PROBLEM 2

(5.1) Consider the following MDP. It has states \{0,1,2,3,4\} with 4 as the starting state. In every state, you can take one of two possible actions: walk (W) or jump (J). The Walk action decreases the state by one. The Jump action has probability 0.5 of decreasing the state by two, and probability 0.5 of leaving the state unchanged. Actions will not decrease the state below zero: you will remain in state 0 no matter which action you will take (i.e., state 0 is a terminal state). Jumping in state 1 leads to state 0 with probability 0.5 and state 1 with probability 0.5. This definition leads to the following transition functions:

- For states \(k \geq 1\), \(T(k, W, k - 1) = 1\)
- For states \(k \geq 2\), \(T(k, J, k - 2) = T(k, J, k) = 0.5\)
- For state \(k = 1\), \(T(k, J, k - 1) = T(k, J, k) = 0.5\)

The reward gained when taking an action is the distance travelled squared: \(R(s, a, s') = (s - s')^2\). The discount factor is \(\gamma = 0.5\).

(a) (4 points) Suppose we initialize \(Q_0(s, a) = 0\) for all \(s \in \{0,1,2,3,4\}\) and \(a \in \{J,W\}\). Evaluate the Q-values \(Q_1(s, a)\) after exactly one Q-value iteration.

<table>
<thead>
<tr>
<th></th>
<th>(s_0)</th>
<th>(s_1)</th>
<th>(s_2)</th>
<th>(s_3)</th>
<th>(s_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) (4 points) What is the policy that we would derive from \(Q_1(s, a)\)?

<table>
<thead>
<tr>
<th>(s_1)</th>
<th>(s_2)</th>
<th>(s_3)</th>
<th>(s_4)</th>
</tr>
</thead>
</table>

(c) (2 points) What are the values \(V_1(s)\) corresponding to \(Q_1(s, a)\)?

<table>
<thead>
<tr>
<th>(s_0)</th>
<th>(s_1)</th>
<th>(s_2)</th>
<th>(s_3)</th>
<th>(s_4)</th>
</tr>
</thead>
</table>

(d) (4 points) Will the policy change after the second iteration? If your answer is “yes”, briefly describe how
**PROBLEM 4**

**Problem 5** Consider a reinforcement learning problem specified by the following Markov Decision Process (MDP).

![Diagram of MDP](image)

We have five states representing steps along one direction. Call these states $S_1$, $S_2$, $S_3$, $S_4$, and $S_5$. From each state, except the end states, we can move either left or right. The available actions in state $S_1$ is just to move right while the action available in $S_5$ is to move left. We can move left or right in each intermediate state. The reward for taking any action is 1 except when moving right from $S_4$ or left from $S_5$ which provide reward 10. Assume a discount factor $\gamma = 0.5$. Note that $\sum_{i=1}^{\infty} 0.5^i = 1$.

(a) **(5 points)** What is the optimal policy for this MDP? Specify action (L/R) to take in each state.

\[
S_1 : (   ) \quad S_2 : (   ) \quad S_3 : (   ) \quad S_4 : (   ) \quad S_5 : (   )
\]  \hspace{1cm} (12)

(b) Suppose we apply value iteration on this MDP. What is the value of state $S_3$ after

(2 points) one value iteration

(2 points) two value iterations

(3 points) $\infty$ number of value iterations
Problem 2 Suppose we have a recommender problem with \( n \) users \( a \in \{1, \ldots, n\} \) and \( m \) items \( i \in \{1, \ldots, m\} \). For simplicity, we will treat the target rating values as class labels, i.e., using \( \{-1, 1\} \) ratings (dislikes, likes). Each user is likely to provide feedback for only a small subset of possible items and thus we must constrain the models so as not to overfit. Our goal here is to understand how a simple neural network model applies to this problem, and what its constraints are. To this end, we introduce an input unit corresponding to each user and each item. In other words, there are \( n + m \) input units. When querying about a selected entry, \((a, i)\), only the \( a^{th} \) user input unit and \( i^{th} \) item input unit are active (set to 1), the rest are equal to zero and will not affect the predictions. Put another way, only the outgoing weights from these two units matter for predicting the value (class label) for entry \((a, i)\). Figure below provides a schematic representation of the model.

User \( a \) has two outgoing weights, \( U_{a1} \) and \( U_{a2} \), and item \( i \) has two outgoing weights, \( V_{i1} \) and \( V_{i2} \). These weights are fed as inputs to the two hidden units in the model. The hidden units evaluate

\[
\begin{align*}
z_1 &= U_{a1} + V_{i1}, & f(z_1) &= \max\{0, z_1\} \\
z_2 &= U_{a2} + V_{i2}, & f(z_2) &= \max\{0, z_2\}
\end{align*}
\]

Thus, for \((a, i)\) entry, our network outputs

\[
F(a, i; \theta) = W_1 f(z_1) + W_2 f(z_2) + W_0
\]

where \( \theta \) denotes all the weights \( U, V, \) and \( W \). In a vector form, each user \( a \) has a two-dimensional vector of outgoing weights \( \vec{u}_a = [U_{a1}, U_{a2}]^T \) and each item \( i \) has \( \vec{v}_i = [V_{i1}, V_{i2}]^T \). The input received by the hidden units, if represented as a vector, is then \( \vec{z} = [z_1, z_2]^T = \vec{u}_a + \vec{v}_i \).
Consider a simple version of the problem where we have only two users, \{a, b\}, and two items \{1, 2\}. So the recommendation problem can be represented as a 2x2 matrix. We will initialize the first layer weights as shown in Figure (NN) below.

(2.1) **(4 points)** Using the initial input-to-hidden layer weights, each of the four user-item pairs in the 2x2 matrix are mapped to a corresponding feature representation \([f(z_1), f(z_2)]^T\) (hidden unit activations). Please mark the points on the right with the correct pair, e.g., (a,1), that it corresponds to.

![Figure (NN): Outgoing weight vectors from user/item input units (left); hidden layer activations (right)](image-url)
(2.2) **(4 points)** Suppose we keep the input to the hidden layer weights \((U’s\ and\ V’s)\) at their initial values shown in Figure (NN), and only estimate the weights \(W\) corresponding to the output layer. Different choices of the output layer weights will result in different predicted 2x2 matrices of \([-1, 1]\) labels. Which (if any) of the following matrices the neural network cannot reproduce with any choice of the output layer weights \(W_1, W_2,\) and \(W_0\)?

\[
\begin{array}{cc}
1 & 2 \\
\hline
a & +1 & +1 \\
b & +1 & -1 \\
\end{array}
\quad\quad
\begin{array}{cc}
1 & 2 \\
\hline
a & -1 & +1 \\
b & +1 & -1 \\
\end{array}
\]

Learning a new representation for examples (hidden layer activations) is always harder than learning the linear classifier operating on that representation. In neural networks, the representation is learned together with the end classifier using stochastic gradient descent. We initialize the output layer weights as \(W_1 = W_2 = 1\) and \(W_0 = -1\).

(2.3) **(2 points)** Assume that all the weights are initialized as provided above. What is the class label \((+1/-1)\) that the network would predict in response to \((b, 2)\) (user \(b,\) item 2)?

(2.4) **(6 points)** Assume that we observe the opposite label from your answer to the previous question. In other words, there is a training signal at the network output. All the weights are initialized as before. Please mark (check the boxes) of all the weights in Figure (SGD) that would change (have non-zero update) based on a single stochastic gradient descent step in response to \((b, 2)\) with our specific weight initialization and the target label. Note that the input units \(a, b\) and 1, 2 are activated with 0’s and 1’s as shown inside the circles. We are not asking about whether \(W_0\) would change.

![Figure (SGD): Neural network for stochastic gradient descent.](image)

\[
F(b, 2; \theta) = W_1 f(z_1) + W_2 f(z_2) + W_0
\]
**PROBLEM 6**

**Problem 5** Consider a robot that can either stand still and CHARGE (using its solar panels) or it can scurry around and EXPLORE. The robot’s state (as we measure it) represents only how charged its battery is and can be EMPTY, LOW or HIGH. The robot is very eager to explore and this is how the rewards are set. The MDP transition probabilities and rewards are specified as shown below.

<table>
<thead>
<tr>
<th>s</th>
<th>a</th>
<th>s’</th>
<th>T(s,a,s’)</th>
<th>R(s,a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>EXPLORE</td>
<td>LOW</td>
<td>1.0</td>
<td>+2</td>
</tr>
<tr>
<td>LOW</td>
<td>EXPLORE</td>
<td>EMPTY</td>
<td>1.0</td>
<td>+2</td>
</tr>
<tr>
<td>EMPTY</td>
<td>EXPLORE</td>
<td>EMPTY</td>
<td>1.0</td>
<td>-10</td>
</tr>
<tr>
<td>HIGH</td>
<td>CHARGE</td>
<td>HIGH</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>LOW</td>
<td>CHARGE</td>
<td>HIGH</td>
<td>1.0</td>
<td>-1</td>
</tr>
<tr>
<td>EMPTY</td>
<td>CHARGE</td>
<td>HIGH</td>
<td>1.0</td>
<td>-10</td>
</tr>
</tbody>
</table>

Note that the reward only depends on the robot’s current state and action, not the state that it transitions to.

(5.1) **(3 points)** Based on the transitions and rewards (without further calculation), what is the optimal policy for this robot if we set the discount factor \( \gamma = 0 \)?

\[
\begin{align*}
\pi^*_0(\text{HIGH}) &= \\
\pi^*_0(\text{LOW}) &= \\
\pi^*_0(\text{EMPTY}) &= 
\end{align*}
\]

(5.2) **(4 points)** Could changing the discount factor \( \gamma \) change the optimal action to take in any state? (check all that apply)

( ) when \( s = \text{HIGH} \)?
( ) when \( s = \text{LOW} \)?
( ) when \( s = \text{EMPTY} \)?

(5.3) **(4 points)** Let’s see how the robot values its states, and recovers those values through value iteration, when the discount factor is set to \( \gamma = 0.5 \). We start with all zero values as shown in the first value column. Please fill out the table

<table>
<thead>
<tr>
<th>s</th>
<th>( V_0(s) )</th>
<th>( V_1(s) )</th>
<th>( V_2(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPTY</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOW</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGH</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5.4) **(4 points)** If the robot uses values \( V_2(s) \) as the true (converged) values, which action would it take in state \( s = \text{LOW} \)? (Show your calculation)
Problem 2 Consider a simple two layer neural network for classifying points on the plane. Our network has additional constraints beyond the two-layer architecture. The main constraint is that all the incoming weights to the output layer, $w_j$, $j = 1, \ldots, m$, are set equal to one, save for the offset parameter $w_0$ which remains adjustable. The hidden layer units can be chosen arbitrarily, including their number $m$.

\[
F = \sum_{j=1}^{m} f_j w_j + w_0
\]

The weights $w_{ij}$, $i = 0, 1, 2$, $j = 1, \ldots, m$, can be chosen as needed where $w_{0j}$ is the offset parameter for the $j^{th}$ hidden unit.

(2.1) (2 points) If the activation function is $\text{sign}(\cdot)$, hidden units act as linear classifiers and can be drawn graphically as such. Write down the normal vector to the decision boundary of the linear classifier corresponding to the $i^{th}$ hidden unit?

Figure NN1: training set of points

(2.2) (6 points) Figure NN1 shows the points we wish to classify correctly. Please draw graphically (as linear classifiers, including orientation) the smallest number of hidden units that enable our constrained output layer to classify the points correctly. For this part you must assume that hidden units are ReLU units.
(2.3) **(4 points)** Neural networks are powerful but can be challenging to train. Consider a simple deep architecture shown below where there is a single unit in each layer.

![Diagram of neural network](image)

Each unit uses ReLU activation such that \( f_i = \text{ReLU}(w_i f_{i-1} + w_0) \), \( i = 1, \ldots, m \), where \( f_0 = x \). The output unit is linear \( F = w_f m + w_0 \). For a given input \( x \), we observe target output \( y \), and measure loss \( \text{Loss}(F, y) \), where \( F \) is the activation of the final linear unit in response to \( x \). In order to train these models with gradient descent, we must be able to calculate gradients. Let

\[
d_i = \frac{\partial}{\partial f_i} \text{Loss}(F, y), \quad i = 1, \ldots, m
\]

(1)

Write down a gradient descent update rule for parameter \( w_1 \) using \( d_i, i = 1, \ldots, m \).

---

(2.4) **(4 points)** Suppose \( \frac{\partial}{\partial F} \text{Loss}(F, y) = 1 \), i.e., we didn’t quite predict the response correctly. In this case, which of the following statements are necessarily true in our deep architecture? Check all that apply.

\[
\begin{align*}
\text{() } & d_{i-1} = w_i f_i d_i, \quad i = 2, \ldots, m \\
\text{() } & d_{i-1} = w_i [f_i \geq 0] d_i, \quad i = 2, \ldots, m \\
\text{() } & d_m = w \\
\text{() } & d_1 \rightarrow 0 \text{ as } m \text{ increases (vanishing gradient)}
\end{align*}
\]

(2)
Problem 8

Problem 6 You are running a 3 mile race. Every 10 minutes you must decide whether to walk or run for the next 10 minutes based on your current distance from the start (represented as states 0, 1, and 2 but no actions will be taken from state 3 because you will have already finished). If you walk, you will advance 1 mile over the next 10 minutes. If you run, you have a 50% chance to advance 1 mile and a 50% chance to advance 2 miles over the next 10 minutes. You want to finish the race, but running is tiring and takes effort. You will receive a reward of 10 for finishing the race (ending up in state 3). However, every time you run, you get an additional “reward” -1. You decide to use a Markov Decision Process with $\gamma=0.5$ to determine what action you should take from each state. The full table of transition probabilities and rewards is shown below.

<table>
<thead>
<tr>
<th>s</th>
<th>a</th>
<th>s’</th>
<th>$T(s,a,s’)$</th>
<th>$R(s,a,s’)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>WALK</td>
<td>1</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>WALK</td>
<td>2</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>WALK</td>
<td>3</td>
<td>1.0</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>RUN</td>
<td>1</td>
<td>0.5</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>RUN</td>
<td>2</td>
<td>0.5</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>RUN</td>
<td>2</td>
<td>0.5</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>RUN</td>
<td>3</td>
<td>0.5</td>
<td>+9</td>
</tr>
<tr>
<td>2</td>
<td>RUN</td>
<td>3</td>
<td>1.0</td>
<td>+9</td>
</tr>
</tbody>
</table>

(6.1) **(3 points)** Suppose we initialize $Q_0(s,a) = 0$ for all $s \in \{0, 1, 2\}$ and all $a \in \{WALK, RUN\}$. We assume that the values in state $s = 3$ are always zero for any action. Evaluate the Q-values $Q_1(s,a)$ after exactly one Q-value iteration.

<table>
<thead>
<tr>
<th>a</th>
<th>s=0</th>
<th>s=1</th>
<th>s=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WALK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(6.2) **(3 points)** What is the ideal policy derived from $Q_1(s,a)$?

$$
\pi_1^*(s = 0) = $$
$$
\pi_1^*(s = 1) = $$
$$
\pi_1^*(s = 2) = $$

(9)

(6.3) **(3 points)** What are the values $V_1(s)$ using the values of $Q_1(s,a)$ calculated above?

<table>
<thead>
<tr>
<th>s=0</th>
<th>s=1</th>
<th>s=2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(6.4) (3 points) Consider now iterating Q-values one more time to obtain $Q_2(s, a)$. We are only interested in here what happens at $s = 0$. For what range of values of the discount factor $0 \leq \gamma \leq 1$ would the action derived from $Q_2(0, a)$ suggest that we RUN?
PROBLEM 9

(1.1) **(3 points)** If we solve for the maximum margin separator in the separable case, will the margin increase, decrease, or stay the same if we remove one point from the training set and resolve? Please check all that are possible.

( ) increase, ( ) decrease, ( ) stays the same

(1.2) **(2 points)** If a dataset is not linearly separable, can we make it so by omitting some of the coordinates? (Y/N) ( )

(1.3) **(4 points)** Suppose we have only three classifiers in our set $\mathcal{H} = \{h_1, h_2, h_3\}$ with corresponding test errors $\mathcal{E}(h_1) < \mathcal{E}(h_2) < \mathcal{E}(h_3)$. We sampled five training sets of different sizes. Based on each training set, we selected the classifier $\hat{h} \in \mathcal{H}$ with the lowest training error, and recorded the corresponding test error $\mathcal{E}(\hat{h})$. These test errors are shown as points below. Rank the plots (1,2,3) according to how likely they are to happen as a function of the training set size, i.e., as we increase $n$. Rank 1 means the most likely to occur.

![Test Error Plots](image.png)
Problem 2  We would like to build a neural network model that can detect whether two inputs are the same or not. To this end, we assume two input coordinates $x_1$ and $x_2$, two hidden units $f_1$ and $f_2$, and a single output neurons $y$. All units except input units involve ReLU non-linearity.

![Diagram of neural network](image)

(2.1) **(6 points)** Specify the parameter values in the network such that output $y = 1$ if $x_1 = x_2$ and $y = 0$ if $|x_1 - x_2| \geq \epsilon$. Two values of the parameter matrix are pre-specified.

\[
\begin{bmatrix}
v_0 \\
v_1 \\
v_2 \\0
\end{bmatrix} = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix},
\begin{bmatrix}
w_{01} & w_{02} \\
w_{11} & w_{12} \\
w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} \epsilon \\ \epsilon \\ \epsilon \end{bmatrix},
\begin{bmatrix}
0 \\
1 \\0
\end{bmatrix}
\]

(2.2) **(3 points)** Suppose now for simplicity that $x_t$ is binary 0/1 and that we receive $x_1, x_2, \ldots$, in a sequence. Our goal is to detect whether successive pairs of inputs are identical using a recurrent neural network. We do so by feeding the RNN state as an input to the above feed-forward detector. Which of the following state update equations would be suitable for this purpose? You can assume that $s^0 = [0, 0]^T$.

\[
\begin{align*}
(1) & \quad s^t = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} s^{t-1} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_t \\
(2) & \quad s^t = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} s^{t-1} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_t \\
(3) & \quad s^t = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} s^{t-1} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} x_t
\end{align*}
\]
### Problem 7
The following graph specifies the states and transition probabilities for a Markov Decision Process (MDP). There are only two actions in this MDP: \(a = S\) (stay) or \(a = M\) (move). If you elect to stay, you remain in the same state with probability one. If you move, you change states according to the probabilities specified below. \(F\) is the only terminal state where you don’t move even if you select \(a = M\).

![Graph diagram]

The rewards in this MDP are associated with state transitions such that

\[
R(B \rightarrow D) = -1, \quad R(D \rightarrow F) = +10, \quad R(A \rightarrow C) = -2,
\]

and all the remaining rewards are zero. The discount factor is \(\gamma = 0.5\).

1. **(4 points)** Suppose we initialize the values as
   
   \[
   V_0(A) = -1, \quad V_0(B) = 2, \quad V_0(C) = 1, \quad V_0(D) = 1, \quad V_0(E) = 0, \quad V_0(F) = 3
   \]

   What would be the resulting action to take in states \(A\) and \(C\)?

2. **(3 points)** Calculate \(V_1(C)\) after one value iteration.

3. **(2 points)** Suppose we perform value iteration until convergence obtaining \(V^*(s)\), \(s = A, B, C, D, E, F\). What is the resulting \(V^*(F)\)? ( )

4. **(3 points)** Are we guaranteed to get the cumulative discounted reward equal to \(V^*(D)\) if we begin in state \(D\) and act optimally according to the converged values? (Y/N) ( )
Problem 5  Consider a simple feed-forward neural network model that takes \( x \in \mathbb{R} \) as an input and has two ReLU hidden units.

\[
\begin{align*}
    \begin{bmatrix}
    z_1 \\
    z_2 \\
    \end{bmatrix}
    &= \begin{bmatrix}
    x \\
    \end{bmatrix} \begin{bmatrix}
    w_{11} & w_{01} \\
    w_{12} & w_{02} \\
    \end{bmatrix}
    = \begin{bmatrix}
    1 & 1 \\
    2 & -2 \\
    \end{bmatrix}
\end{align*}
\]

(14)

(5.1) (2 points) When is the output of the first hidden unit, i.e., \( f_1 \), exactly zero as a function of \( x \in \mathbb{R} \)?

(5.2) (6 points) Given the parameters in Eq(14), sketch in the figure below how examples \( x \in [-2, 2] \) map to the 2-dimensional feature coordinates \((f_1, f_2)\). In other words, map the interval \([-2, 2]\) in the \(x\)-space to the 2-dimensional space of hidden unit activations.

(5.3) (2 points) Are the training examples \((x = -1, y = -1), (x = 1, y = 1), \) and \((x = 2, y = -1)\) linearly separable in the \((f_1, f_2)\) coordinates? (Y/N) ( )
(5.4) (4 points) Suppose \( w_1 = 1, w_2 = 1, \) and \( w_0 = 0. \) We use \( \text{Loss}_h(yf) = \max\{0, 1 - yf\} \) to measure the loss given the network output \( f \) and true label \( y. \) The parameters will potentially change if we perform a gradient descent step in response to \((x, y)\) where \( y = -1. \) Provide the range of values of \( x \) such that \( w_{q2} \) decreases after the update: \( x \in (\quad )\)

(5.5) (3 points) The architecture and training details are sometimes quite relevant to how well a particular neural network model works. Suppose we use ReLU activation functions for all the hidden units (one layer). Briefly explain what will happen if we initialized all the weights to zero, and all the offset parameters to -1, and ran the stochastic gradient descent method to learn the parameters

(5.6) (6 points) We experimented with three different architecture/training combinations:

( ) \( m \) hidden units, no regularization
( ) \( 2m \) hidden units, no regularization
( ) \( 2m \) hidden units, dropout regularization

Student who carried out the experiments observed that

Model A: training error 0.2, validation error 0.35
Model B: training error 0.25, validation error 0.3
Model C: training error 0.1, validation error 0.4

Please assign the models A, B, and C to their best fitting setups above.
Problem 4  Consider a 2-layer feed-forward neural network that takes in \( x \in \mathbb{R}^2 \) and has two ReLU hidden units. Note that the hidden units have no offset parameters.

\[
\begin{align*}
    z &= f(z_1)v_1 + f(z_2)v_2 + v_0 \\
    z_1 &= x_1w_{11} + x_2w_{12} \\
    z_2 &= x_1w_{12} + x_2w_{22} \\
    f(z_j) &= \max\{0, z_j\}
\end{align*}
\]

(4.1) **6 points** The values of the weights in the hidden layer are set such that they result in the \( z_1 \) and \( z_2 \) “classifiers” as shown in the figure by the decision boundaries and the corresponding normal vectors marked as (1) and (2). Approximately sketch on the right how the input data is mapped to the 2-dimensional space of hidden unit activations \( f(z_1) \) and \( f(z_2) \). Only map points marked ‘a’ through ‘f’. Keep the letter indicators.

(4.2) **2 points** If we keep these hidden layer parameters fixed but add and train additional hidden layers (applied after this layer) to further transform the data, could the resulting neural network solve this classification problem? (Y/N) ( )
(4.3) **(3 points)** Suppose we stick to the 2-layer architecture but add lots more ReLU hidden units, all of them without offset parameters. Would it be possible to train such a model to perfectly separate these points? (Y/N) ( )

(4.4) **(3 points)** Initialization of the parameters is often important when training large feed-forward neural networks. Which of the following statements is correct? Check T or F for each statement.

( ) If we use tanh or linear units and initialize all the weights to very small values, then the network initially behaves as if it were just a linear classifier

( ) If we randomly set all the weights to very large values, or don’t scale them properly with the number of units in the layer below, then the tanh units would behave like sign units.

( ) A network with sign units cannot be effectively trained with stochastic gradient descent

(4.5) **(3 points)** There are many good reasons to use convolutional layers in CNNs as opposed to replacing them with fully connected layers. Please check T or F for each statement.

( ) Since we apply the same convolutional filter throughout the image, we can learn to recognize the same feature wherever it appears.

( ) A fully connected layer for a reasonably sized image would simply have too many parameters

( ) A fully connected layer can learn to recognize features anywhere in the image even if the features appeared preferentially in one location during training
PROBLEM 14

Shallow neural network

1. (8 points)

We consider classifying points on the plane using a neural net with one hidden layer as shown above. All hidden units are ReLUs.

(a) Suppose that the $y$ values in the training data are elements of $\{-1, 1\}$ and we wish to make predictions that minimize the hinge-loss. Mark all activation functions that are appropriate for the output layer.

[ ] sigmoid [ ] tanh [ ] ReLU [ ] linear

(b) Consider the positively and negatively labeled points shown below: Suppose we use our neural net specified above to classify these points (that are labeled $\{-1, 1\}$). What is the smallest number of hidden ReLUs that suffice to correctly separate the positive from the negative points?

(c) For each of the $m$ (where $m$ is your answer to the previous question) hidden ReLUs in a network that correctly classifies the data, draw the boundary at which its output transitions from 0 to positive. (Note that these boundaries are not unique).
3.1 Regular guys

We are interested in regularizing the terms separately in logistic regression.

(a) Consider the data in the figure below where we fit the model

\[ P(y = 1 \mid x, w) = \text{Sigmoid}(w_0 + w_1x_1 + w_2x_2) \]

Suppose we fit the model by maximum likelihood, that is, we minimize

\[ J(w) = -\log \Pr(D_{\text{train}}; w) \]

Sketch a possible decision boundary corresponding to \( w^* \).

(b) Is your decision boundary unique?

(c) How many classification errors does it make on the training set?

(d) Now suppose we regularize only the \( w_0 \) parameter; that is, we minimize

\[ J(w) = -\log \Pr(D_{\text{train}}; w) + \lambda w_0^2 \]

with \( \lambda \) approaching \( \infty \). Sketch a possible decision boundary corresponding to \( w^* \).

(e) How many classification errors does it make on the training set?

(f) Now suppose we regularize only the \( w_1 \) parameter; that is, we minimize

\[ J(w) = -\log \Pr(D_{\text{train}}; w) + \lambda w_1^2 \]

with \( \lambda \) approaching \( \infty \). Sketch a possible decision boundary corresponding to \( w^* \).

(g) How many classification errors does it make on the training set?

(h) Now suppose we regularize only the \( w_2 \) parameter; that is, we minimize

\[ J(w) = -\log \Pr(D_{\text{train}}; w) + \lambda w_2^2 \]

with \( \lambda \) approaching \( \infty \).

(i) How many classification errors does it make on the training set?
PROBLEM 17

11.1 A bumpy hill

This is figure 8.8 from Murphy’s textbook. It shows the progress of a gradient descent algorithm, minimizing squared error in a regression problem. Rather than doing the “batch” gradient descent method, where each step goes down the gradient of the squared error, this algorithm randomly selects a training example and takes a step down the gradient of squared error for that example only.

Which of the following are true? Why?

- Gradient descent steps always decrease the objective.
- Gradient descent steps with appropriately chosen step size always decrease the objective.
- Stochastic gradient descent steps always decrease the objective.
- Stochastic gradient descent steps with appropriately chosen step size always decrease the objective.

- There is a training method that at each step, randomly choose a element from the training set and take a gradient step based only on the error for that example. This method is an instance of stochastic gradient descent (using empirical error as metric).
- There are circumstances in which stochastic gradient descent is to be preferred to exact gradient descent.
- Training by randomly choosing an element of your training set, randomly perturbing the x values, and taking a gradient step based only on the error for that example is an instance of stochastic gradient descent with the same metric as exact gradient descent on the empirical error.
- Training by randomly choosing an element of your training set, randomly perturbing the x values, and taking a gradient step based only on the error for that example is an instance of stochastic gradient descent with the same metric as exact gradient descent on the ridge regression error.
PROBLEM 22

Grady Ent decides to train a single sigmoid unit using the following error function:

$$E(w) = 1/2 \sum_i (y(x^i, w) - y^{i*})^2 + 1/2\beta \sum_j w_j^2$$

where $y(x^i, w) = s(x^i \cdot w)$ with $s(z) = 1/(1 + e^{-z})$ being our usual sigmoid function.

1. Write an expression for $\partial E/\partial w_j$. Your answer should be in terms of the training data.
   
   We’re using $y$ in the last line here as shorthand for $y(x^i, w)$, that is, the output of the network on input $x^i$.

2. What update should be made to weight $w_j$ given a single training example $x, y^{*}$. Your answer should be in terms of the training data.

3. Here are two graphs of the output of the sigmoid unit as a function of a single feature $x$. The unit has a weight for $x$ and an offset. The two graphs are made using different values of the magnitude of the weight vector ($\|w\|^2 = \sum_j w_j^2$).

![Graph A](image1)

![Graph B](image2)

Which of the graphs is produced by the larger $\|w\|^2$? Explain.

4. Why might penalizing large $\|w\|^2$, as we could do above by choosing a positive $\beta$, be desirable?
PROBLEM 23

14 Backpropagation

Here you see a very small neural network: it has one input unit, one hidden unit (logistic), and one output unit (linear).

Let's consider one training case. For that training case, the input value is 1 (as shown in the diagram), and the target output value $t = 1$. We're using the following loss function:

$$E = \frac{1}{2} (t - y)^2$$

Please supply numeric answers; the numbers in this question have been constructed in such a way that you don't need a calculator. Show your work in case of mis-calculation in earlier steps.

(a) What is the output of the hidden unit for this input?

(b) What is the output of the output unit for this input?

(c) What is the loss, for this training case?

(d) What is the derivative of the loss with respect to $w_2$, for this training case?

(e) What is the derivative of the loss with respect to $w_1$, for this training case?

(f) With sigmoidal activation, the derivative with respect to $w_1$ and $w_2$ are

$$\frac{\partial E}{\partial w_2} = (t - y)z, \text{ and } \frac{\partial E}{\partial w_1} = (t - y) \cdot w_2 \cdot z \cdot (1 - z) \cdot x.$$

Assume that we now use the rectified linear unit (ReLU) as our activation (or a ramp function). This means that $z = \max(0, w_1 x + w_0)$. What is the derivative of the loss with respect to $w_1$ and $w_2$ at differentiable points with ReLU? Don't use numerical value for this question.
Problem 24

15 Neural Net

Data points are: Negative: (-1, 0) (2, -2) Positive: (1, 0)

Recall that for neural nets with sigmoidal output units, the negative class is represented by a desired output of 0 and the positive class by a desired output of 1. Hint: Some useful values of the sigmoid $s(z)$ are $s(-1) = 0.27$ and $s(1) = 0.73$.

Assume we have a single sigmoid unit:

Assume that the weights are $w_0 = 0, w_1 = 1, w_2 = 1$. What is the computed $y$ value for each of the points on the diagram above?

(a) $x = (-1, 0)$

(b) $x = (2, -2)$

(c) $x = (1, 0)$

(d) What would be the change in $w_2$ as determined by backpropagation using a step size ($\eta$) of 1.0? Assume the squared loss function. Assume that the input is $x = (2, -2)$ and the initial weights are as specified above. Show the formula you are using as well as the numerical result.

1. $\Delta w_2 =$
1 ReLU Backpropagation

1.1 Single output network

The rectified linear unit (ReLU) is a popular activation function for hidden layers. The activation function is a ramp function $f(z) = \max(0, z)$ where $z = wx$. This has the effect of simply thresholding its input at zero. Unlike the sigmoid, it does not saturate near 1 and is also simpler in gradient computations, resulting in faster convergence of SGD. Furthermore, ReLUs can allow networks to find sparse representations, due to their thresholding characteristic, whereas sigmoids will always generate non-zero values. However, ReLUs can have zero gradient when the activation is negative, blocking the backpropagation of gradients.

Here you use a very small neural network: it has one input unit, taking in a value $x$, one hidden unit (ReLU), and one output unit (sigmoid). We include a bias term of -0.2 on the sigmoid unit.

![Diagram of a neural network with one input, two hidden units (ReLU), and one output unit (sigmoid). The diagram shows the flow of data from input to output with weights $w_1$, $w_2$, bias $-0.2$.]

We use the following quantities in this problem:

$$z_1 = w_1 x$$
$$a_1 = \text{ReLU}(z_1)$$
$$z_2 = w_2 a_1 - 0.2$$
$$y = \sigma(z_2)$$

The weights are initially $w_1 = \frac{1}{10}$ and $w_2 = -1$.

Let’s consider one training example. For that training case, the input value is $x = 2$ (as shown in the diagram), and the target output value $t = 1$. We’re using the following loss function:

$$E = \frac{1}{2} (y - t)^2$$

Please supply numeric answers; the numbers in this question have been constructed in such a way that you don’t need a calculator. Show your work in case of mis-calculation in earlier steps.

(a) What is the output of the hidden unit for this input?

(b) What is the output of the output unit for this input?

(c) What is the loss, for this training example?
(d) Write out an abstract symbolic expression for derivative of the loss with respect to $w_1$ as repeated applications of the chain rule. For example, for the derivative of the loss with respect to $w_2$, we would write $\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_2}$.

(e) Write the expression for each partial derivative in the chain rule expansion from the previous part. For example, $\frac{\partial y}{\partial z_2} = y(1 - y)$.

(f) What is the derivative of the loss with respect to $w_1$, for this training example?

(g) What would the update rule for $w_1$ be? With $\eta = $ 

(h) If $\eta$ is large enough, $w_1$ will update from its current value of 0.1 to a negative value. Assume our new value is $w_1 = -0.1$. What will be the output of the output unit for an input of $x = 2$?

(i) What will happen when we try to update the weight, using this new example, for $w_1$ for any value of target? Why?

(j) Is it a bad idea to have a ReLU activation at the output layer?

(k) Consider the following activation function:

$$f(z) = \begin{cases} 
  z & \text{if } z > 0 \\
  \alpha z & \text{if otherwise.}
\end{cases}$$

for some small alpha, e.g. $\alpha = 0.01$, and $z = wx$. Does this address the problem of dying ReLUs?

1.2 Multiple output network

(l) $a_1 = \text{ReLU}(0, w_1 x)$

$$y_1 = \sigma(w_2 a_1)$$
\[ y_2 = \sigma(w_3 a_1) \]

Write out an abstract symbolic expression for the derivative of the loss with respect to \( w_1 \) for the network above with two output units, as repeated applications of the chain rule.

Multi-output (multi-class) networks are used in many settings such as object recognition, where we are trying to classify an image as being one of \( K \) objects. Each of the \( K \) possible objects would correspond to an output unit in the network. For this purpose, the sigmoid activation and squared loss are replaced by softmax activation and cross-entropy loss. This is similar to the multi-class logistic regression we saw in the week 4 exercises.

The softmax is given by:

\[ y_t = \frac{e^{x_t}}{\sum_{j=1}^{K} e^{x_j}}. \]

(m) When \( K > 3 \), why might sigmoid units be a bad idea?

**PROBLEM 26**

**2.1 Weight Decay**

We can add \( L_1 \) and \( L_2 \) penalties to our cost function to regularize our network and prevent overfitting, as we have done with other classification methods. See Neural Networks and Deep Learning by Michael Nielson for derivations and details.

(a) The loss for \( L_2 \) regularized case is:

\[ L(w) = E(w) + \lambda |w|^2 \]

where \( E(w) \) is the error. The update for the \( L_2 \) regularized case is:

\[ w := (1 - \frac{\eta \lambda}{n}) w - \eta \frac{\partial E}{\partial w} \]

with regularization. How does this update affect our weights \( w \) as we increase \( \lambda \)?

**2.2 Dropout**

Dropout is a regularization technique to reduce overfitting to the training data. A randomly selected portion (e.g. 20%) of neurons are ignored during each update cycle of training - these neurons 'drop out' of the network.

During training, without dropout, we compute the activations of a layer \( l \) as \( a_l = f(z_l) \). Assume we are using a dropout probability of 20%, \( a_l = f(z_l)m_l \), where \( m_l \) is a mask of the same length with 20% of the entries set to 0 and the others set to 1.

(a) How might this have a regularization affect?

(b) During the feedforward pass without dropout, the inputs to layer \( l + 1 \) are \( z_{l+1} = W_{l+1} a_l \). With 20% dropout, how show we change this expression?
3 Convolutional Network Architecture**

Consider the following 1D input and two convolutional filters:

\[
\begin{array}{cccc}
1 & 2 & -1 & 1 \\
\end{array}
\begin{array}{ccc}
1 & 0 & -1 \\
-1 & 1 & -1 \\
\end{array}
\]

(a) Convolve the first filter with the input.

(b) Convolve the second filter with the input.

(c) What would the output of the convolutional layer look like with both filters?

(d) In the previous part, the spatial size of the output decreased from that of the original input - i.e. it had fewer neurons. In some settings, we preserve the size by applying zero padding. What would the output of the convolutional layer look like in this case?

(e) Continuing to use zero padding of size 1, we now use a stride length of 2. What is the output of the convolutional layer now? Compare how this relates to the case with no stride.

(f) If our filters were size 5, rather than size 3, with stride 1, how much padding would you apply to maintain output size?

(g) What is a general expression for the output size (number of neurons) in a convolutional layer, in terms of the filter size (F), stride length (S), amount of padding (P), and the input size (W)?

(h) Consider a 10x10 grayscale image input (no RGB channel). If you made a 1-layer fully connected network, with 1 hidden, how many weight parameters would be required in the hidden layer including bias terms (ignore the output layer)?

(i) Consider a 10x10 grayscale image input (no RGB channel). If you made a 2-layer fully connected network, with 10 and 5 hidden units respectively, how many weight parameters would be required in these hidden layers including bias terms (ignore the output layer)?

(j) With the same 10x10 input image, you now use a convolutional layer with \( F = 2 \) (square filters), \( S = 2 \) (in both dimensions), and \( P = 0 \). How is the output volume of the first convolutional layer with 2 filters, including bias terms?

(k) How many parameters does the above convolutional layer have?

(l) One advantage of convolutional networks is that the number of free parameters can be controlled by parameter sharing for a filter across spatial dimensions. Each filter is replicated across the entire visual field (image). That is, if a filter is used at one spatial coordinate \((x_1, y_1)\), the same weights are also used in a different position \((x_2, y_2)\). Adopting the parameter sharing scheme, how many parameters does the convolutional layer now have?
PROBLEM 28

(a) Like Jody last week, Evelyn sees that you are handling a multi-class problem with 4 classes and suggests that you use as a target value \( y^{(1)} \) a vector of two output values, each of which can be 0 or 1, to encode which one of four classes it belongs to.

What is good or bad about Evelyn’s approach?

(b) Seeing you working on an implementation of a neural network with sigmoid units, where the inputs are in the range \([0, 1]\), Jean suggests that you initialize the weights randomly in the range \([0, 100]\).

Is Jean’s idea good?

PROBLEM 29

5 Activation energy

A bunch of AI students walked into a bar...and promptly started debating alternative activation functions for a neural network intended to do binary classification. The input data is in \( \mathbb{R}^2 \); the weight vector is 3-dimensional.

The network in fact has just a single unit with two input dimensions.

For each of the suggested activation functions, say:

- Whether it makes sense to train it with the cross-entropy objective function:

\[
y \log o + (1 - y) \log(1 - o)
\]

where \( y \) is the desired output and \( o \) is the actual (the output of the network).
• Whether it can represent the same class of separators in the two-dimensional input space as
the sigmoidal activation function.

• If it can represent the same class, explain why.

• If it cannot, then draw a separator that can be represented by a sigmoidal activation on a linear
combination of the inputs but not by this activation on a linear combination of the inputs or a separator
that can be represented by this activation function on a linear combination of the inputs, but not by a sigmoidal
activation function on a linear combination of the inputs and explain why.

\[
f(a) = \sin(a)
\]

Sensible to use cross-entropy: ○ Yes ○ No
Same class of separators as sigmoid? ○ No ○ No
If yes, explain why. If no, draw separator as explained above.

---

(b)

\[
f(a) = \begin{cases} 
0 & \text{if } a < 0 \\
1 & \text{if } a > 1 \\
x & \text{otherwise}
\end{cases}
\]

Sensible to use cross-entropy: ○ Yes ○ No
Same class of separators as sigmoid? ○ Yes ○ No
If yes, explain why. If no, draw separator as explained above.
(c) \[ f(α) = e^α \]
Sensible to use cross-entropy:  ○ Yes  ○ No
Same class of separators as sigmoid?  ○ Yes  ○ No
If yes, explain why. If no, draw separator as explained above.

(d) \[ f(α) = α^2 \]
Sensible to use cross-entropy:  ○ Yes  ○ No
Same class of separators as sigmoid?  ○ Yes  ○ No
If yes, explain why. If no, draw separator as explained above.

\[ f(a) = a \]

Sensible to use cross-entropy:  ○ Yes  ○ No

Same class of separators as sigmoid?  ○ Yes  ○ No

If yes, explain why. If no, draw separator as explained above.
6 Neural network short answer

1. If you train a neural-net classifier using stochastic gradient descent, and you use a too small learning rate, what can go wrong?

2. How can you determine that you have used too small a learning rate?

3. If you train a neural-net classifier using stochastic gradient descent, and you use a too large learning rate, what can go wrong?

4. How can you determine that you have used too large a learning rate?

5. Is final error on the training set a good measure for finding out whether the learning rate was set too large?

6. Suppose you train a neural-net classifier using stochastic gradient descent, and you measure training error and validation error after N iterations.
   Which of the two errors is a better estimate for the error on unseen future data and why?

7. In a convolutional layer of a neural network, what gives rise to a smaller output dimensionality: a small stride or a large stride?

8. In a convolutional layer of a neural network, can you use 1x1 convolution to learn a linear classifier on top of the inputs? If yes, what format should the input have?

9. For effective training of a neural network, the network should have at least 4 times as many weights as there are training samples? If yes, why? If no, why not?

10. Assume the following (momentum) weight update rule:

\[
\begin{align*}
    m_0 &= 0 \\
    m_t &= \beta m_{t-1} + (1 - \beta) \frac{\partial E}{\partial w} \\
    w_t &= w_{t-1} + \lambda m_t
\end{align*}
\]  

   (6.1) \quad (6.2) \quad (6.3)

11. For what value of \( \beta \) do you obtain the standard gradient descent rule?
PROBLEM 31

Data points are: Negative: (-1, 0) (2, -2) Positive: (1, 0)

Recall that for neural nets with sigmoidal output units, the negative class is represented by a desired output of 0 and the positive class by a desired output of 1. Hint: Some useful values of the sigmoid \( s(z) \) are \( s(-1) = 0.27 \) and \( s(1) = 0.73 \).

Assume we have a single sigmoid unit:

Assume that the weights are \( w_0 = 0, w_1 = 1, w_2 = 1 \). What is the computed \( y \) value for each of the points on the diagram above?

(a) \( x = (-1, 0) \)

(b) \( x = (2, -2) \)

(c) \( x = (1, 0) \)

(d) What would be the change in \( w_2 \) as determined by backpropagation using a step size (\( \eta \)) of 1.0? Assume the squared loss function. Assume that the input is \( x = (2, -2) \) and the initial weights are as specified above. Show the formula you are using as well as the numerical result.

1. \( \Delta w_2 = \)
1 Decision Trees

Data points are: Negative: (-1, 0) (2, 1) (2, -2) Positive: (0, 0) (1, 0)

Construct a decision tree using the algorithm described in the notes for the data above.

1. Show the tree you constructed in the diagram below. The diagram is more than big enough, leave any parts that you don’t need blank.

2. Draw the decision boundaries on the graph at the top of the page.
3. Explain how you chose the top-level test in the tree. The following table may be useful.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>-(x/y)*lg(x/y)</th>
<th>x</th>
<th>y</th>
<th>-(x/y)*lg(x/y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.50</td>
<td>1</td>
<td>5</td>
<td>0.46</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.53</td>
<td>2</td>
<td>5</td>
<td>0.53</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.39</td>
<td>3</td>
<td>5</td>
<td>0.44</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.50</td>
<td>4</td>
<td>5</td>
<td>0.26</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.31</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. What class does the decision tree predict for the new point: (1, -1.01)
Problem 33

2 Nearest Neighbors

Data points are: Negative: (-1, 0) (2, 1) (2, -2) Positive: (0, 0) (1, 0)

1. Draw the decision boundaries for 1-Nearest Neighbors on the graph above. Your drawing should be accurate enough so that we can tell whether the integer-valued coordinate points in the diagram are on the boundary or, if not, which region they are in.

2. What class does 1-NN predict for the new point: (1, -1.01) Explain why.

3. What class does 3-NN predict for the new point: (1, -1.01) Explain why.
Problem 34

4 Machine Learning — Continuous Features (20 points)

In all the parts of this problem we will be dealing with one-dimensional data, that is, a set of points \((x^i)\) with only one feature (called simply \(x\)). The points are in two classes given by the value of \(y^i\). We will show you the points on the \(x\) axis, labeled by their class values; we also give you a table of values.

4.1 Nearest Neighbors

<table>
<thead>
<tr>
<th>(i)</th>
<th>(x^i)</th>
<th>(y^i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>

1. In the figure below, draw the output of a 1-Nearest-Neighbor classifier over the range indicated in the figure.
2. In the figure below, draw the output of a 5-Nearest-Neighbor classifier over the range indicated in the figure.
4.2 Decision Trees

Answer this problem using the same data as in the Nearest Neighbor problem above.

Which of the following three tests would be chosen as the top node in a decision tree?

\[ x \leq 1.5 \quad x \leq 5 \quad x \leq 10.5 \]

Justify your answer.

You may find this table useful.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>-(x/y)*lg(x/y)</th>
<th>x</th>
<th>y</th>
<th>-(x/y)*lg(x/y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.50</td>
<td>1</td>
<td>8</td>
<td>0.38</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.53</td>
<td>3</td>
<td>8</td>
<td>0.53</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.39</td>
<td>5</td>
<td>8</td>
<td>0.42</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.50</td>
<td>7</td>
<td>8</td>
<td>0.17</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.31</td>
<td>1</td>
<td>9</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0.46</td>
<td>2</td>
<td>9</td>
<td>0.48</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.53</td>
<td>4</td>
<td>9</td>
<td>0.52</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0.44</td>
<td>5</td>
<td>9</td>
<td>0.47</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0.26</td>
<td>7</td>
<td>9</td>
<td>0.28</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>0.43</td>
<td>8</td>
<td>9</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.53</td>
<td>1</td>
<td>10</td>
<td>0.33</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.22</td>
<td>3</td>
<td>10</td>
<td>0.52</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0.40</td>
<td>7</td>
<td>10</td>
<td>0.36</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>0.52</td>
<td>9</td>
<td>10</td>
<td>0.14</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6 Pruning Trees (20 points)

Following are some different strategies for pruning decision trees. We assume that we grow the decision tree until there is one or a small number of elements in each leaf. Then, we prune by deleting individual leaves of the tree until the score of the tree starts to get worse. The question is how to score each possible pruning of the tree.

For each possible definition of the score below, explain whether or not it would be a good idea and give a reason why or why not.

1. The score is the percentage correct of the tree on the training set.

2. The score is the percentage correct of the tree on a separate validation set.

3. The score is the percentage correct of the tree, computed using cross validation.
4. The score is the percentage correct of the tree, computed on the training set, minus a constant $C$ times the number of nodes in the tree.
   
   $C$ is chosen in advance by running this algorithm (grow a large tree then prune in order to maximize percent correct minus $C$ times number of nodes) for many different values of $C$, and choosing the value of $C$ that minimizes training-set error.

5. The score is the percentage correct of the tree, computed on the training set, minus a constant $C$ times the number of nodes in the tree.
   
   $C$ is chosen in advance by running cross-validation trials of this algorithm (grow a large tree then prune in order to maximize percent correct minus $C$ times number of nodes) for many different values of $C$, and choosing the value of $C$ that minimizes cross-validation error.
Problem 36

Problem 4: Learning (25 points)

Part A: (5 Points)

Since the cost of using a nearest neighbor classifier grows with the size of the training set, sometimes one tries to eliminate redundant points from the training set. These are points whose removal does not affect the behavior of the classifier for any possible new point.

1. In the figure below, sketch the decision boundary for a 1-nearest-neighbor rule and circle the redundant points.

![Decision Boundary Diagram]

2. What is the general condition(s) required for a point to be declared redundant for a 1-nearest-neighbor rule? Assume we have only two classes (+, -). Restating the definition of redundant ("removing it does not change anything") is not an acceptable answer. Hint – think about the neighborhood of redundant points.
Part B: (5 Points)

Which of H or V would be preferred as an initial split for a decision (identification) tree? Justify your answer numerically.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>-(x/y)*lg(x/y)</th>
<th></th>
<th></th>
<th>-(x/y)*lg(x/y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.50</td>
<td>1</td>
<td>8</td>
<td>0.38</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.53</td>
<td>3</td>
<td>8</td>
<td>0.53</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.39</td>
<td>5</td>
<td>8</td>
<td>0.42</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.50</td>
<td>7</td>
<td>8</td>
<td>0.17</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.31</td>
<td>1</td>
<td>9</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0.46</td>
<td>2</td>
<td>9</td>
<td>0.48</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.53</td>
<td>4</td>
<td>9</td>
<td>0.52</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0.44</td>
<td>5</td>
<td>9</td>
<td>0.47</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0.26</td>
<td>7</td>
<td>9</td>
<td>0.28</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>0.43</td>
<td>8</td>
<td>9</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.53</td>
<td>1</td>
<td>10</td>
<td>0.33</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.22</td>
<td>3</td>
<td>10</td>
<td>0.52</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0.40</td>
<td>7</td>
<td>10</td>
<td>0.36</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>0.52</td>
<td>9</td>
<td>10</td>
<td>0.14</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 37

Problem 1: Classification (40 points)

The picture above shows a data set with 8 data points, each with only one feature value, labeled f. Note that there are two data points with the same feature value of 6. These are shown as two X's one above the other, but they really should have been drawn as two X's on top of each other, since they have the same feature value.

Part A: (10 Points)

1. Consider using 1-Nearest Neighbors to classify unseen data points. On the line below, darken the segments of the line where the 1-NN rule would predict an O given the training data shown in the figure above.

2. Consider using 5-Nearest Neighbors to classify unseen data points. On the line below, darken the segments of the line where the 5-NN rule would predict an O given the training data shown in the figure above.
3. If we do 8-fold cross-validation using 1-NN on this data set, what would be the predicted performance? Settle ties by choosing the point on the left. Show how you arrived at your answer.
Part B: (8 Points)

Using this same data set, show the decision tree that would be built from this data. Assume that the tests in the tree are of the form \( f \leq c \). For each test show the approximate value of the average disorder for that test. To help you compute this, there's a small table of values of \(- (x/y) \log(x/y)\) for small integer \( x \) and \( y \).

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>-(x/y)*log(x/y)</th>
<th>x</th>
<th>y</th>
<th>-(x/y)*log(x/y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.50</td>
<td>1</td>
<td>8</td>
<td>0.38</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.53</td>
<td>3</td>
<td>8</td>
<td>0.53</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.39</td>
<td>5</td>
<td>8</td>
<td>0.42</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.50</td>
<td>7</td>
<td>8</td>
<td>0.17</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.31</td>
<td>1</td>
<td>9</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0.46</td>
<td>2</td>
<td>9</td>
<td>0.48</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.53</td>
<td>4</td>
<td>9</td>
<td>0.52</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0.44</td>
<td>5</td>
<td>9</td>
<td>0.47</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0.26</td>
<td>7</td>
<td>9</td>
<td>0.28</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>0.43</td>
<td>8</td>
<td>9</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.53</td>
<td>1</td>
<td>10</td>
<td>0.33</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.22</td>
<td>3</td>
<td>10</td>
<td>0.52</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0.40</td>
<td>7</td>
<td>10</td>
<td>0.36</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>0.52</td>
<td>9</td>
<td>10</td>
<td>0.14</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 2: Overfitting (20 points)

For each of the supervised learning methods that we have studied, indicate how the method could overfit the training data (consider both your design choices as well as the training) and what you can do to minimize this possibility. There may be more than one mechanism for overfitting, make sure that you identify them all.

Part A: Nearest Neighbors (5 Points)

1. How does it overfit?

2. How can you reduce overfitting?

Part B: Decision Trees (5 Points)

1. How does it overfit?

2. How can you reduce overfitting?