CSCI 5525 Machine Learning HW2

Problem1

Method
We solve the SVM dual problem by solving a convex quadratic programming problem

$$\text{(P1)} \quad \begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_i \alpha_i \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq c \end{aligned}$$

using CVXOPT package. Once we found the solution to dual variables $\boldsymbol{\alpha}^*$, the SVM parameters can be found as follows if there exist α_j s.t. $0 < \alpha_j < c$

$$\mathbf{w}^* = \sum_{i:\alpha_i > 0} \alpha_i y_i \mathbf{x}_i \tag{1}$$

$$b^* = y_j - \mathbf{x}_j^T \mathbf{w}^* \tag{2}$$

and the number of support vectors is the number of positive elements of $\boldsymbol{\alpha}^*$.
Results
Table 1 is the error rate on the training and validation data sets when using different c values. Fig. 1 is the plot of test performance. Among the given c values, 0.01 is the best one. The classifier tends to be overfitting on the training data once we increase the c value, since the training error is already close to 0 but the classifier behaves worse on the test data.

Table 1: svm_cvx error statics

	c=0.01	c=0.1	c=1	c=10	c=100
traning error mean	0.0024	0.0	0.0	0.0	0.0
traning error std std	0.0002	0.0	0.0	0.0	0.0
test error mean	0.0154	0.0285	0.028	0.026	0.0245
test error std	0.0043	0.008	0.035	0.005	0.007

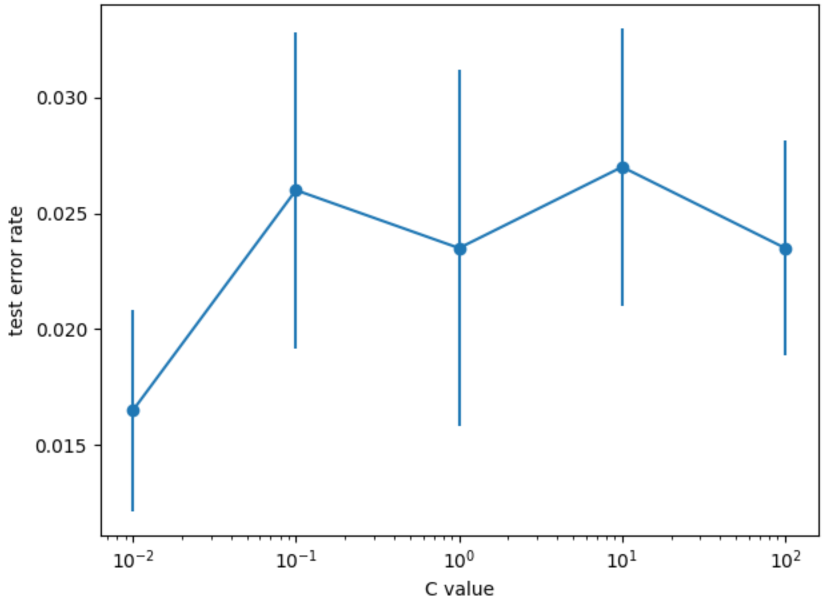

Figure 1: Test performance as C increases

Fig. 2 3 show the geometric margin and number of support vectors as C increases. For larger values of C, the misclassified examples are severely punished, then it would make sense that the classifier chooses a smaller margin with less support vectors.

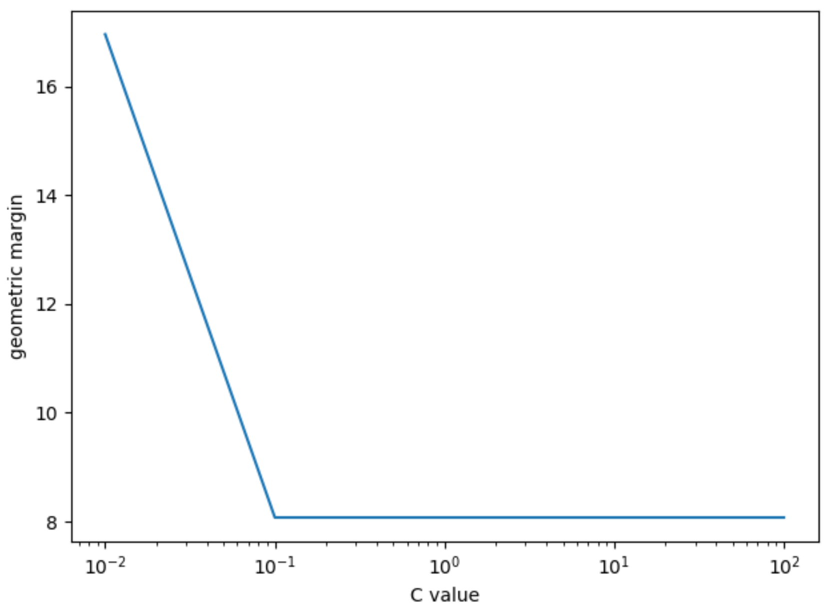

Figure 2: Geometric margin as C increases

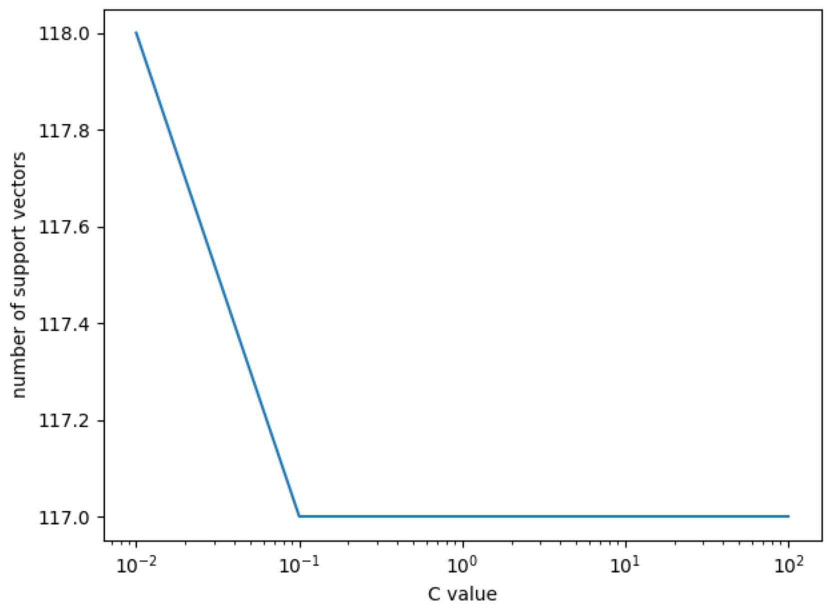

Figure 3: Number of support vectors as C increases

Questions
Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and $\boldsymbol{\xi}$

$$\text{(P2)} \qquad \begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \boldsymbol{\xi}_i \\ & y_i f(\mathbf{x}_i) \geq 1 - \boldsymbol{\xi}_i \\ & \boldsymbol{\xi}_i \geq 0 \end{aligned}$$

The inequality constraints are equivalent to

$$\boldsymbol{\xi}_i(\mathbf{w}, b) = \max(0, 1 - y_i f(\mathbf{x}_i)) \tag{3}$$

hence (P2) is equivalent to the following problem which is of the hinge loss form

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \boldsymbol{\xi}_i(\mathbf{w}, b)$$

In the above derivation, all the margin constraints are satisfied because of the equivalency.

Problem2

We solve the hinge loss formation of the SVM primal problem either by Pegasos or Softplus.
Method: Pegasos
Pegasos solves the following non-smooth convex optimization problem by projected subgradient method

$$\text{(P3)} \qquad \min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_{(\mathbf{x},y)\in S} l(\mathbf{w}; (\mathbf{x}, y))$$

where

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

Algorithm 1: Primal Estimated sub-GrAdient SOlver for SVM

Input: Training set S, regularization parameter λ, number of iterations T, size of training subset k

Initialize: Choose w^0 s.t. $||w^0|| \leq 1/\sqrt{\lambda}$

For $t = 0, 1, \ldots, T-1$

  Choose $A_t \subset S$, where $|A_t| = k$

  Set $A_t^+ = \left\{i \in A_t : \ y_i\langle w^t, x_i \rangle < 1\right\}$

  Set step size $\eta_t = \frac{1}{\lambda t}$

  Set $w^{t+\frac{1}{2}} = (1 - \eta_t \lambda) w^t + \frac{\eta_t}{k} \sum_{(x,y) \in A_t^+} yx$ (subgradient descent)

  Set $w^{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{||w^{t+\frac{1}{2}}||}\right\} w^{t+\frac{1}{2}}$ (project $w^{t+\frac{1}{2}}$ into optimal set B)

Output: w^T

Method: Softplus

Softplus solves the non-smooth convex optimization problem by approximating the hinge loss function with the softplus function

$$f_a(x) = a \log(1 + \exp(x/a)) \tag{4}$$

since $\max(0, x) = \lim_{a \to 0} f_a(x)$, thus the problem becomes the following smooth convex optimization problem

$$\text{(P4)} \qquad \min_{\mathbf{w}} \lambda ||\mathbf{w}||^2 + \frac{1}{N} a \sum_{i=1}^{N} \log\left(1 + \exp((1 - y_i \mathbf{w}^T \mathbf{x}_i)/a)\right)$$

and the gradient is

$$\nabla_{\mathbf{w}} = 2\lambda \mathbf{w} - \frac{1}{N} \sum_{i=1}^{N} \frac{y_i \exp((1 - y_i \mathbf{w}^T \mathbf{x}_i)/a)}{1 + \exp((1 - y_i \mathbf{w}^T \mathbf{x}_i)/a)} \mathbf{x}_i \tag{5}$$

Results

Table 2 is the average run time of Pegasos and Softplus over 5 runs on the entire dataset along with the standard deviation. We use the ktot (the total number of gradient computations, and the batch of size k would account for k computations) as the stopping criterion, so the computation complexities for different k values should be the same. From the table we saw that the run time is more relevant with the number of iterations since smaller k values would lead to more iterations and the cost per iteration might be more dominant than the computation complexity in the actual run time.

Fig. 4 is the plot of Pegasos iterations for 5 runs with each k value and Fig. 5 is for Softplus. From the plots we can see that (1) Pegasos converges faster than Softplus (2) Pegasos reaches smaller loss function value (3) Softplus cannot work when $k = 1$ (which is yet to find out the reason) while Pegasos can still work.

Table 2: Run time statics

	k=1	k=20	k=200	k=1000	k=2000
Pegasos run time mean	12.816	6.354	0.721	0.712	0.736
Pegasos run time std	0.194	0.082	0.021	0.017	0.035
Softplus run time mean	15.101	14.754	0.83	0.854	0.86
Softplus run time std	1.093	0.365	0.035	0.025	0.02

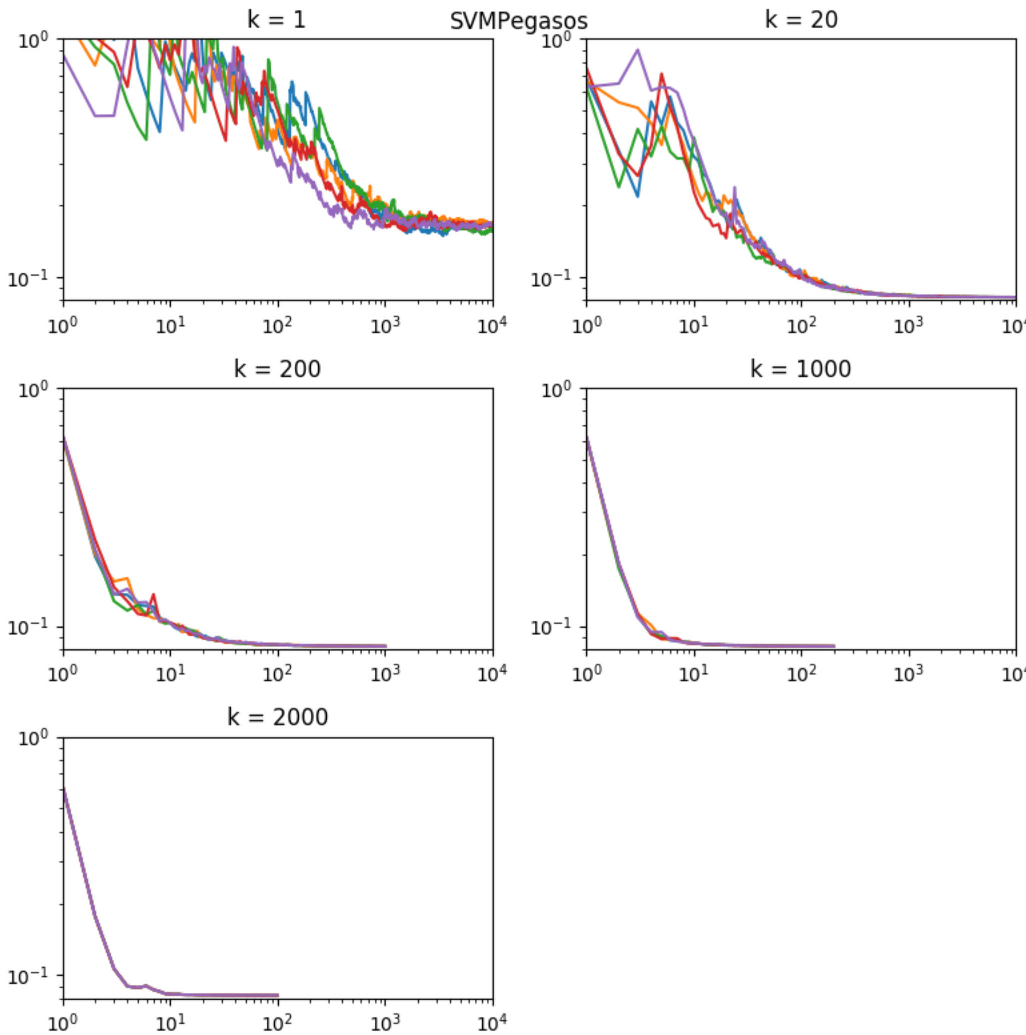

Figure 4: Pegasos loss function values over iterations

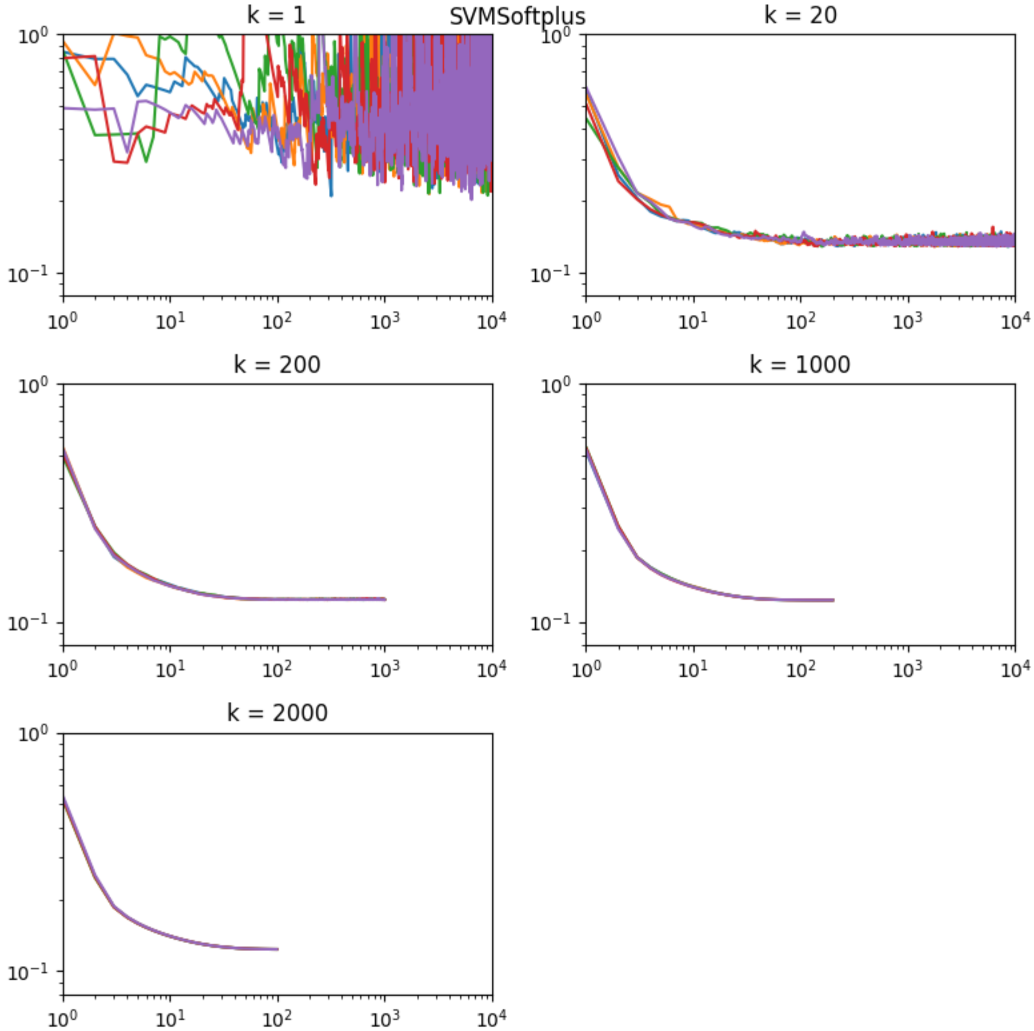

Figure 5: Softplus loss function values over iterations