convergence will be faster. Therefore, the mini-batch pegasos algorithm works on the mini-batches as itself is the extension of the basic pegasos algorithm with k batches.

The purpose of projection step is to restrict the learning parameter w with the help of regularization parameter λ within the bounds i.e. ball radius $\frac{1}{\sqrt{\lambda}}$. and updating w such that it remain in the bounds.

(d) What is the importance of a kernel operator $K(x, x^l)$?

A. The kernel operators are used because they represent the functions of the instances rather than the original instance feature. The kernel operator modifies the representation of the original feature into some function and uses it without direct access to original feature or explicit access to the weight parameter. It works with non-linear and complex representations which captures good amount of features.

(e) Describe, in written English, the meanings of the formulas wt+1 = . . . on page 12. (If you feel it helpful, you may refer to the algorithm in Figure 3.)

A. The formulas for w_{t+1} are weight updating step for kernelized pegasos algorithm which follows the lemma and theorem. according to the Kernalized Pegasos Algorithm, α counts the each instance considered while it is misclassified point. Initially α_1 starts with 0. In the weight updating for α, we increment α_{t+1} if it is a misclassified point or else no increment. We just include this α terms in the weight updating that we got from theorem1 and we return α_{t+1} for the kernelized pegasos algorithm.

(f) What is the main contribution of section 5?

A. The main contribution of section 5 is to use other loss functions besides hinge loss. In order to apply Pegasos to other loss functions we need to satisfy two requirements: one is the loss function should be convex and the other is norm of subgradient should be atmost R. We have the loss functions which satisfy the above requirements namely Binary classification with log-loss, Regression with ∈-insensitive loss, Cost-sensitive multiclass categorization, Multiclass categorization with log-loss, Sequence prediction. For subgradients two properties are defined in this section.

(g) Earlier in the semester, we saw both hinge loss and log-loss as surrogate loss functions (to 0-1 loss). Describe, in English and/or with an example (your choice), when hinge loss and log-loss will (each) be low (close to or equal to 0) and high. In your answer, consider the asymptotic behavior of both (and speculate on the implications for using them as a loss function).

A. Hinge Loss is used by the classifiers if we need a decision boundary to classify the data. Consider classifying to -1 class or +1 class. If the prediction is greater than or equal to 1 then there will be 0 loss or sometimes low hinge loss but if its less than 1 then high hinge loss is assigned, then it will distance the -1 class from the boundary of +1 class. The hinge loss for the wrong classified data increases linearly in negative direction. The hinge loss penalizes heavily for the misclassification which maximizes margin between the classes, therefore we get a good decision boundary with hinge loss.
Log-loss/Logistic loss sees how close the predicted value from the true value is. If the deviation from true value is high, then we have high log-loss loss. Loss for the log-loss increases logarithmically. Log-loss is used for probabilistic models where we see how closely the values are related. If the prediction of log-loss is closer to 1 then we have low loss else high loss. Log-loss ties to minimize average loss across all inputs and improves for the true values. Hinge Loss can be used for margin-based classifiers whereas log-loss is used for probabilistic predictions.

(h) Describe, in English, equations 20 or 21 (the multiclass variants of hinge loss and log-loss). When will the loss be high and when will it be low?

A. For equation 20, multiclass hinge loss, if the predicted class is equal to the true class then the loss is 0 or sometimes low, whereas for the misclassified class loss will be high. So the classes are distanced with good bounds i.e. it wants to create distinguishing boundaries for different classes.
For equation 20, multiclass log-loss, if the predicted value is close to the true value then we have minimal loss, else we have high loss. It minimizes the loss by improving for true labels.

(i) The bottom of page 14 formally defines the subgradient for a function f (remember the notation <u, q> indicates the dot product between vectors u and q). Argue why the subgradient for binary hinge loss (presented in the table on page 15) is a subgradient. You may use, without proof, the fact that hinge loss is a convex function.

A. We know that hinge loss is a convex function. The derivative of the convex function has a gradient. So the equation of the hinge loss is max(0, 1- yi z), but 0 is not differentiable. We need to consider the subgradients. The subgradient should satisfy the 2 properties of f defined at bottom of page 14. The convex function will have the derivative touching from it. So from the properties of the subgradients we define subgradients for the term which is not equal to 0. This subgradient satisfies the properties of the subgradients which are in page 14. Now its differentiable, Therefore the subgradients of hinge loss are subgradients.

(j) What is the definition of a Gaussian kernel (sometimes also called an RBF—radial basis function)?

A. The Gaussian Kernel or RBF-Radial Bias Function is the kernel which measures the similarity between the input features. It is defined as:
$K(x,x^{l}) = \exp(-\Upsilon \, |x-x^{l}|^{2})$
Hyper parameter Υ controls the kernels width.
The closeness of input data is high then the difference between $|x-x^{l}|$will be small, therefore we have squared norm and hyper parameter which ensures kernel have larger values.

(k) Now write a (roughly) 1/2-1 page analysis of the entire paper. In your analysis, include a summary of the method, the results, and what you took away from the paper. Identify and discuss items that were confusing, underspecified, or counter-intuitive. This analysis may reuse, as appropriate, portions of your previous answers

A.

The paper "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM" by the authors {Shalev-Shwartz, Shai and Singer, Yoram and Srebro, Nathan and Cotter, Andrew}, implements the Stochastic subgradient descent for solving optimization problem by SVM and the algorithm they implemented is called Pegasos, which is a fast convergence optimization even for large datasets as it has good runtime compared to other SVM. They have discussed about the variants of the basic pegasos algorithm and have presented their algorithm objective with the help of analysis and experimentation by comparing with other models.
The basic Pegasos algorithm works by minimizing the primal objective of SVM given by Eq1 and Eq2 i.e., minimizing the unconstrained empirical loss with minimization term by selecting a good stepsize and a random single training instance. The run time does not depend on large datasets. Similar approaches are seen which are similar to authors work like Interior point methods, Decomposition methods, Primal Optimization, Stochastic gradient descent, Online methods.
The projection step is considered for basic pegasos algorithm to update the weight and it performed similar when this projection wasn't considered. The basic pegasos extends to the minibatches, mini-batch Pegasos algorithm rather than single instance, here k batches are considered. The k batches consider all the different features of the instances which the earlier basic pegasos algorithm with one single instance can't see. Due to the batches, the convergence will be faster. The authors have defined the analysis on the convergence of pegasos based on the lemmas and theorems stated.

The basic pegasos extends to kernelized model, where without directly using feature we use kernels. They modify the empirical minimization by including the kernel operator in the equations. Although the solution is represented with variable α, we still do subgradients w.r.t weight parameter w. Chapelle does subgradients w.r.t α, although α doesn't guarantee convex function. But w will guarantee convex. Chapelle introduces preconditioning for subgradients w.r.t α and it has observed good results, but number of iterations will be more compared to pegasos.
Pegasos can also use other loss functions besides hinge loss. In order to apply Pegasos to other loss functions we need to satisfy two requirements: one is the loss function should be convex and the other is norm of subgradient should be atmost R. We have the loss functions which satisfy the above requirements namely Binary classification with log-loss, Regression with ∈-insensitive loss, Cost-sensitive multiclass categorization, Multiclass categorization with log-loss, Sequence prediction. For subgradients two properties are defined w.r.t differentiability.
Finally, the paper discusses about including bias term for the model, just add the bias term in the loss function. It helps if the distribution is uneven. One approach is adding one feature for each instance, increasing dimension of input, do not include bias term explicitly, same convergence rate. Second approach is explicitly including bias term by not penalizing it, slow convergence. Third approach considers both the above methods, it works only for large batch size, convergence was same as similar to model which didn't consider the bias term. For fixed bias, its similar to SVM training without bias term.
The experiments for Pegasos are compared with SVM-Perf, LASVM, SVM-Light, Stochastic Dual Coordinate Ascent(DCA). Bias term is not considered for any model as there was not much change. The runtime is dependent on the regularization parameter λ. Evaluation with linear kernel considering the datasets: astro-ph, CCAT, cov1. No proper convergence for Pegasos or DCA. SVM-Perf works fast but still Pegasos and DCA has good run time. Strange case for Pegasus as its run time was longer for cov1 rather than CCAT.
Evaluation with Gaussian Kernel considering the datasets: Reuters, Adult, USPS, MNIST. LASVM works good on all the datasets and it outperforms Pegasos and DCA. Although Pegasos is a simple algorithm it performs good with a good runtime. The comparison is done based on the number of kernel evaluations which normalizes the throughput of each method, LASVM performed good with this too. While Pegasos and DCA being simple algorithms its performance has dominated by kernel evaluations. The effect of λ is that runtime increases propotional to λ. DCA performs better if λ is small. For Astro-ph sampling without replacement outperforms the uniform i.i.d sampling, for the rest of the paper uniform sampling with replacements are considered.
The pegasos is compared with Norma [24] by Kivinen, Smola, Williamson and to the method by Zhang [37]. Pegasos performs better than Norma and Zhang's method, as it converges faster. Norma poor performance is because of small λ. Zhang method has learning rate fixed, it takes computation time whereas Pegasos saves it.
Overall Pegasos justifies the objective of the authors with a good runtime with fast convergence rate. By including all the complexities such as non-linear kernel, mini-batches it still performs in competitive runtime. The experimentations gave results according to the objective function.The details regarding the mini-batches I think were underspecified, would expect more details for that.

References:

Dataset: https://redirect.cs.umbc.edu/courses/graduate/678/spring23/materials/a4-data/2d-kmeans-input.ssv

Welcome to PyTorch Tutorials — PyTorch Tutorials 1.13.1+cu117 documentation

ShalevSiSrCo10.pdf (huji.ac.il)

3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.2.2 documentation

12-knn-kmeans.pdf (umbc.edu)

13-kernel-svm.pdf (umbc.edu)

14-em.pdf (umbc.edu)

K-means clustering (umbc.edu)

2.1. Gaussian mixture models — scikit-learn 1.2.2 documentation

scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation

NumPy Reference — NumPy v1.24 Manual

numpy.random.poisson — NumPy v1.24 Manual

Tutorials — Matplotlib 3.7.1 documentation