

## CS534 — Midterm

**Name:**

1. (VC-dimension)

- a. (5pt) Give the definition of  $VC(H)$ , which is the VC-dimension of hypothesis space  $H$ ? (Make sure that you also define any terms that are used as part of the definition.)

**Solution:**  $VC(H)$  is the maximum cardinality of any set of instances that can be shattered by  $H$ . We say that  $H$  shatters a set of points if and only if it can assign any possible labeling to those points.

- b. (7pt) Prove that for any finite hypothesis space  $H$  that  $VC(H) \leq \log_2 |H|$ .

**Solution:** For any set of distinct points  $S$  of size  $m$ , there are  $2^m$  distinct ways of labeling those points. This means that for  $H$  to shatter  $S$  it must contain at least  $2^m$  distinct hypotheses. This tells us that if the VC dimension of  $H$  is  $m$  then we must have  $2^m$  hypotheses, i.e.  $2^m \leq |H|$  or equivalently that  $m = VC(H) \leq \log_2 |H|$ .

- c. (8pt) Consider a domain with  $n$  binary features and binary class labels. Let  $H_d$  be the hypothesis space that contains all decision trees over those features that have depth no greater than  $d$ . (The depth of a decision tree is the depth of the deepest leaf node.) What is the VC-dimension of  $H_d$ ? Prove your answer.

**Solution:** The original intention was to also have the constraint that  $d \geq n$ , that is depth bound is at least as large as the number of features, so in effect we can grow the maximally discriminative tree if desired. I will give a solution for the  $d \geq n$  case, which is what most everyone appeared to be assuming anyway. Handling the general case where  $d \leq n$  is quite tricky. I gave full credit for a correct upper or lower bound for the  $d \leq n$  case.

First note that any tree in  $H_d$  can be represented by a tree of exactly depth  $d$  in  $H_d$ . So we will restrict our attention to trees of exactly depth  $d$ . All of these trees have  $2^d$  leaf nodes. Also note that there are a total of  $2^n$  examples in our instance space, which gives us an immediate upper bound on the VC-dimension of  $H_d$ , i.e.  $VC(H_d) \leq 2^n$ .

To get a lower bound let  $S$  contain the set of all possible  $2^n$  instances. Since we have that  $d \geq n$  it is straightforward to create a tree of depth  $n$  with a leaf node for each example and furthermore we can label the leaf nodes in all possible ways. This shows that we can shatter the set  $S$  with  $H_d$ , which implies that  $VC(H_d) \geq 2^n$ . Combining the upper and lower bound tell us that  $VC(H_d) = 2^n$ .

2. (Perceptron)

- a. (5pt) In class we derived the perceptron algorithm as gradient descent on the hinge loss objective function given by

$$J(w) = \frac{1}{N} \sum_{i=1}^N \max(0, -y_i w \cdot x_i)$$

where  $x_i$  is the input feature vector for the  $i$ 'th training instance and  $y_i$  is the label for the  $i$ 'th instance. We also consider using an alternative loss function known as the 0/1 loss given by,

$$J(w) = \frac{1}{N} \sum_{i=1}^N L(\text{sgn}(w \cdot x_i), y_i)$$

where  $L(y, y') = 1$  if  $y \neq y'$  and 0 otherwise.

What was the reason for using the hinge loss instead of 0/1 loss?

**Solution:** For 0/1 loss the surface of  $J(w)$  is divided into regions that are piecewise constant that are connected via discontinuities. The gradient of this surface then is either zero or undefined, which means that gradient descent will be a useless approach for minimizing  $J(w)$ . The hinge loss rather scales according to the functional margin and  $J(w)$  ends up being piecewise linear and convex with a minimum valued region containing all weight vectors with zero training error. Performing gradient descent on this  $J(w)$  is useful and will lead to the minimum region.

- b. (6pt) Consider two different weight vectors  $w$  and  $w'$  that are consistent with the training data where  $w$  has a large margin and  $w'$  has a very small margin. Which weight vector will achieve a better hinge loss? Explain.

**Solution:** Both weight vectors will have a hinge loss of zero. This shows a potentially undesirable aspect of the perceptron algorithm in that it does not care about the margin of weight vectors on examples that are being correctly prediction. Thus, the perceptron algorithm is just as happy with a weight vector that is “just barely” consistent with the data as with a weight vector that has a large margin.

- c. (9pt) Derive the gradient of the hinge loss objective  $\nabla_w J(w)$ .

**Solution:** See class notes on the perceptron algorithm.

3. (Support Vector Machines)

- a. (6pt) In class we introduced two notions of margin: the *functional margin* and the *geometric margin*. When deriving the SVM optimization problem why did we choose to maximize the geometric margin rather than the functional margin? Specifically, what would go wrong if we simply tried to maximize the functional margin and how does maximizing the geometric margin correct this problem?

**Solution:** The functional margin of  $w$  for a training example is  $y_i x_i \cdot w$ . From this we see that the problem of maximizing the functional margin is ill-conditioned as one can make the value arbitrarily large by simply scaling  $w$  by a constant. Rather the geometric margin can be interpreted as the functional margin but normalized by the norm of the weight vector  $y_i x_i \cdot \frac{w}{\|w\|}$  and thus scaling  $w$  by a constant does not effect the geometric margin.

- b. (7pt) Consider the following constrained optimization problem for finding the maximum margin classifier:

$$\begin{aligned} & \max_{w,b,\gamma} \gamma \\ \text{subject to: } & y^i \frac{(w \cdot x^i + b)}{\|w\|} \geq \gamma, \quad i = 1, \dots, N \end{aligned}$$

Prove that this problem is equivalent to the following optimization problem solved by SVMs. Make sure that you explain each step.

$$\begin{aligned} & \min_{w,b} \|w\|^2 \\ \text{subject to: } & y^i (w \cdot x^i + b) \geq 1, \quad i = 1, \dots, N \end{aligned}$$

**Solution:** This derivation is in the SVM slides.

- c. (7pt) Consider the objective function that the softmargin SVM tries to minimize:

$$\|w\|^2 + C \sum_i \xi_i$$

Suppose that we have a linearly separable training set. Is it true that the solution weight vector  $w$  will achieve zero training error. If true then explain why. If false then show a 1-d linearly separable training set where you would expect the training error will be non-zero, explain.

**Solution:** The solution will not necessarily be consistent with the training data. This is because the optimization problem calls for a trade-off between maximizing the margin (i.e. minimizing  $\|w\|$ ) and minimizing the slack variables, which become larger when there are margin violation. So if it is possible to dramatically decrease  $\|w\|$  by having a small number of errors (or margin violations) then such a weight vector will be returned.

As an example, consider the following training set:

$$\{(x_1 = -1, y_1 = -1), (x_2 = -0.8, y_1 = 1), (x_3 = 1, y_3 = 1)\}$$

It should be clear that we can find a consistent solution for this problem, in particular a decision boundary at -0.9 classifies all of the data correctly. However, we can show that a

weight vector that corresponds to a decision boundary at 0 will achieve a better objective for moderate values of  $C$ . To see this, first consider solving for the “hard” maximum margin solution. This corresponds to the case when all of the  $\eta_i = 0$ .

For this case, the support vectors will be  $x_1 = -1$  and  $x_2 = -0.8$ . We know that the constraints for the support vectors will be tight leading to the following equations: (after some simplification of the support vector constraints from the class notes)

$$\begin{aligned}w - b &= 1 \\ -0.8w + b &= 1\end{aligned}$$

The solution weight vector is  $w = 40$ . Since  $\eta_i = 0$  for all  $i$  we have that the objective is  $40^2 = 160$  for any value of  $C$ .

Now consider a solution corresponding to a decision boundary at 0. Here we will select the  $w$  so that the  $x_1 = -1$  and  $x_3 = 1$  are the support vectors. If we go through the same process as above we get that  $w = 1$ . In this case we have that  $\eta_1 = \eta_3 = 0$ , but that  $\eta_2 = 1.8$ . The objective function then is  $1^2 + 1.8C = 1 + 1.8C$ . Clearly for a wide range of  $C$  values this is less than 160, showing that there is at least one solution that has a better objective value than the consistent maximum margin solution.

4. a. (4pt) Consider a training set whose labels were randomly corrupted. For the  $k$ -nearest neighbor classifier, which of the following choices of  $k$  is more robust to the labeling noise:  $k=1$  and  $k=4$ ? Explain

**Solution:** We would expect that  $k = 4$  would be more robust. This is because given 4 nearest neighbors the probability that the decision would be changed by random label noise is less than with just a single nearest neighbor. In particular, the random noise would need to conspire to change the label of a majority of the neighbors in a way that disagrees with the non-noise prediction.

- b. (4pt) Consider a training set that has a relatively large number of completely random features, that is, features that have no correlation with the class label. Which algorithm would you choose for this data set  $k$ -nearest neighbor or a decision tree learner? Explain.

**Solution:** A decision-tree algorithm would be preferable. Nearest neighbor algorithms, even for  $k > 1$  are very sensitive to the addition of random features. The reason is that for moderate numbers of such features they end up dominating the distance computation and all data points end up looking equally far apart. The set of  $k$  nearest neighbors then becomes largely arbitrary. Rather a decision tree algorithm at least attempts to select features at internal nodes that are correlated with the class labels. This ideally helps avoid including the random features in the final model and will result in a more robust model.

- c. (4pt) Why is it a good idea to initialize the weights of a neural network to be close to zero?

**Solution:** Initializing the weights close to zero starts the sigmoid units off in their linear region, which makes it easier to adjust their responses based on the error feedback. If the nodes start out in the saturated regions, the gradients are very small and it can take many iterations to significantly change the response of the network.

- d. (4pt) Why might it be a bad idea to initialize all weights of a neural network to be zero?

**Solution:** If we initialize all of the weights to be exactly zero then the weight adjustments via back propagation will be identical for all hidden units. Hence all of the hidden units will behave identically and it will be as if there is only one hidden unit. Thus, it is better to initialize the weights to small random values, which will result in hidden units that uncover different structure in the data.

- e. (4pt) Consider learning a predictor for whether someone has cancer or not. If the predictor predicts cancer then the subject will be sent to a doctor, otherwise the subject will be sent home. Write down a loss function  $L(\hat{y}, y)$  that you think is appropriate for this task. Explain.

**Solution:** Assuming that life vs. death is the primary driving force it would be preferable to have a classifier with a very small false negative error rate. That is, the likelihood of predicting “not cancer” when the patient actually has cancer should be very small. To achieve this behavior, one would set the loss for predicting “not cancer” when the true label is “cancer” to a large value. The relative cost of a false positive, i.e. predicting cancer when the patient does not have cancer, would be much lower and might be based on the cost in dollars that must be absorbed by the patient and system and might depend on the patient.

5. (PAC Learning)

- a. (10pt) In class we showed that if a consistent learning algorithm for a finite hypothesis space  $H$  is provided with

$$m \geq \frac{1}{\epsilon} \left( \ln |H| + \ln \frac{1}{\delta} \right)$$

randomly drawn training instances, then we can state a certain guarantee. What is that guarantee? Make sure to clearly indicate the roles of  $\epsilon$  and  $\delta$ .

**Solution:** The guarantee is that with at least  $1 - \delta$  probability over draws of training sets of size  $m$ , any consistent hypothesis will have true error no more than  $\epsilon$ .

- b. (10pt) Consider the hypothesis space  $H$  of decision stumps for an input space containing  $n$  binary features. That is, each hypothesis is a decision tree that contains exactly one binary test. Prove that  $H$  is PAC-learnable. You may use the inequality from part a if you would like.

**Solution:** We can show that the consistent learner is a PAC-learning algorithm for decision stumps. First, note that the number of decision stumps is  $4n$ , i.e. one stump for each possible root test and way of labeling the leaves. Note that many of these stumps will represent the same underlying function (in particular the stumps where both leaves are either positive or negative). The above bound tells us that a training set of size

$$m \geq \frac{1}{\epsilon} \left( \ln(4n) + \ln \frac{1}{\delta} \right)$$

is sufficient to guarantee the accuracy and reliability requirements of PAC learnability. We also argue that we can find a consistent decision stump when one exists in polynomial time in the size of the training set, which we see from above needs only be polynomially large. The algorithm is trivial. Simply test each of the  $4n$  decision stumps to see if one of them is consistent. The total running time is then

$$O(nm) = O\left(\frac{n}{\epsilon} \left[ \ln(4n) + \ln \frac{1}{\delta} \right]\right)$$

which meets the polynomially runtime requirements of PAC learnability.