CS 5350/6350: Machine Learining Spring 2018

Homework 4

Handed out: 11 April, 2018
Due date: 23 April, 2018

General Instructions

- 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 discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. 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 10 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- The homework is due by **midnight of the due date**. Please submit the homework on Canvas.
- Some questions are marked **For 6350 students**. Students who are registered for CS 6350 should do these questions. Of course, if you are registered for CS 5350, you are welcome to do the question too, but you will not get any credit for it.

1 VC dimension

1. [25] Suppose we have a finite hypothesis space H

   (a) [10] Suppose $|H| = 2^{10}$. What is the VC dimension of H?
   $\text{VC}(H) <= \log_2(2^{10}) = 10$

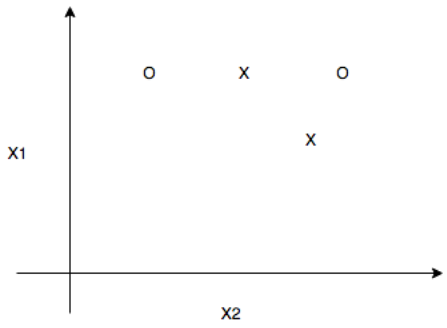
   (b) [15] Generally, for any finite H, what is $\text{VC}(H)$?
   $\text{VC}(H) <= \log_2(|H|)$

2. [25] Prove that linear classifiers in a plane cannot shatter any 4 distinct points.

   To show this we have to show that for any positioning of the points an adversarial can label the points in such a way that they cannot be shattered.

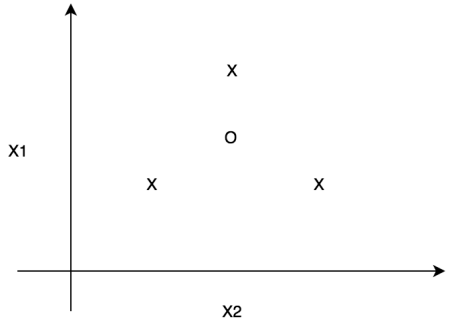
   In order to show we must show all the possible cases.

(a) For case 1 where 3 points are in a line, and the $4th$ point can be anywhere. In this case when three points are in a line no matter where the other point is there is always a labeling that will not shatter the positioning.

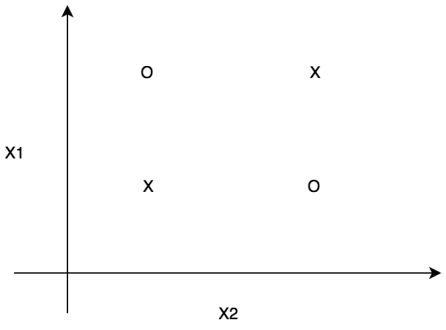


(b) Then there is the second case where 3 points are not in a line, and there is two other cases that fall into this category.

i. Where the $4th$ point is anywhere inside a triangle of the other 3 points.



ii. Where the $4th$ point is anywhere outside of the triangle



Since all of these are all the possible configurations we can have of the points then we can say conclude that lenear classifires in a plane cannot shatter any 4 distinct points.

3. [20] Consider our infinite hypothesis space H are all rectangles in a plain. Each rectangle corresponds to a classifier — all the points inside the rectangle are classified as positive, and otherwise classified as negative. What is VC(H)?
Infinity

2 Support Vector Machines

1. [20 points] Prove that the objective function of soft SVM is convex,

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

If we consider:
$\nabla J(w) = w$ If $\rightarrow \max(0, 1 - y_i w x_i) = 0$
$\nabla J(w) = w - C * N * y_i x_i$ If $\rightarrow otherwise$

If we take $\nabla^2 J(w)$ of both of these we get:

$\nabla^2 J(w) = 1$
$\nabla^2 J(w) = 1$

Hence since $\nabla^2 J(w) = 1 > 0$ then $J(w)$ is convex.

2. [10 points] Illustrate how the Occam's Razor's principle is reflected in SVM objective function and optimization.

The objective function includes the regularization penalty, which prevents the weights from getting too large. Occam's razor states that we must find the simplest hypothesis to explain the data, and with the SVM objective function it is measured in terms of **w**

3. [30 points] Suppose we have the following training dataset. 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

I used the following values for C, and N and bias b:
$C = 1$
$N = 3$
$b = 1$

$w \leftarrow (1-0.01)([0.0150, -0.0300, 0.0090, 0.0300]) + [0.0150, -0.0300, 0.0090, 0.0300] * 1 * 3 * 1 * [0.5000, -1.0000, 0.3000, 1.0000]$
$\nabla J^0 = [0.0000, 0.0000, 0.0000, 0.0000] - 1 * (3 * 1) = [0.5000, -1.0000, 0.3000, 1.0000]$
$w \leftarrow [0.0000, 0.0000, 0.0000, 0.0000]$

$w \leftarrow (1-0.01)([0.0449, 0.0303, 0.0689, -0.0003]) + [0.0449, 0.0303, 0.0689, -0.0003] * 1 * 3 * -1 * [-1.0000, -2.0000, -2.0000, 1.0000]$
$\nabla J^0 = [0.0150, -0.0300, 0.0090, 0.0300] - 1 * (3 * -1) = [-1.0000, -2.0000, -2.0000, 1.0000]$
$w \leftarrow [0.0150, -0.0300, 0.0090, 0.0300]$

$w \leftarrow (1-0.01)([0.0894, 0.0360, -0.0068, 0.0297]) + [0.0894, 0.0360, -0.0068, 0.0297] * 1 * 3 * 1 * [1.5000, 0.2000, -2.5000, 1.0000]$
$\nabla J^0 = [0.0599, 0.0003, 0.0779, 0.0297] - 1 * (3 * 1) = [1.5000, 0.2000, -2.5000, 1.0000]$
$w \leftarrow [0.0599, 0.0003, 0.0779, 0.0297]$

$w \leftarrow (1-0.005)([0.0965, 0.0208, -0.0022, 0.0446]) + [0.0965, 0.0208, -0.0022, 0.0446] * 1 * 3 * 1 * [0.5000, -1.0000, 0.3000, 1.0000]$
$\nabla J^1 = [0.1493, 0.0363, 0.0711, 0.0594] - 1 * (3 * 1) = [0.5000, -1.0000, 0.3000, 1.0000]$
$w \leftarrow [0.1493, 0.0363, 0.0711, 0.0594]$

$w \leftarrow (1-0.005)([0.1110, 0.0507, 0.0278, 0.0293]) + [0.1110, 0.0507, 0.0278, 0.0293] * 1 * 3 * -1 * [-1.0000, -2.0000, -2.0000, 1.0000]$
$\nabla J^1 = [0.2457, 0.0571, 0.0689, 0.1040] - 1 * (3 * -1) = [-1.0000, -2.0000, -2.0000, 1.0000]$
$w \leftarrow [0.2457, 0.0571, 0.0689, 0.1040]$

$w \leftarrow (1-0.005)([0.1329, 0.0535, -0.0099, 0.0442]) + [0.1329, 0.0535, -0.0099, 0.0442] * 1 * 3 * 1 * [1.5000, 0.2000, -2.5000, 1.0000]$
$\nabla J^1 = [0.3567, 0.1078, 0.0967, 0.1333] - 1 * (3 * 1) = [1.5000, 0.2000, -2.5000, 1.0000]$
$w \leftarrow [0.3567, 0.1078, 0.0967, 0.1333]$

$w \leftarrow (1-0.0025)([0.1363, 0.0458, -0.0076, 0.0516]) + [0.1363, 0.0458, -0.0076, 0.0516] * 1 * 3 * 1 * [0.5000, -1.0000, 0.3000, 1.0000]$
$\nabla J^2 = [0.4896, 0.1613, 0.0868, 0.1775] - 1 * (3 * 1) = [0.5000, -1.0000, 0.3000, 1.0000]$
$w \leftarrow [0.4896, 0.1613, 0.0868, 0.1775]$

$w \leftarrow (1-0.0025)([0.1435, 0.0607, 0.0074, 0.0439]) + [0.1435, 0.0607, 0.0074, 0.0439] * 1 * 3 * -1 * [-1.0000, -2.0000, -2.0000, 1.0000]$
$\nabla J^2 = [0.6259, 0.2071, 0.0792, 0.2290] - 1 * (3 * -1) = [-1.0000, -2.0000, -2.0000, 1.0000]$
$w \leftarrow [0.6259, 0.2071, 0.0792, 0.2290]$

$w \leftarrow (1-0.0025)([0.1431, 0.0606, 0.0074, 0.0438])$
$\nabla J^2 = [0.1431, 0.0606, 0.0074, 0.0438]$

$w \leftarrow [0.1431, 0.0606, 0.0074, 0.0438]$

Final w: $[0.1431, 0.0606, 0.0074, 0.0438]$

3 Programming Assignments

1. We will implement SVM with stochastic sub-gradient descent. We will reuse the dataset for Perceptron implementation. 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 testing datasets, feature values and labels are separated by commas. Set the maximum epochs T to 50. 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 $\{\frac{10}{873}, \frac{100}{873}, \frac{300}{873}, \frac{500}{873,}\frac{700}{873}\}$. Don't forget to convert the labels to be in $\{1, -1\}$.

   (a) [50 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 test error.

   $C = \frac{10}{873}$
   Accuracy testing: 0.9639278557114228
   Test error = 0.03607214428857719

   $C = \frac{100}{873}$
   Accuracy testing: 0.9879759519038076
   Test error = 0.012024048096192397

   $C = \frac{300}{873}$
   Accuracy testing: 0.9879759519038076
   Test error = 0.012024048096192397

   $C = \frac{500}{873}$
   Accuracy testing: 0.9879759519038076
   Test error = 0.012024048096192397

   $C = \frac{700}{873}$
   Accuracy testing: 0.9859719438877755
   Test error = 0.014028056112224463

   (b) [50 points] Use the schedule $\gamma_t = \frac{\gamma_0}{1+t}$. Report the test error for each setting of C.

$C = \frac{10}{873}$
Accuracy testing: 0.9639278557114228
Test error = 0.03607214428857719

$C = \frac{100}{873}$
Accuracy testing: 0.9879759519038076
Test error = 0.012024048096192397

$C = \frac{300}{873}$
Accuracy testing: 0.9879759519038076
Test error = 0.012024048096192397

$C = \frac{500}{873}$
Accuracy testing: 0.9879759519038076
Test error = 0.012024048096192397

$C = \frac{700}{873}$
Accuracy testing: 0.9859719438877755
Test error = 0.014028056112224463

(c) [20 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 test errors. What can you conclude?

$C = \frac{10}{873}$
Model Parameters: $\frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$ and $\frac{\gamma_0}{1+t}$
$[-25288.5887076, -15063.8240872, -16022.5625078, -2782.24411486, 27386.1415048]$
$[-21325.0217736, -10889.112699, -10492.8624735, -3512.49220633, 8911.99304415]$

$\frac{\gamma_0}{1+\frac{\gamma_0}{d}t} = 0.987975951904$
$\frac{\gamma_0}{1+t} = 0.963927855711$

$C = \frac{100}{873}$
Model Parameters: $\frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$ and $\frac{\gamma_0}{1+t}$
$[-52441.4926675, -36188.5609245, -38209.6769276, -10145.7203947, 58579.8787385]$
$[-32492.0416935, -21117.4124267, -21532.6932631, -5031.71006657, 32825.5843357]$

$\frac{\gamma_0}{1+\frac{\gamma_0}{d}t} = 0.98997995992$
$\frac{\gamma_0}{1+t} = 0.987975951904$

$C = \frac{300}{873}$
Model Parameters: $\frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$ and $\frac{\gamma_0}{1+t}$

$[-110346.392081, -75928.2314648, -78133.2954785, -27554.3258578, 104358.914249]$
$[-45287.1509343, -30057.5965674, -32806.9801795, -7789.72628667, 43844.6231866]$

$\frac{\gamma_0}{1+\frac{\gamma_0}{d}t} = 0.985971943888$
$\frac{\gamma_0}{1+t} = 0.987975951904$

$C = \frac{500}{873}$
Model Parameters: $\frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$ and $\frac{\gamma_0}{1+t}$
$[-158098.660428, -112712.633892, -115071.502339, -50226.2395035, 143474.365839]$
$[-60703.5318481, -39702.8592759, -44171.35262, -12975.570726, 52167.76998]$

$\frac{\gamma_0}{1+\frac{\gamma_0}{d}t} = 0.985971943888$
$\frac{\gamma_0}{1+t} = 0.987975951904$

$C = \frac{700}{873}$
Model Parameters: $\frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$ and $\frac{\gamma_0}{1+t}$
$[-209124.451871, -144252.562456, -149288.400186, -59164.1727278, 185618.815655]$
$[-78860.6395292, -52367.748611, -57456.4070289, -19013.9104801, 61452.5135766]$

$\frac{\gamma_0}{1+\frac{\gamma_0}{d}t} = 0.985971943888$
$\frac{\gamma_0}{1+t} = 0.985971943888$

When C is smaller the accuracy seemed to be less accurate, but as it got higher it did better. We know that $C = infinity$ overfits the data, meaning that in our case when C got bigger it fit the data better.