CS 5350/6350: Machine Learning Spring 2020

Homework 4

Handed out: 19 Mar, 2020
Due date: 11:59pm, 4 Apr, 2020

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free to discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You do not need to include original problem descriptions in your solutions. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 15 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- *Your code should run on the CADE machines.* **You should include a shell script, `run.sh`, that will execute your code in the CADE environment. Your code should produce similar output to what you include in your report.**

  You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.
- Please do not hand in binary files! We will *not* grade binary submissions.
- The homework is due by **midnight of the due date**. Please submit the homework on Canvas.

1 Paper Problems [40 points + 10 bonus]

1. [9 points] The learning of soft SVMs is formulated as the following optimization problem,

$$\begin{aligned} \min_{\mathbf{w},b,\{\xi_i\}} \quad & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \xi_i \\ \text{s.t. } \forall 1 \le i \le N, \quad & y_i(\mathbf{w}^\top\mathbf{x}_i + b) \ge 1 - \xi_i, \\ & \xi_i \ge 0 \end{aligned}$$

   where N is the number of the training examples. As we discussed in the class, the slack variables $\{\xi_i\}$ are introduced to allow the training examples to break into the margin so that we can learn a linear classifier even when the data is not linearly separable.

(a) [3 point] What values ξ_i can take when the training example $\mathbf{x}_i$ breaks into the margin?

Answer. If x_i breaks into the margin, but correctly classified, then it should be between 0 and 1. That is, when $\xi_i \in (0, 1]$.

If x_i breaks into the margin, but is incorrectly classified, then it should be bigger than 1 and less than 2. That is, when $1 < \xi_i < 2$.

(b) [3 point] What values ξ_i can take when the training example $\mathbf{x}_i$ stays on or outside the margin?

Answer. If x_i is correctly classified, then ξ_i should be zero.

If x_i is incorrectly classified, but stays on or outside the margin, then $\xi_i \geq 2$.

(c) [3 point] Why do we incorporate the term $C \cdot \sum_i \xi_i$ in the objective function? What will happen if we throw out this term?

Answer. We incorporate the sum $\sum_i \xi_i$ to penalize weight vectors that make mistake. In fact, it allows as few examples as possible to violate the margin. And incorporate C in order to trade off between large margin and small hinge-loss.

The regularization term tries to maximize the margin. So, if we throw out the term $C \cdot \sum_i \xi_i$, then margin can be maximized arbitrarily, consequently, we will allow many examples to violate the margin and so most examples will be classified wrongly. Moreover, the minimum of $\min_{\mathbf{w},b,\{\xi_i\}} \ \frac{1}{2}\mathbf{w}^\top\mathbf{w}$ would be 0 always (note that 0 would be one of the possible answers of the objective function $\frac{1}{2}\mathbf{w}^\top\mathbf{w}$, which is the smallest possible, as ξ_i can go towards infinity and so satisfy in constraints).

2. [6 points] Write down the dual optimization problem for soft SVMs. Please clearly indicate the constraints, and explain how it is derived. (Note: do NOT directly copy slides content, write down your own understanding.)

Answer. The primal SVM is:

$$\min_{\mathbf{w},b,\{\xi_i\}} \ \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \xi_i \quad s.t. \quad \forall i,\, y_i(\mathbf{w}^\top\mathbf{x}_x + b) \geq 1 - \xi_i,\, \xi_i \geq 0.$$

Its Lagrange form is:

$$\min_{\mathbf{w},b,\{\xi_i\}} \max_{\alpha_i \geq 0,\, \beta_i \geq 0} \ \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \xi_i - \sum_i \beta_i\xi_i - \sum_i \alpha_i(y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1 + \xi_i).$$

This is obtained in a standard way. Just we should note that the constraints are $g_i(x) \leq 0$ and here $-y_i(\mathbf{w}^\top\mathbf{x}_x + b) + 1 - \xi_i \leq 0$ and $-\xi_i \leq 0$, which justifies the negative signs for two last sums.

The dual SVM comes from changing the min max problem to max min. So, it is:

$$\max_{\alpha_i \geq 0,\, \beta_i \geq 0} \min_{\mathbf{w},b,\{\xi_i\}} \ \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \xi_i - \sum_i \beta_i\xi_i - \sum_i \alpha_i(y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1 + \xi_i).$$

In general, the answer to the dual optimization problem, say d^*, is less tan or equal to the answer of primal optimization, say p^*, i.e. $d^* \leq p^*$. The difference $p^* - d^*$ is called duality gap. However, thanks to Slater's Theorem, for SVM, duality gap is 0, that is, $d^* = p^*$. This means that solving dual SVM is solving primal SVM.

To solve the dual SVM, we call the objective function L and first solve the inner min-optimization problem. As usual, we go through partial derivatives. So, we have to solve the system of equations

$$\frac{\partial L}{\partial \mathbf{w}} = 0, \quad \frac{\partial L}{\partial b} = 0, \quad \frac{\partial L}{\partial \xi_i} = 0.$$

Solving these, we get

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i, \quad \sum_i \alpha_i y_i = 0, \quad \alpha_i + \beta_i = C.$$

Substituting these into dual SVM we get

$$\max_{\alpha_i \geq 0, \beta_i \geq 0} \; -\frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_i \alpha_i \quad s.t. \quad \sum_i \alpha_i y_i = 0, \; \alpha_i + \beta_i = C, \; \forall i.$$

Now we can use $0 \geq \beta_i = C - \alpha_i$ to remove β_i and get the following form of dual SVM:

$$\max_{0 \leq \alpha_i \leq C, \forall i\}, \sum_i \alpha_i y_i = 0} \; -\frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_i \alpha_i.$$

Employing the fact that $\max(-f) = -\min(f)$ we can turn the dual SVM into the following optimization problem:

$$\min_{0 \leq \alpha_i \leq C, \forall i\}, \sum_i \alpha_i y_i = 0} \; \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_i \alpha_i.$$

This is now a quadratic optimization problem that we can utilize existing libraries, for instance in python, to solve it.

3. [10 points] Continue with the dual form. Suppose after the training procedure, you have obtained the optimal parameters.

   (a) [4 points] What parameter values can indicate if an example stays outside the margin?

Answer.

   (b) [6 points] if we want to find out which training examples just sit on the margin (neither inside nor outside), what shall we do? Note you are not allowed to examine if the functional margin (i.e., $y_i(\mathbf{w}^\top \mathbf{x}_i + b)$) is 1.

Answer.

4. [6 points] How can we use the kernel trick to enable SVMs to perform nonlinear classification? What is the corresponding optimization problem?

Answer.

5. [9 points] Suppose we have the training dataset shown in Table 1. We want to learn a SVM classifier. We initialize all the model parameters with 0. We set the learning rates for the first three steps to $\{0.01, 0.005, 0.0025\}$. Please list the sub-gradients of the SVM objective w.r.t the model parameters for the first three steps, when using the stochastic sub-gradient descent algorithm.

x_1	x_2	x_3	y
0.5	-1	0.3	1
-1	-2	-2	-1
1.5	0.2	-2.5	1

Table 1: Dataset

Answer. We have $\mathbf{w}_0 = [0, 0, 0]$, $b = 0$, $\mathbf{w}^0 = [0, 0, 0, 0]$. We set $N = 3$, the number of examples, and $C = 1/3$. According to the algorithm we know that $\nabla J^t = [\mathbf{w}_0; 0] - C \cdot N y_i \mathbf{x}_i$ if $y_i \mathbf{w}^\top \mathbf{x}_i \leq 1$ and $[\mathbf{w}_0; 0]$ otherwise. Updating rule for $\mathbf{w}$ is $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t \nabla J^t$. Therefore, we have:

(1) Since $y_1 \mathbf{w}^{0\top} \mathbf{x}_1 = 0 \leq 1$, we have $\nabla J^1 = [-0.5, 1, -0.3, -1]$ and $\mathbf{w}^1 = \mathbf{w}^0 - 0.01 \nabla J^1 = [0.005, -0.01, 0.003, 0.01]$.

(2) Since $y_2 \mathbf{w}^{0\top} \mathbf{x}_2 = -0.019 \leq 1$, we have $\nabla J^2 = [-0.995, -2.01, -1.997, 1]$ and $\mathbf{w}^2 = \mathbf{w}^1 - 0.005 \nabla J^2 = [0.01, 0., 0.013, 0.005]$.

(3) Since $y_3 \mathbf{w}^{0\top} \mathbf{x}_3 = -0.0125 \leq 1$, we have $\nabla J^3 = [-1.49, -0.2, 2.513, -1]$ and $\mathbf{w}^3 = \mathbf{w}^2 - 0.0025 \nabla J^2 = [0.014, 0.001, 0.007, 0.007]$.

6. [**Bonus**][10 points] Let us derive a dual form for Perceptron. Recall, in each step of Perceptron, we add to the current weights $\mathbf{w}$ (including the bias parameter) $y_i \mathbf{x}_i$ for some misclassified example $(\mathbf{x}_i, y_i)$. We initialize $\mathbf{w}$ with $\mathbf{0}$. So, instead of updating $\mathbf{w}$, we can maintain for each training example i a mistake count c_i — the number of times the data point $(\mathbf{x}_i, y_i)$ has been misclassified.

- [2 points] Given the mistake counts of all the training examples, $\{c_1, \ldots, c_N\}$, how can we recover $\mathbf{w}$? How can we make predictions with these mistake counts?

Answer. Since the updating rule is $\mathbf{w}_{t+1} = \mathbf{w}_t + r(y_i \mathbf{x}_i)$, where r is the learning rate, with $\mathbf{w}_0 = 0$ we can recover $\mathbf{w}$ as $\mathbf{w} = \mathbf{w}_0 + r \sum_{i=1}^N c_i y_i \mathbf{x}_i = r \sum_{i=1}^N c_i y_i \mathbf{x}_i$.
The prediction of example $\mathbf{x}$ would be $\text{sgn}((\sum_{i=1}^N c_i y_i \mathbf{x}_i)^\top \mathbf{x}) = \text{sgn}(\sum_{i=1}^N c_i y_i \mathbf{x}_i^\top \mathbf{x})$. Note that r is a positive constant and so there is no role in sign.

- [3 points] Can you develop an algorithm that uses mistake counts to learn the Perceptron? Please list the pseudo code.

Answer. The algorithm is:

Input: Training set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^n, y_i \in \{-1, 1\}$;
Output: Weight vector $\mathbf{w}$;
Initialize $c_i = 0$ for $i = 1, \ldots, N$;
for *epoch* $= 1, \ldots, T$ **do**
  Shuffle the data;
  for *each training example* $(x_i, y_i) \in D$ **do**
    if $y_i \sum_{j=1}^N c_j y_j \mathbf{x}_j^\top \mathbf{x}_i < 0$ **then**
      $c_i = c_i + 1$
    end
  end
end
Return $\sum_{j=1}^N c_j y_j \mathbf{x}_j^\top$

- [5 points] Can you apply the kernel trick to develop a nonlinear Perceptron? If so, how do you conduct classification? Can you give the pseudo code fo learning this kernel Perceptron?

Answer.

2 Practice [60 points + 10 bonus]

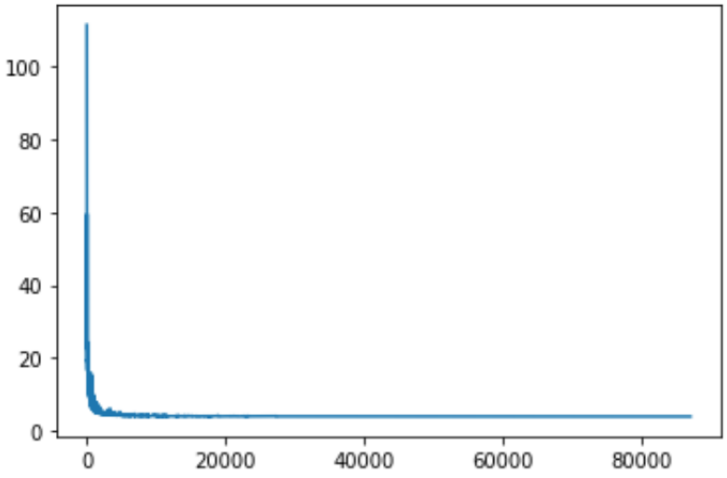
1. [2 Points] Update your machine learning library. Please check in your implementation of Perceptron, voted Perceptron and average Perceptron algorithms. Remember last time you created the folders "Perceptron". You can commit your code into the corresponding folders now. Please also supplement README.md with concise descriptions about how to use your code to run these algorithms (how to call the command, set the parameters, etc). Please create a new folder "SVM" in the same level as these folders.

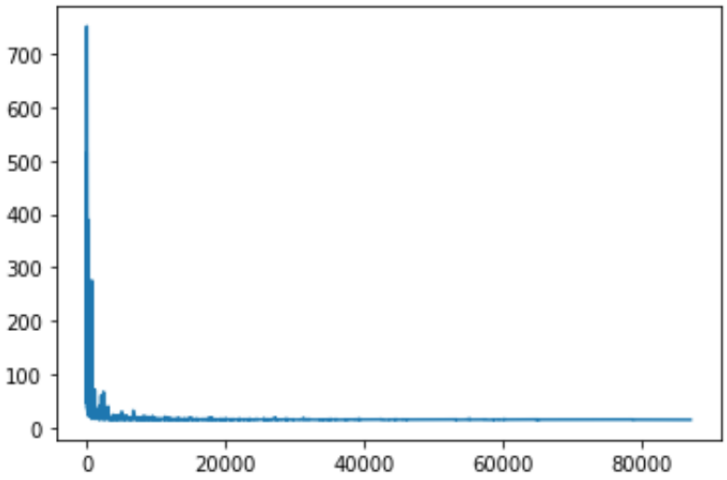
2. [28 points] We will first implement SVM in the primal domain with stochastic subgradient descent. We will reuse the dataset for Perceptron implementation, namely, "bank-note.zip" in Canvas. The features and labels are listed in the file "classification/data-desc.txt". The training data are stored in the file "classification/train.csv", consisting of 872 examples. The test data are stored in "classification/test.csv", and comprise of 500 examples. In both the training and test datasets, feature values and labels are separated by commas. Set the maximum epochs T to 100. Don't forget to shuffle the training examples at the start of each epoch. Use the curve of the objective function (along with the number of updates) to diagnosis the convergence. Try the hyperparameter C from $\{\frac{100}{873}, \frac{500}{873}, \frac{700}{873}\}$. Don't forget to convert the labels to be in $\{1, -1\}$.

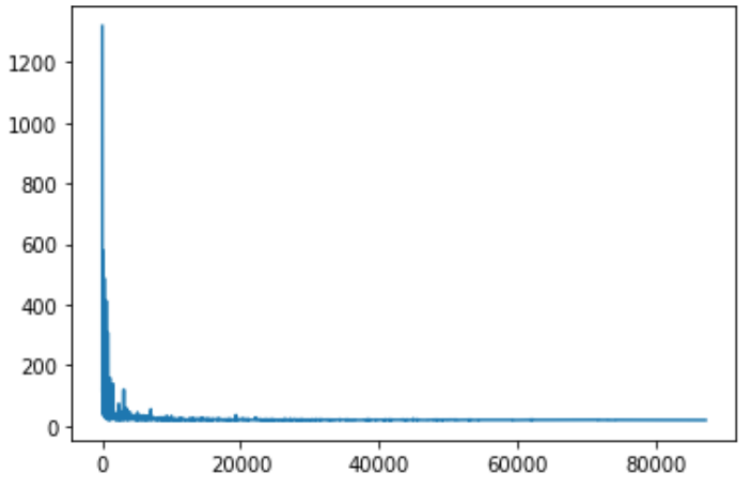
   (a) [12 points] Use the schedule of learning rate: $\gamma_t = \frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$. Please tune γ_0 and d to ensure convergence. For each setting of C, report your training and test error.

Answer. I have set $\gamma_0 = 0.001$ and $d = 0.0001$. The results are as follows. Figures are for $C = \frac{100}{873}, \frac{500}{873}, \frac{700}{873}$ respectively.

	873*C	Train Error	Test Error	Weights (rounded to 5 decimals)
1	100.0	0.011468	0.012	[-0.87184, -0.6089, -0.67538, -0.04169, 1.2741]
2	500.0	0.008028	0.010	[-1.90746, -1.18856, -1.4124, -0.21485, 1.99582]
3	700.0	0.011468	0.012	[-2.31125, -1.4003, -1.6729, -0.25868, 2.22483]



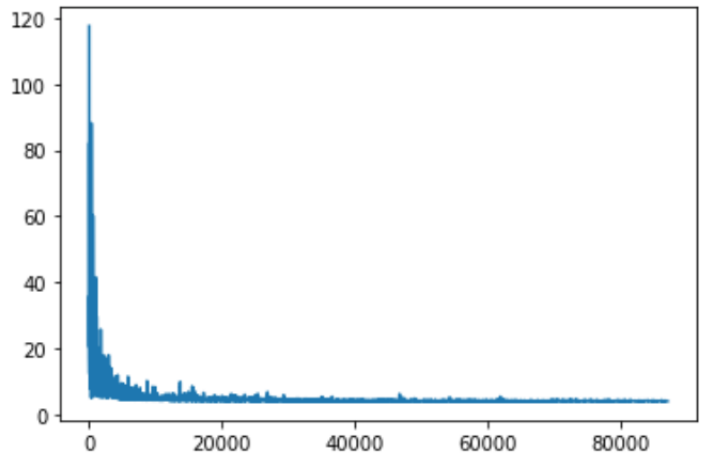


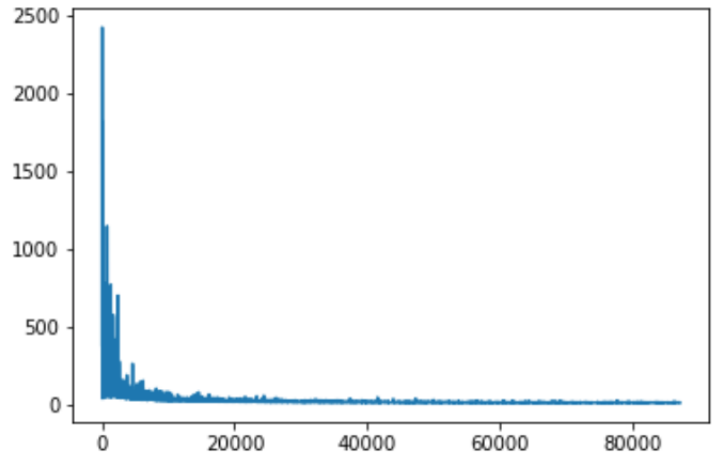


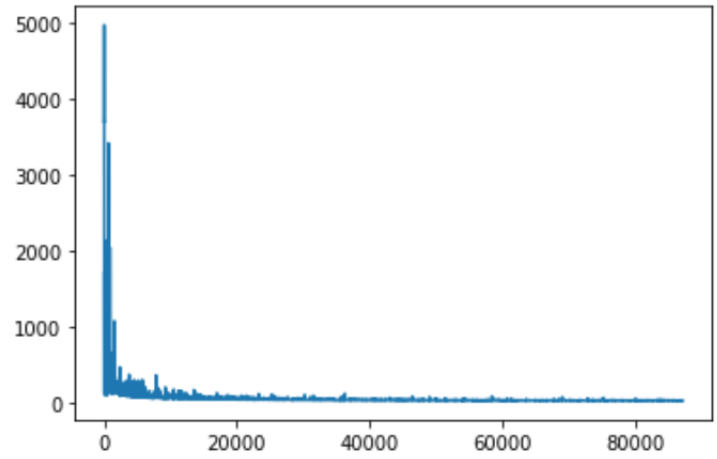
(b) [12 points] Use the schedule $\gamma_t = \frac{\gamma_0}{1+t}$. Report the training and test error for each setting of C.

Answer. Figures are for $C = \frac{100}{873}, \frac{500}{873}, \frac{700}{873}$ respectively.

	873*C	Train Error	Test Error	Weights (rounded to 5 decimals)
1	100.0	0.006881	0.008	[-0.99179, -0.69138, -0.78751, -0.05951, 1.43281]
2	500.0	0.011468	0.010	[-2.28519, -1.42427, -1.64892, -0.24906, 2.25194]
3	700.0	0.008028	0.010	[-2.68562, -1.58096, -1.92669, -0.33096, 2.67223]







(c) [6 points] For each C, report the differences between the model parameters learned from the two learning rate schedules, as well as the differences between the training/test errors. What can you conclude?

Answer. The weight vectors as model parameters learned from the two learning rate schedules are presented in tables in parts (a) and (b). If for each C we calculate Euclidean distance between corresponding weight vectors, regarding learning rate, we get the following numbers:

$$\|W^a_{100/873} - W^b_{100/873}\| = 0.243$$

$$\|W^a_{500/873} - W^b_{500/873}\| = 0.566$$

$$\|W^a_{700/873} - W^b_{700/873}\| = 0.665$$

As we see, their differences in magnitude is not considerable. The direction of all weight vectors are the same as well.

However, looking at train errors, we observe that for $C = 100/873, 700/873$ it is decreased for $\gamma_0 = d = 0.001$, but for $C = 500/873$ it is increased. Concerning test errors, we see that for $C = 500/873$ it is the same for both setting of hyperparameters but for $C = 100/873, 700/873$ they are declined when we changed $\gamma_0 = 0.001$ and $d = 0.0001$ in part (a) to $\gamma_0 = d = 0.001$ in part (b).

3. [30 points] Now let us implement SVM in the dual domain. We use the same dataset, "bank-note.zip". You can utilize existing constrained optimization libraries. For Python, we recommend to use "scipy.optimize.minimize", and you can learn how to use this API from the document at `https://docs.scipy.org/doc/scipy-0.19.0/reference/generated/scipy.optimize.minimize.html`. For Matlab, we recommend to use the internal function "fmincon"; the document and examples are given at `https://www.mathworks.com/help/optim/ug/fmincon.html`. For R, we recommend to use the "nloptr" package with detailed documentation at `https://cran.r-project.org/web/packages/nloptr/nloptr.pdf`. In principle, you can choose any nonlinear optimization algorithm. But we recommend to use L-BFGS for their robustness and excellent performance in practice.

   (a) [10 points] First, run your dual SVM learning algorithm with C in $\{\frac{100}{873}, \frac{500}{873}, \frac{700}{873}\}$. Recover the feature weights $\mathbf{w}$ and the bias b. Compare with the parameters learned with stochastic sub-gradient descent in the primal domain (in Problem 2) and the same settings of C, what can you observe? What do you conclude and why?

Answer.

   (b) [15 points] Now, use Gaussian kernel in the dual form to implement the nonlinear SVM. Note that you need to modify both the objective function and the prediction. The Gaussian kernel is defined as follows:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\gamma}).$$

   Test γ from $\{0.1, 0.5, 1, 5, 100\}$ and the hyperparameter C from $\{\frac{100}{873}, \frac{500}{873}, \frac{700}{873}\}$. List the training and test errors for the combinations of all the γ and C values. What is the best combination? Compared with linear SVM with the same settings of C, what do you observe? What do you conclude and why?

Answer.

   (c) [5 points] Following (b), for each setting of γ and C, list the number of support vectors. When $C = \frac{500}{873}$, report the number of overlapped support vectors between consecutive values of γ, i.e., how many support vectors are the same for $\gamma = 0.01$ and $\gamma = 0.1$; how many are the same for $\gamma = 0.1$ and $\gamma = 0.5$, etc. What do you observe and conclude? Why?

Answer.

   (d) [**Bonus**] [10 points] Implement the kernel Perceptron algorithm you developed in Problem 8 (Section 1). Use Gaussian kernel and test γ from $\{0.1, 0.5, 1, 5, 100\}$. List the training and test errors accordingly. Compared with the nonlinear SVM, what do you observe? what do you conclude and why?

Answer.