CS 5350/6350: Machine Learning Spring 2021

Homework 4

Handed out: 30 Mar, 2021
Due date: 11:59pm, 15 Apr, 2021

- 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 can ξ_i take when the training example $\mathbf{x}_i$ breaks into the margin?

When the training example breaks into the margin, ξ can only take values where:

$$\xi \geq 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)$$

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

When the training example stays on or outside the margin, ξ can only take values where:

$$\xi \geq 0$$

(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?

If we throw out the term $C \cdot \sum_i \xi_i$ then we will go from soft SVM back to a Hard SVM and this will mean that we will not allow separating hyperplanes to label any example incorrectly.

We become too strict with the hyperplanes we can use.

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.)

The support vector machine function originated from the problem of trying to bound an infinite hypothesis space given an agnostic setting. To solve this problem, we figured out we could describe different infinite hypothesis spaces by identifying their VC dimensions. Then, through occam's razor, we were able to bound the generalization error of a single hypothesis by incorporating the VC dimension of an infinite hypothesis space.

Therefore, to further reduce our generalization error, we decided to reduce our infinite hypothesis space which lead to the idea of maximizing the margin of a separating hyperplane. We observed we could reduce the hypothesis space if we bounded the functional margin to be only 1. This method doesn't work too often on the real world as the method implies the training data is linearly separable. This method is know as Hard SVM.

To relax the constraints Hard SVM imposes on the data, we then created Soft SVM that finds the best margin for a data by allowing some examples to be incorrectly labeled. Soft SVM is exactly Hard SVM but with an added term that penalizes hypothesis functions based on their errors.

Both terms together look like this:

$$SVMobjectivefunction = \min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i max(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b))$$

The first term is the optimization of the margin while the second term is the penalty for every point that was incorrectly classified. The C in the second term is simply an arbitrary number that scales the penalty of every incorrectly classified example. The penalty of a point is derived by reducing the margin to allow for a slack. This slack is called ξ. Because we set the functional margin equal to one in the hard SVM, in the Soft SVM we subtract 1 by ξ. Setting Hard equal to one basically means that every point point lies on or outside the margin. ξ is used only to reduce the margin therefore this means that ξ needs to always be greater than or equal to zero. If ξ is less than zero then we will impose a stricter margin causing an even Harder SVM.

Finally the term $max(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)$ means that when classifying an example, the penalty can only be greater than zero or equal to zero. The 0 simply places a lower bound on the penalty and the $1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)$ term states that the penalty equals the distance a point is away from the incorrect side of their respective margin (i.e., a point increases its penalty when its inside the margin, when it gets incorrectly classified, and starts to greatly increase its penalty the further it moves from the margin)

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?

   If you look at this term:

   $$max(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)$$

   you can notice that 0 will always be returned when an example stays outside the margin. Therefore parameters yi, xi, and b all influence a points classification of being inside or outside the margin.

   (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.

   You could check if examples are neither inside nor outside the margin if you look at this term :

   $$max(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)$$

and check which examples are correctly classified yet their maximum penalty is still 0.

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

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

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?
   - [3 points] Can you develop an algorithm that uses mistake counts to learn the Perceptron? Please list the pseudo code.
   - [5 points] Can you apply the kernel trick to develop an nonlinear Perceptron? If so, how do you conduct classification? Can you give the pseudo code fo learning this kernel Perceptron?

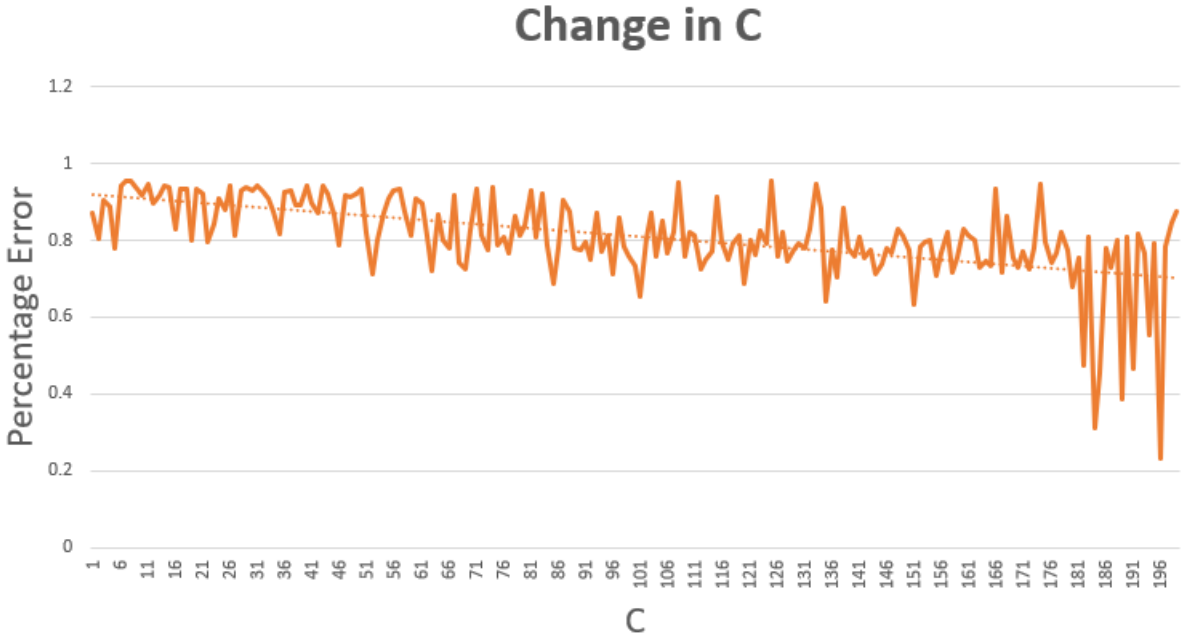
2 Practice [60 points + 10 bonus]

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.

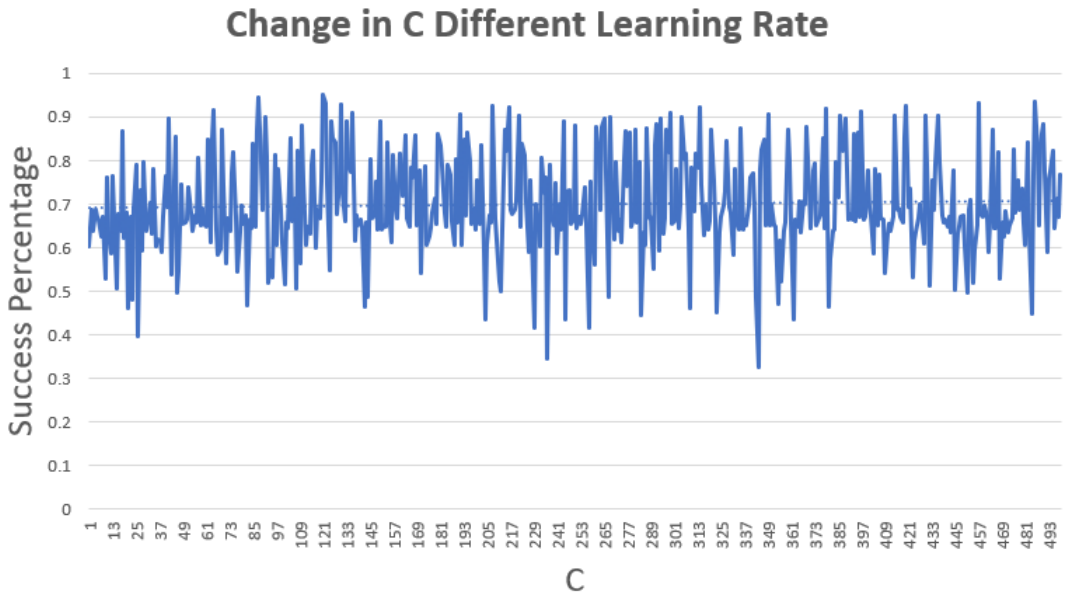
2. [28 points] We will first implement SVM in the primal domain with stochastic sub-gradient descent. We will reuse the dataset for Perceptron implementation, namely,

"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 $\{1, -1\}$.

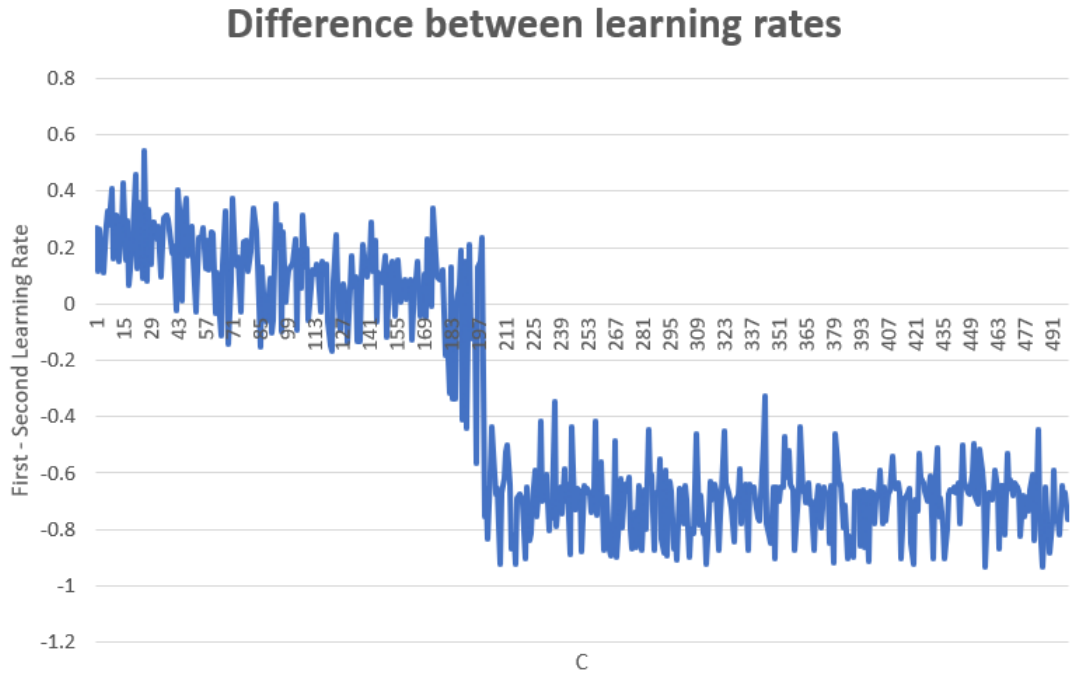
(a) [12 points] Use the schedule of learning rate: $\gamma_t = \frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$. Please tune γ_0 and d to ensure convergence. For each setting of C, report your training and test error.



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



(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?



This graph calculates the differences between the learning rates of the first learning rate(question a) and the second learning rate(question b). From this graph we can conclude that the second learning rate performs much better as C increases. Both learning rates are equally as good as this graph also shows that the first learning rate(question a) performs equally as good as the second learning rate (question) when C is smaller.

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?

   (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?

   (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?

   (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?