CS 5350/6350: Machine Learning Fall 2021

Homework 4

Handed out: 4 Nov, 2021
Due date: 11:59pm, 19 Nov, 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 ξ_i can take when the training example $\mathbf{x}_i$ breaks into the margin?
Answer: When $\xi_i > 0$ is when the corresponding example can break into the margin

(b) [3 point] What values ξ_i can take when the training example $\mathbf{x}_i$ stays on or outside the margin?
Answer: When $\xi_i = 0$ is when the corresponding examples stays on/outside the margin.

(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 term because we want the ξ_i values to be small. When the values are small we will have less examples break the margin. If we get rid of the term then this cannot be accomplished.

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 soft SVM objective is defined as $\min_{\mathbf{w},\{\xi_i\}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$ where $\forall i, y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$. From here we can apply the formulation for lagrange multipliers and get our min max optimization problem:

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

We can then switch our min and max operations to get the dual form. To solve the inner minimization problem we need to find when the partial derivative with respect to the minimizing parameters is zero. $\frac{\partial L}{\partial \mathbf{w}} = 0$ when $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$, $\frac{\partial L}{\partial b} = 0$ when $\sum_i \alpha_i y_i = 0$, $\frac{\partial L}{\partial \xi_i} = 0$ when $\alpha_i + \beta_i = C$. We can plug these equalties back into our formula above:

$$\max_{\{\alpha_i \geq 0, \beta_i \geq 0\}} -\frac{1}{2}\sum_i\sum_j \alpha_i y_i \mathbf{x}_i \alpha_j y_j \mathbf{x}_j + \sum_i \alpha_i$$

with the contraints $\sum_i \alpha_i y_i = 0$ and $\alpha_i + \beta_i = C$. For the second constraint we can get rid of the β_i term by saying $0 \leq \alpha_i \leq C$ since the β_i term has to be ≥ 0. We can make this a minimization problem by simply multiplying the formula by -1. Giving us the final formulation:

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

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: When $\alpha_i = 0$ is when an example stays outside the margin i.e. not a support vector

(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: We see when going over the KKT condition we get the formulatons:

$$\forall i, \beta_i \xi_i = 0, \alpha_i(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i) = 0, \alpha_i + \beta_i = C$$

When $0 < \alpha < C$ then we know in this case $\beta > 0$ (because of the third equality) and $\therefore$ the slack variable $\xi_i = 0$ (because of the first equality) giving us $y(\mathbf{w}^T \mathbf{x}_i + b) = 1$ (because of the second equality). So our example will lie on the margin when $0 < \alpha_i < C$.

4. [6 points] How can we use the kernel trick to enable SVMs to perform nonlinear classification? What is the corresponding optimization problem?
Answer: We can map examples to a higher dimensional space and then represent the dot product of the mapped examples using a kernel method. Using the kernel trick we don't have to explicitly compute the dot product by simply just computing the kernel method. The corresponding optimization problem is therefore:

$$\min_{\{0 \le \alpha_i \le C\}, \sum_i \alpha_i y_i = 0} \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i \alpha_i$$

Where we simply replace the linear dot product from the orginal linear dual svm optimization problem with our kernel method.

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:
Our $\mathbf{w} = \mathbf{0}$ and $\gamma = 0.01$. $1 - y_0 \mathbf{w}^T \mathbf{x}_0 > 0 \therefore$

$$\nabla J^t = [\mathbf{w}_0; 0] - N * y_0 * \mathbf{x}_0 = [-1.5, 3., -0.9, -3.]$$

Our $\mathbf{w} = [0.015, -0.03, 0.009, 0.03]$ and $\gamma = 0.005$. $1 - y_1 \mathbf{w}^T \mathbf{x}_1 > 0 \therefore$

$$\nabla J^t = [\mathbf{w}_0; 0] - N * y_0 * \mathbf{x}_0 = [-2.985, -6.03, -5.991, 3.]$$

Our $\mathbf{w} = [0.029925, 0.00015, 0.038955, 0.015]$ and $\gamma = 0.0025$. $1 - y_2 \mathbf{w}^T \mathbf{x}_2 > 0 \therefore$

$$\nabla J^t = [\mathbf{w}_0; 0] - N * y_0 * \mathbf{x}_0 = [-4.470075, -0.59985, 7.538955, -3.]$$

$\mathbf{w} = [0.04110019, 0.00164963, 0.02010761, 0.0225]$

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.
   Answer: `https://github.com/nrtominaga/MachineLearning`

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 γ_0 and a to ensure convergence. For each setting of C, report your training and test error.
   Answer:
   Best a val: 5
   Best gamma val: 0.0001
   C = 100/873
   weights = [-0.50654929 -0.33600659 -0.33179845 -0.03240315 0.55920626]
   training error = 0.014908256880733946
   testing error = 0.012

C = 500/873
weights = [-2.42037002 -1.44766963 -1.78343885 -0.05381059 2.94387689]
training error = 0.008027522935779817
testing error = 0.012
C = 700/873
weights = [-3.52104517 -2.04909787 -2.25269316 0.09664708 4.12268757]
training error = 0.01261467889908257
testing error = 0.012

(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:
C = 100/873
weights = [-0.21062063 -0.11077495 -0.14367248 -0.01818523 0.20556991]
training error = 0.006880733944954129
testing error = 0.006
C = 500/873
weights = [-1.13311666 -0.59432375 -0.77315284 -0.09596924 1.10004728]
training error = 0.006880733944954129
testing error = 0.006
C = 700/873
weights = [-1.40889678 -0.74011976 -0.95410358 -0.12612295 1.35391865]
training error = 0.009174311926605505
testing error = 0.006

(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 weights between the learning rates did not seem too different but between different values of C we see the weights could be different by an order of 10. The second learning rate schedule seemed to have a much better training and testing error but there was not much difference between the values of C.

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?

Answer: The weights are much smaller in magnitude. I'm pretty sure this is because of the regularization term.

(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?

Answer: Please see the output from running my code. I had run into some bugs and had to make a fix last minute so all relevant information will be printed there (also it seemed like all the output would exceed the page limit). Sorry about this.

(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: Please see the output from my code

(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?