CS 5350/6350: Machine Learning Fall 2023

Homework 4

Handed out: 7 Nov, 2023
Due date: 11:59pm, 22 Nov, 2023

- 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?

The slack variable ξ_i will range between 0 and 1 inclusive. When a training example breaks into the margin, that means that it is not correctly classified by the hyperplane which is defined by the weight vector $\mathbf{w}$ and the bias term. If ξ_i is 1, then the example is classified incorrectly and if 0 it is correctly classified.

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

When a training example lies on or outside the margin, it means that it is either perfectly classified or classified wrong completely. Therefore it can take the value of 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?

The term $C \cdot \sum_i \xi_i$ is in the objective function because it balances the margin maximization and error minimization. It controls the trade-off between these two goals. The optimization problem would focus only on maximizing the margin which would leave to a biased model and over-fitting.

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 dual optimization problem is a variant of the primal optimization problem from soft SVMs where we maximize a function of Lagrange multipliers. This helps with computational efficiency and interpretation. The dual optimization problem is seeking to minimize:

$L(w, b, \alpha) = 1/2 w^T w + C\Sigma_i \xi_i + \Sigma_i \alpha(1 - \xi_i - y_i(x_i^T w + b))$

Where w is the weights, b is the bias, alpha are the Lagrange multipliers, C is the trade-off, ξ_i is the slack variable. i is between 1 to N. We wish to minimize this problem, such that:

$min(w, \xi_i, b) L(w, \xi_i, \alpha), \xi_i >= 0.$

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?

The parameter values that indicate if an example stays outside the margin are the Lagrange multipliers. They are associated with the constraints in the problem. Lagrange multipliers will be zero if the prediction is correct.

(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.

We can analyze the lagrange multipliers associated with the constraints in the dual soft SVMs. The non zero lagrange multipliers that are still small are the closest to being on the margin.

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

   We can use the kernel trick to allow SVMs to perform nonlinear classification by allowing the input data to hypothetically exist in a higher dimensional space which allows the data to be linearly seperable by a hyperplane. The optimization problem with the kernel trick is roughly the same as the original SVM problem, but with a kernel function added which takes the rough form like $k(x_1, x_2) = \alpha(x_1) * alpha(x_2)$.

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 $\{0.01, 0.005, 0.0025\}$ and hyperparameter $C = 1$. 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

   With w = 0,0,0 , b = 0, and C = 1, we can set x = the first set of entries for xi (0.5, -1, 0.3) and y = 1. We can then make a prediction p given w*x + b. The margin violation is the max of 0 and (1 - y * p). The subgradient for w is then equal to -y * x * margin violation. The subgradient with respect to b is -y * margin violation. We can then calculate the learning rates for w and b as learning rate * subgradient w or b respectively. Ultimately, we find that the sub-gradients are:

   For LR 0.01: [0.03069384 0.01244718 -0.00245973]

   For LR 0.005: [0.01517235 0.00611165 -0.00111309]

   For LR 0.0025: [0.00754295 0.00302789 -0.00052804]

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.

   Done :)

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: $\gamma_t = \frac{\gamma_0}{1+\frac{\gamma_0}{a}t}$. Please tune $\gamma_0 > 0$ and $a > 0$ to ensure convergence. For each setting of C, report your training and test error.

   I tuned gamma to be 1 and alpha to be 100. My train and test error for each learning rate are as follows:

   C=0.1145475372279496: Train error = 0.4954

   C=0.1145475372279496: Test error = 0.9120

   C=0.572737686139748: Train error = 0.5080

   C=0.572737686139748: Test error = 0.3400

   C=0.8018327605956472: Train error = 0.5195

   C=0.8018327605956472: Test error = 0.3560

   (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=0.1145475372279496: Train error = 0.4920

   C=0.1145475372279496: Test error = 0.4100

   C=0.572737686139748: Train error = 0.5000

   C=0.572737686139748: Test error = 0.0760

C=0.8018327605956472: Train error = 0.4943

C=0.8018327605956472: Test error = 0.2520

(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?

The differences between the two learning rates show that the learning rate of $\gamma_t = \frac{\gamma_0}{1+t}$ returns better traing and test error on average than the other learning rate. The C value of $\frac{100}{873}$ is the worst for both learning rates, and the C value of $\frac{500}{873}$ was the best. I can conclude that the decaying learning rate approach is more effective and that the smallest C value is too small to capture the structure of the data, and the other is too large. This means the model becomes overfit on too large of C values.

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 using "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`. We recommend using SLSQP to incorporate the equality constraints. For Matlab, we recommend using the internal function "fmincon"; the document and examples are given at `https://www.mathworks.com/help/optim/ug/fmincon.html`. For R, we recommend using the "nloptr" package with detailed documentation at `https://cran.r-project.org/web/packages/nloptr/nloptr.pdf`.

(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? Note that if your code calculates the objective function with a double loop, the optimization can be quite slow. To accelerate, consider writing down the objective in terms of the matrix and vector operations, and treat the Lagrange multipliers that we want to optimize as a vector! Recall, we have discussed about it in our class.

(b) [15 points] Now, use Gaussian kernel in the dual form to implement the non-linear 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?