CS 5350/6350: Machine Learning Spring 2019

Homework 4

Handed out: 20 Mar, 2019
Due date: 11:59pm, 5 Apr, 2019

Author: Cade Parkison

1 Paper Problems [40 points]

1. [3 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) [1 point] What values ξ_i can take when the training example $\mathbf{x}_i$ breaks into the margin?

   The slack variable ξ_i is greater than zero when the training example $\mathbf{x}_i$ is inside the margin.

   (b) [1 point] What values ξ_i can take when the training example $\mathbf{x}_i$ stays on or outside the margin?

   The slack variable ξ_i is equal to zero when the training example $\mathbf{x}_i$ is inside the margin.

   (c) [1 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?

   This term allows the trade off between minimizing error and margin size. With this term, we can solve non-separable data at the cost of adding to the objective function for each example that is inside the margin. Without this term, the problem becomes a hard SVM and solutions would not be feasible for non-separable data.

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.)

In optimization, duality is a way to find bounds on an optimization problem. By augmenting the cost function, it is possible to turn a constrained problem into an unconstrained problem that gives lower bounds on the original constrained optimization problem. In the case of SVM, the dual problem solution is the same as the primal problem solution.

Primal:

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

The first step is to remove the constraints from the original primal problem and add them to the cost function with lagrangian multipliers α_i and β_i. We can also switch the order of the min-max.

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

This is now our Lagrangian function, and we can take it's partial derivatives and set to zero to solve the inner minimization problem.

$$\begin{aligned} \frac{\delta L}{\delta \mathbf{w}} &= \mathbf{0} \\ \frac{\delta L}{\delta b} &= 0 \\ \frac{\delta L}{\delta \xi_i} &= 0 \end{aligned}$$

Using these partial derivatives and simplifying, we get the following equations.

$$\begin{aligned} \mathbf{w} &= \sum_i \alpha_i y_i \mathbf{x}_i \\ \sum_i \alpha_i y_i &= 0 \\ \alpha_i + \beta_i &= C \end{aligned}$$

The next step is to replace $\mathbf{w}$ with the value above, and add the contraints above.

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

Finally, by constraining $\alpha_i \leq C$ we can remove β_i and add the constraints as the factors to minimize in the final form of the dual problem. Notice we changed the max to a min by negating the entire equation.

$$\min_{0 \geq \alpha_i \geq 0, \sum_i \alpha_i y_i = 0} \quad \frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j {\mathbf{x}_i}^T \mathbf{x}_j - \sum_i \alpha_i$$

3. [8 points] Continue with the dual form. Suppose after the training procedure, you have obtained the optimal parameters.

   (a) [2 points] What parameter values can indicate if an example stays outside the margin?

   If an example stays outside the margin, it will have a zero $alpha_i$ value.

   (b) [3 points] What are common and what are different when we use these parameters and the slack variables $\{\xi_i\}$ to determine the relevant positions of the training examples to the margin?

   For both the α_i and ξ_i parameters, when either are zero, we know the training example is outside the margin. However, the slack variable is not upper bounded, but α_i must be less than the regularization term, C.

   (c) [3 points] Now 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.

4. [3 points] How can we use the kernel trick to enable SVMs to perform nonlinear classification? What is the corresponding optimization problem?

5. [5 points] Prove that the primal objective function of the soft SVM is convex to $\mathbf{w}$ and b,

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

6. [2 points] Illustrate how the Occam's Razor principle is reflected in the SVM objective and optimization in Problem 5.

7. [3 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

8. [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 $\{\frac{1}{873}, \frac{10}{873}, \frac{50}{873}, \frac{100}{873}, \frac{300}{873}, \frac{500}{873}, \frac{700}{873}\}$. 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?

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 or CG 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 $\{0.01, 0.1, 0.5, 1, 2, 5, 10, 100\}$ 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 $\{0.01, 0.1, 0.5, 1, 2, 5, 10, 100\}$. List the training and test errors accordingly. Compared with the nonlinear SVM, what do you observe? what do you conclude and why?