ECE523: Engineering Applications of Machine Learning and Data Analytics

I acknowledge that this exam is solely my effort. I have done this work by myself. I have not consulted with others about this exam in any way. I have not received outside aid (outside of my own brain) on this exam. I understand that violation of these rules contradicts the class policy on academic integrity.

Name: Solution

Signature: ____________________

Date: ____________________

Instructions: There are four problems. You have 50 minutes to complete the exam. Partial credit is given for answers that are partially correct. No credit is given for answers that are wrong or illegible. Write neatly.

Problem 1: __________

Problem 2: __________

Problem 3: __________

Problem 4: __________

Total: __________

Problem #1 – Support Vector Machines (10 Points)

Using a kernel function is equivalent to mapping data into a higher dimensional space then taking the linear dot product in that space. (1) Please explain what is the advantage of using the kernel function compared to doing the explicit mapping. (2) Given that we have an expression for $\mathbf{w}$, do we still need to explicitly compute the mapping at the time of classification? Why or why not? (3) What is a support vector? Feel free to support your conclusions with equations.

Solution

(1) The main advantage of the kernel trick is the reducing computational complexity that avoid computing the maps into feature space. We are able to compute dot products between extremely high dimensional vectors without explicitly calculating the high dimensional vectors, rather, we work with the feature vectors in the original space. Furthermore, we are also able to solve non-linear problems.

(2)The output of the SVM is given by $g(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + b$; however, given that we are using kernels there is a projection of $\Phi(\mathbf{w})$ and $\Phi(\mathbf{x})$. Using this information and knowing the form of $\mathbf{w}$, we have:

$$\begin{aligned}
g(\mathbf{x}) &= \Phi(\mathbf{w})^{\mathsf{T}}\Phi(\mathbf{x}) + b \\
&= \left(\sum_{j\in SV} \alpha_j y_j \Phi(\mathbf{x}_j)\right)^{\mathsf{T}} \Phi(\mathbf{x}) + b \\
&= \sum_{j\in SV} \alpha_j y_j \Phi(\mathbf{x}_j)^{\mathsf{T}}\Phi(\mathbf{x}) + b \\
&= \sum_{j\in SV} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}) + b
\end{aligned}$$

which shows that we never need to explicitly compute Φ and the kernel still works! That is the output of the SVM is the results of linear combinations of the kernel computer w.r.t. the support vectors and the data we seek to classify.

(3) Finally a support vector, as shown in the last equation are the vectors that have a non-zero α_j. Another acceptable answer would be to say that the support vectors are the training data that fall within the margin.

Problem #2 – AdaBoost (15 Points)

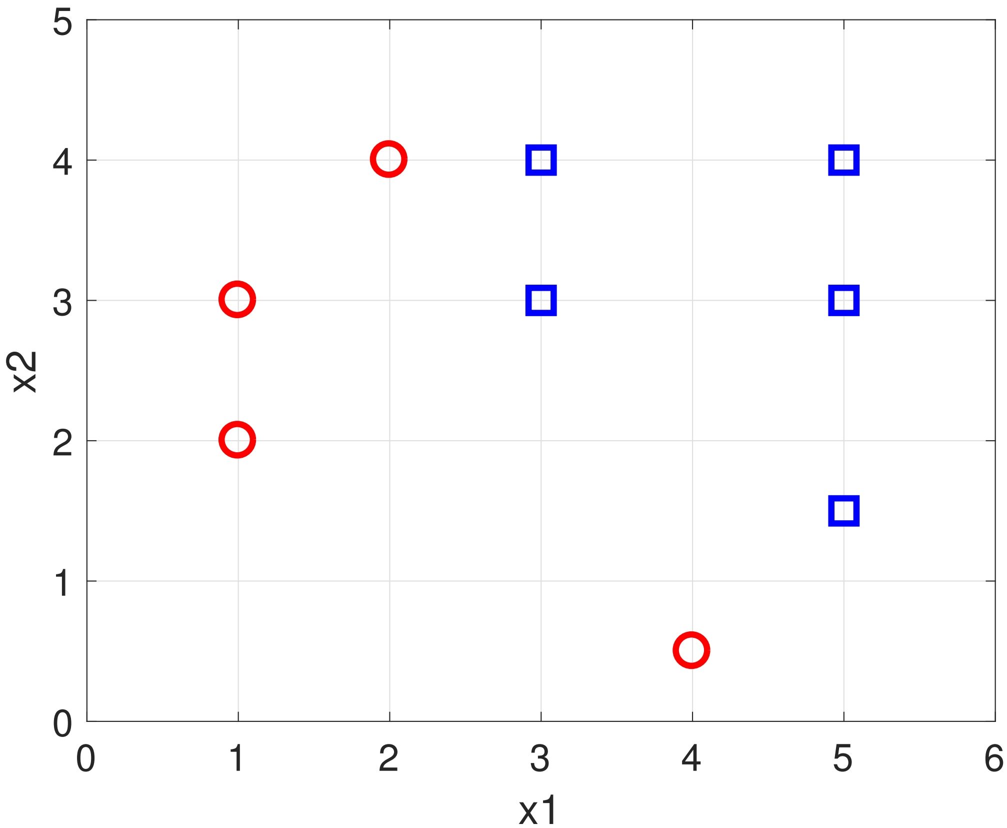

Figure 1: Labeled training points for Problem 2.

Consider the labeled training points in Figure 1, where ○ and □ denote positive and negative labels, respectively. We wish to apply AdaBoost with a threshold classifier (i.e., pick an axis then pick a threshold to label the data). In each boosting iteration, we select the threshold that minimizes the weighted training error, breaking ties arbitrarily. Use the AdaBoost pseudo-code to help with this question

1. In Figure 1, draw a decision boundary on x_1-axis (i.e., vertical line) corresponding to the first threshold that the boosting algorithm could choose. Label this boundary (1), and also indicate +/- side of the decision boundary. Solution: See Figure 2(a). The first line drawn is the one that will follow the minimization of error on $\mathcal{D}_1$ since we only make one mistake. If we were to start off with (2)'s threshold then we would have made two mistakes.

2. In the same figure also circle the point(s) that have the highest weight after the first boosting iteration. Solution: This red circle that is misclassified by one is circled.

3. What is the weighted error of the first threshold after the first boosting iteration, i.e., after the points have been re-weighted? Solution: The error is 0.5. We proved this result in class.

4. Draw a decision boundary corresponding to the second threshold using the weights, again in Figure 1, and label it with (2), also indicating the +/- side of the boundary. For clarity grading exams draw a decision boundary on x_1 (i.e., vertical line). Solution: Misclassifying the point that is now circled comes with an error of 0.5, therefore, we need to classify it correctly. Placing the line to the right of this data point will classify it correctly, which means that we misclassify two of the blue squares, which have a low misclassification rate.

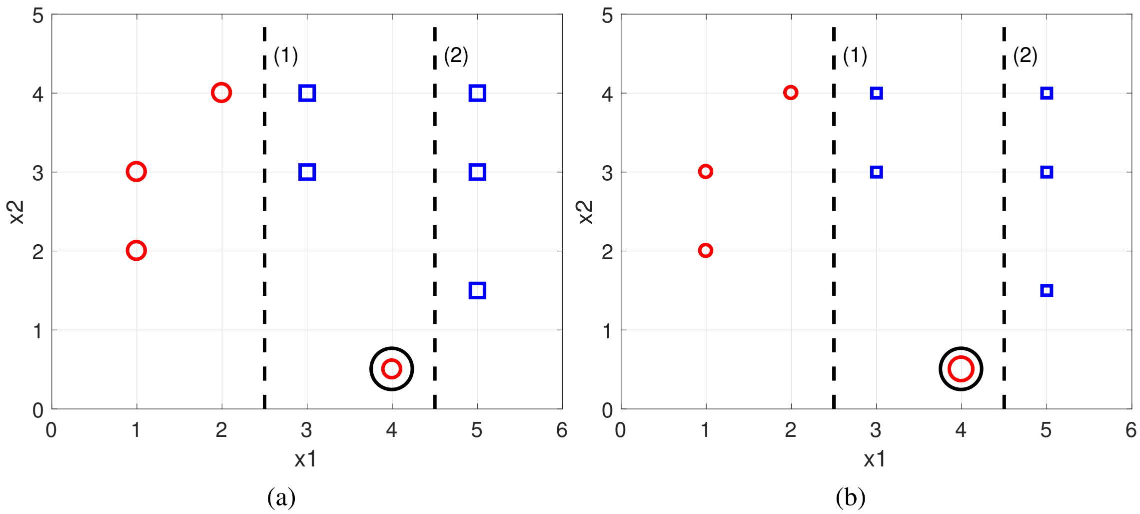

Figure 2: Solution to Problem 2.

Problem #3 – Random Short Answer (15 Points)

(SA:1) AdaBoost has a weight for every sample in the training data set, $\mathcal{D}_t(i)$ (where t is the learning round on the ith instance), and we said that this weight: (a) provided a measure of the relative difficulty to classify the point; and (b) was used to build the next classifier h_{t+1}. Describe two methods how these weights can be used to train h_{t+1}.

Solution

- *Optimization*: Incorporate the weights $\mathcal{D}_t(i)$ into the cost function of the weak learning algorithm.
- *Sampling*: Randomly sample from the distribution $\mathcal{D}_t$ to generate a new training data set. Data with higher cost have a larger probability to be used in the "new" training data set.

(SA:2) What is an appropriate way to train a deep neural network? The key word in that sentence is "appropriate".

Solution

The appropriate way to train a deep neural network is to use unsupervised pre-training (i.e., with an autoencoder or RBM) then apply backpropagation.

(SA:3) AdaBoost requires that the error of a weak classifier is at least better than 50%. (1) Why is this requirement needed? (2) What are some of the issues associated with requiring 50% accuracy on any arbitrary data set? Recall that

$$\varepsilon_t = \sum_{i=1}^{n} \mathcal{D}_t(i) \mathbb{1}_{h_t(\mathbf{x}_i) \neq y_i}$$

where $\mathbb{1}$ is the indicator function (i.e., $\mathbb{1}_{\text{expression}}$ is 1 if the expression is true otherwise it is zero).

Solution (1) Acceptable responses include those that related to: (a) an assumption that was made about our weak learning algorithm, (b) the Condorcet jury theorem can be used to show the error needs to be less that 0.5 to make boost the ensemble performance, or (c) the weights for the classifiers will be negative if their error is greater than 0.5.

(2) This requirement could be quite difficult to achieve if we have a large number of classes.

(SA:4) In the context of multi-armed bandits, what is regret?

Solution

The regret is defined as the different between the optimal selection strategy select an arm and the strategy we are using.

Problem #4 – True/False: A Gamblers Ruin (10 Points)

[True/False] (1 point): In class, we derived the classifier weights for AdaBoost then in a later lecture, we derived a set of Bayes optimal weights for a generic ensemble of classifiers, which resulted in the same way to calculate classifier voting weights. To quite different approaches and the same conclusion!

Solution: This is not a true statement. Recall that the Bayes optimal weights have a term $\log P(\omega_j)$, which does not appear in the AdaBoost weights.

[True/False] (1 point): Bagging works because it assumes the base models have low variance and high bias. That is we do not need base classifiers that are unstable.

Solution: Bagging works well with unstable classifiers that are low bias and high variance.

[True/False] (1 point): The multi-armed bandit address problems that require exploration of new arms and exploitation of the ones we know perform well.

[True/False] (1 point): The weight correction expression for backpropagation in a neural network are of the same form for both hidden and output nodes. The only difference is how the local gradient is determined.

[True/False] (1 point): One of the disadvantages of deep learning with auto-encoders is that we need a large volume of labeled data to train each layer.

Solution: You do not need labeled data to train an autoencoder! Therefore, you can use as much data available to you with and without labels.

[True/False] (1 point): The first implementation of passive aggressive online learning updates the weights of a linear model at every time step; however, the update is much more aggressive if the model made a mistake.

Solution: The PA algorithm updated the weights only if there was a mistake made, and not if there was a correct classification. Hence the passive in the name of the algorithm.

[True/False] (1 point): A neural network will (likely) find a local minimum for its optimization problem and the same is true for a support vector machine.

Solution: A neural network will (likely) find a local minimum, but the SVM problem is convex and we'll find a globally optimal solution.

[True/False] (1 point): AdaBoost works in successive rounds by having classifiers focus more on the data points that have been misclassified by previous classifiers.

[True/False] (0 point): To the student who answered False last time on the question about free points: No more free points!

Solution:

[True/False] (1 point): The theory behind AdaBoost proves that the error on the testing data is upper bouned by

$$\widehat{\text{err}}(H) \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)}$$

Solution: The error bound is for the training data.

[Accept/Reject] (1 point): "I decided to use the UCB1 multi-armed bandit algorithm for the task of selecting buyers in stock. We felt this is the best selection since the problem relies on modeling consumer interests, which we assumed to be a stationary problem."

Solution: If this problem is not stationary then the UCB1 algorithm is not going to be ideal.

Cheat Sheet

Algorithm 1 Adaboost (Adaptive Boosting)

Input: $\mathcal{S} := \{x_i, y_i\}_{i=1}^n$, learning rounds T, and hypothesis class $\mathcal{H}$

Initialize: $\mathcal{D}_1(i) = 1/n$

1: **for** $t = 1, \ldots, T$ **do**
2: $\quad h_t = \arg\min_{h\in\mathcal{H}} \widehat{\textbf{err}}(h, \mathcal{S}, \mathcal{D}_t)$
3: $\quad \epsilon_t = \sum \mathcal{D}_t(i) \mathbb{1}_{h(\mathbf{x}_i)\neq y_i}$
4: $\quad \alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
5: $\quad \mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t}\exp\left(-\alpha_t y_i h_t(x_i)\right)$
6: **end for**
7: **Output:** $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$