Midterm Solutions

1. **[6 points]** TRUE or FALSE: for any two neighboring Voronoi cells C_1 and C_2 corresponding to two training examples $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, the boundary between C_1 and C_2 is part of the decision boundary of the 1-nearest-neighbor classifer. Briefly justify your answer.

   FALSE. If $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ have the same label, then all the points in their Voronoi cells will also be assigned to that label. Hence, the boundary between the cells will not be part of the decision boundary.

   Marking: 6 points for a correct answer and correct justification. Saying FALSE but providing an incorrect justification gets 2 points. Saying TRUE but giving an explanation that conveys some insight gets 2 points.

   Mean: 3.6/6

2. **[6 points]** In lecture, we originally introduced binary linear classifiers in terms of a linear function followed by a threshold:

$$z = \mathbf{w}^T\mathbf{x} + b$$
$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

   We saw that it is safe to assume the threshold $r = 0$. Briefly explain why we may assume this without limiting the expressive power of the model.

   The threshold can be absorbed into the bias term. Specifically, we set $\tilde{b} = b - r$. Then $\mathbf{w}^\top\mathbf{x} + \tilde{b} \geq 0$ iff $\mathbf{w}^\top\mathbf{x} + bias \geq r$.

   Marking: Most people got full marks. 2 points for answers that argue that you can "scale" the weights.

   Mean: 5.4/6

3. **[6 points]** Give one example of a situation where you would prefer to use L^2 regularization rather than L^1. Briefly justify your answer.

   L^1 regularization encourages sparse weight vectors, while L^2 encourages all the weights to be small but nonzero. We'd prefer L^2 in situations where all the features are likely to be relevant. Examples could include polynomial regression, classifying MNIST digits from pixels, etc.

   Marking: 6 points for contrasting L^2 and L^1 regularization and explaining any reasonable situation. An otherwise correct answer which is missing a situation gets 4 points. Pointing out that L^2 penalizes large weights more, without a good explanation of why this is helpful, gets 4 points.

   Mean: 3.8/6

4. **[12 points]** Carol and Dave are each trying to predict stock prices using neural networks. They formulate this as a regression problem using squared error loss. Carol trains a single multilayer perceptron on a certain training set and uses its predictions on the test set. Dave trains 5 different networks (using exactly the same architecture, training data, etc. as Carol) starting with different random initializations, and averages their predictions on the test set.

   For each of the following questions, please briefly and informally justify your answer. You do not need to provide a mathematical proof.

(a) **[4 points]** Compared with Carol's approach, is the Bayes error for Dave's approach HIGHER, LOWER, or THE SAME?

THE SAME. The Bayes error is a property of the data generating distribution, and doesn't depend on the algorithm that was used.

(b) **[4 points]** Compared with Carol's approach, is the bias for Dave's approach HIGHER, LOWER, or THE SAME?

THE SAME. Sampling multiple hypotheses from the same distribution and averaging their predictions doesn't change the expected predictions due to linearity of expectation. Hence it doesn't change the bias.

(c) **[4 points]** Compared with Carol's approach, is the variance for Dave's approach HIGHER, LOWER, or THE SAME?

LOWER. Averaging over multiple samples reduces the variance of the predictions, even if those samples are not fully independent. (In this case, they're not fully independent as they share the same training set.)

Marking: each part was worth 2 points for the correct answer and 2 points for the explanation. For part (b), mentioning linearity of expectation or some other argument involving expected value was required for full marks. Note that the procedure here isn't bagging, so answers involving bagging lost some points.

Mean: 9.3/12

5. **[8 points]** In lecture, we considered using linear regression for binary classification. Here, we use a linear model

$$y = \mathbf{w}^\top \mathbf{x} + b$$

and squared error loss $\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2$. We saw this has the problem that it penalizes confident correct predictions. Will it fix this problem if we instead use absolute value loss, $\mathcal{L}(y, t) = |y - t|$? Justify your answer mathematically.

This will not fix the problem. If the target is $t = 1$, then a prediction of $y = 10$ will still have a higher loss than a prediction of $y = 1$.

Marking: 2 points for the correct answer, 6 points for the explanation. Valid explanations include a counterexample or an argument about the derivative of the loss. Saying it completely solves the problem, but giving an otherwise correct explanation, gets 6 out of 8.

Mean: 7.3/8

6. **[10 points]** Recall the soft-margin SVM optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \ & \frac{1}{2}\|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \ & t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 - \xi_i \qquad i = 1, \ldots, N \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, \ldots, N \end{aligned}$$

What will go wrong if we eliminate the $\frac{1}{2}\|\mathbf{w}\|_2^2$ term from the cost function? Justify your answer mathematically.

If we eliminate the $\frac{1}{2}\|\mathbf{w}\|_2^2$, then the cost function will no longer encourage a large margin. Suppose that $(\mathbf{w}, b)$ correctly classifies all the data. We can replace it with $\tilde{\mathbf{w}} = \alpha\mathbf{w}$ and

$\tilde{b} = \alpha b$ for large α, and it will automatically satisfy all the margin constraints with all the slack variables being 0. Hence, it achieves the minimum possible cost, namely 0.

Mean: 3.0/10

7. **[12 points]** When we analyzed KNN, we assumed the training examples were sampled densely enough so that the true conditional probability $p(t \mid \mathbf{x})$ is approximately constant in the vicinity of a query point $\mathbf{x}_\star$. Suppose it is a binary classification task with targets $t \in \{0, 1\}$ and $p(t = 1 \mid \mathbf{x}_\star) = 0.6$.

   (a) **[6 points]** What is the asymptotic error rate at $\mathbf{x}_\star$ for a 1-nearest-neighbor classifier? (By asymptotic, I mean as the number of training examples $N \to \infty$.) Justify your answer.

   Let $t_\star$ denote the true target and t_N denote the target at the nearest neighbor. These are independent Bernoulli random variables with parameter 0.6. The classifier makes a mistake if $t_\star = 0$ and $t_N = 1$ or if $t_\star = 1$ and $t_N = 0$. Hence, the probability of a mistake, i.e. the error rate, is $0.4 \cdot 0.6 + 0.6 \cdot 0.4 = 0.48$.

   Marking: Full points for a correct answer with justification. If it's correct apart from a calculation error, it gets 5 points. Applying the Union Bound, but not calculating the actual asymptotic error, gets 3 points. Answers that show some understanding but which isn't very related to the correct solution get 1 point.

   (b) **[6 points]** Approximately what is the asymptotic (as $N \to \infty$) error rate at $\mathbf{x}_\star$ for a K-nearest-neighbors classifier when K is very large? Justify your answer.

   For large K, the asymptotic KNN error rate is approximately the Bayes error rate. In this example, the Bayes classifier will predict $y = 1$. Hence, the error rate is 0.4.

   Marking: Full points for a correct answer with justification. Saying it approaches the Bayes error but not calculating the Bayes error gets 3 points. Answers that show some understanding which isn't very related to the correct solution get 1 point. Note: answers which use the squared error rather than classification version of Bayes error lose 4 points, but this is only deducted once if it happens in both parts.

   Mean: 5.4/12

8. **[10 points]** Consider a pair of random variables X and Y whose joint distribution is as follows:

	$Y = 0$	$Y = 1$
$X = 0$	0	0.5
$X = 1$	0.25	0.25

   (a) **[4 points]** Compute the joint entropy $\mathcal{H}(X, Y)$.

   The joint entropy of multiple categorical random variables is the same as the entropy of a single categorical random variable with the same set of probabilities. So in this case, the entropy is:

$$-0.5 \log 0.5 - 0.25 \log 0.25 - 0.25 \log 0.25 = 1.5.$$

   Marking: 3 marks for the definition, 1 mark for the correct (simplified) answer.

(b) **[6 points]** Compute the conditional entropy $\mathcal{H}(Y \mid X)$.

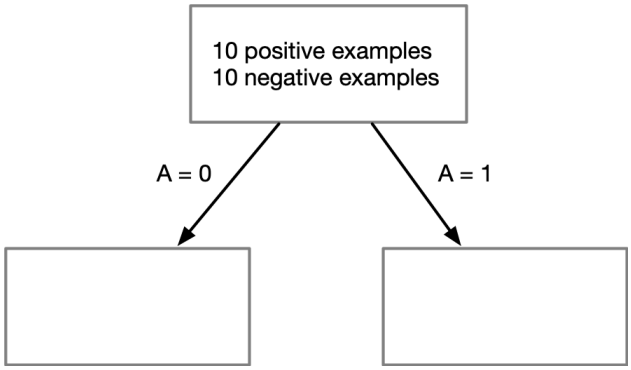
$$\begin{aligned}\mathcal{H}(Y \mid X) &= \sum_x p(x)\, \mathcal{H}(Y \mid X = x) \\ &= 0.5\, \mathcal{H}(Y \mid X = 0) + 0.5\, \mathcal{H}(Y \mid X = 1) \\ &= 0.5 \cdot 0 + 0.5 \cdot 1 \\ &= 0.5\end{aligned}$$

Marking: 5 points for the definitions, 1 point for the correct (simplified) answer. Partially correct definitions get partial marks.

Mean: 8.7/10

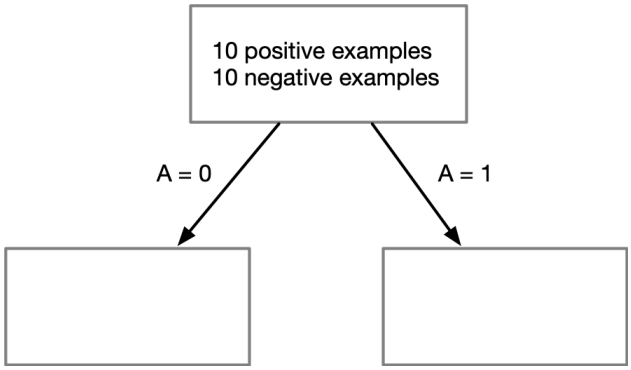
9. **[12 points]** Suppose we are deciding how to split a particular node in a decision tree. Currently 20 training examples (10 positive and 10 negative) belong to this node. If we split on a certain binary-valued attribute A, then some examples will go down the left branch and others will go down the right branch.

   (a) **[6 points]** What is the *minimum* possible value of the information gain for a split? Give an example of a split that achieves this value by indicating the numbers of training examples that go down each branch. Justify your answer.



The minimum possible information gain is 0. Since information gain is nonnegative, it suffices to show one example that achieves 0. Multiple answers are possible; for instance, 5pos/5neg on the left branch and 5pos/5neg on the right branch. To determine the information gain, let X and Y denote the split and the class, respectively. Then $\mathcal{H}(Y) = 1$ because it's a fair coin flip. Similarly, $\mathcal{H}(Y|X) = \sum_x p(x)\mathcal{H}(Y|X = x) = \sum_x p(x) = 1$. The conditional entropies are both 1 because they're also fair coin flips. Hence, $\mathcal{IG} = \mathcal{H}(Y) - \mathcal{H}(Y|X) = 0$.

   (b) **[6 points]** What is the *maximum* possible value of the information gain for a split? Give an example of a split that achieves this value by indicating the numbers of training examples that go down each branch. Justify your answer.



The maximum possible information gain is 1, which can be achieved if all the positive examples go down the left branch and all the negative examples go down the right branch (or vice versa). We already checked that $\mathcal{H}(Y) = 1$ above. The conditional entropies $\mathcal{H}(Y|X = x)$ are all 0 because the conditional distributions are deterministic. Hence, the information gain is 1.

Marking: For each part, it's 2 points for allocating the examples correctly between the bins, 2 points for the value of the information gain, and 2 points for the justification.

Mean: 10.6/12

10. **[8 points]** Consider the following convolution layer in a conv net. The input has a spatial dimension of 12×12, and has 10 channels. The convolution kernels are 3×3, the stride is 2, and we use "valid" convolution, which means that each output neuron only looks at image regions that lie entirely within the spatial bounds of the input. The output dimension is 5×5, with 20 channels.

    For this question, you don't need to show your work or justify your answer, but doing so may help you get partial credit.

    (a) **[4 points]** How many weights are required for this convolution layer?

    A convolutional layer with N input channels, M output channels, and $K \times K$ spatial extent requires MNK^2 weights. Hence, we need

    $$10 \cdot 20 \cdot 3 \cdot 3 = 1800 \text{ weights.}$$

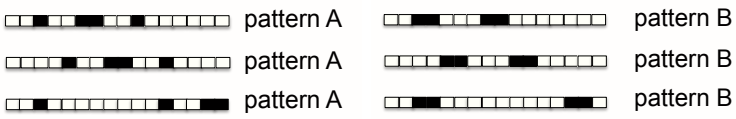
    (b) **[4 points]** Suppose we instead make this a locally connected layer, i.e. don't use weight sharing. How many weights are required?

    The locally connected layer has the same pattern of connections as the convolution layer, but each of the $5 \times 5 = 25$ output locations will have its own separate set of weights. Hence, the total number of weights is $25 \times 1800 = 45000$.

    Marking: Each mistake loses 2 points. An answer that shows some insight gets 1 point. For the second part, calculating it correctly as a fully connected layer gets 2 points.

    Mean: 5.9/8

11. **[10 points]** Recall the example from lecture that we showed linear classifiers were unable to classify. The task is to distinguish the following two patterns in all 16 possible translations with wrap-around. The inputs are 16-dimensional binary vectors, where black indicates 1 and white indicates 0.



Now let's classify these patterns using a conv net.

- First we have a convolution layer with a single convolution kernel of size 3 with weights $\mathbf{w} = (w_1 \; w_2 \; w_3)^\top$ and bias b. Unlike in ordinary convolution layers, this one will use wrap-around. The output of this layer has 16 units.
- We apply the ReLU activation function to this layer.
- We pool together the activations by taking the sum. Call this value z.
- We threshold the result at a value r. If $z \geq r$, it is classified as B, otherwise it's classified as A.

Your task is to choose the weights $\mathbf{w}$, bias b, and threshold r to correctly separate all instances of the patterns. You are not required to show your work, but explaining your reasoning may help you get partial credit.

There are lots of possible solutions. One is:

$$\mathbf{w} = (1 \; 1 \; 0)^\top$$
$$b = -1$$
$$r = 1.5$$

The convolution layer will have an output of 1 exactly in those spatial locations with 2 consecutive inputs being on, and an output of 0 elsewhere. Hence, it will activate in 2 locations for pattern B and 1 location for pattern A. If we threshold it at 1.5, it will separate instances of A and B.

Marking: Full marks for any solution which correctly separates A and B. Full marks if it's correct apart from A and B being reversed. 8 points if it's correct apart from using the wrong threshold. 5 points if there are more mistakes, but the explanation recognizes the main difference between the patterns. 2 points if the answer is totally wrong, but the explanation shows some insight about how to attack the problem.

Mean: 5.8/10