## Problem #1 – Neural Networks (10 Points)

(a) One method for preventing the neural networks' weights from overfitting is to add regularization terms. You will now derive the update rules for the regularized neural network. Recall that the non-regularized gradient descent update rule for $w_1^{t+1}$ is:

$$w_{ji}^{t+1} = w_{ji}^{t} + \eta \sum_{n=1}^{N} e_j(n)\phi'\left(v_j(n)\right) y_i(n)$$

Derive the update rule for $w_{ji}^{t+1}$ in the regularized neural net loss function which penalizes based on the square of each weight. Use $\lambda \geq 0$ to denote the regularization parameter. Use the following regularizer:

$$R(w) = \lambda \sum_i w_i^2$$

Re-express the regularized update rule so that the only difference between the regularized setting and the unregularized setting above is that the old weight $w_{ji}^t$ is scaled by some constant. Explain how this scaling prevents overfitting.

$w^{t+1} = w^t - \eta \frac{\partial}{\partial w}(E + R)$

$\frac{\partial}{\partial w_{ji}} \{ E + R \}$ taking the derivative w.r.t. $w_{ji}$

$= \frac{\partial E}{\partial w_{ji}} + \frac{\partial R}{\partial w_{ji}}$

$\frac{\partial E}{\partial w_{ji}}$: weight update; $\frac{\partial R}{\partial w_{ji}}$: regularization term

$\frac{\partial E}{\partial w_{ji}} = -e_j(n)\phi'(v_j(n))\, y_i(n)$

$\frac{\partial R}{\partial w_{ji}} = 2\lambda w_{ji}$

- $w_{ji}^{t+1} = w_{ji}^{t} + \eta \sum_{n=1}^{N} e_j(n)\phi'(v_j(n))\, y_i(n) - 2\eta\lambda w_{ji}$
- $w_{ji}^{t+1} = (1 - 2\eta\lambda) w_{ji}^{t} + \eta \sum_{n=1}^{N} e_j(n)\phi'(v_j(n))\, y_i(n)$

The constant $(1 - 2\eta\lambda)$ keeps the weights small and close to 0, and that helps preventing the model from overfitting.

(b) The definition of a sigmoid function is given by

$$\phi(x) = \frac{1}{1+\mathrm{e}^{-x}}$$

Show that $\phi'(x) = \phi(x)(1-\phi(x))$

$$\phi' = \frac{d\phi}{dx} = \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right]$$

$$= \frac{d}{dx}(1+e^{-x})^{-1}$$

$$= -(1+e^{-x})^{-2}\cdot(-e^{-x})$$

$$= \frac{1}{(1+e^{-x})}\cdot\frac{e^{-x}}{(1+e^{-x})}$$

$$= \frac{1}{(1+e^{-x})}\cdot\frac{(1+e^{-x})-1}{(1+e^{-x})}$$

$$= \frac{1}{(1+e^{-x})}\cdot\left[\frac{1+e^{-x}}{(1+e^{-x})} - \frac{1}{(1+e^{-x})}\right]$$

$$= \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right)$$

$$\phi'(x) = \phi(x)(1-\phi(x))$$

## Problem #2 – AdaBoost (10 Points)

In class, we discussed the Adaboost algorithm in four general steps and each step was relatively general. Recall that you were told that the weight for a hypothesis $h_t$ in the Adaboost algorithm was given by

$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$$

Some discussions about Adaboost may simply say "*Choose* $\alpha_t$" without giving a closed form expression for $\alpha_t$. For this problem, prove that the expression shown above is the ideal value for $\alpha_t$ and justify your response.

The overall goal is to minimize the training error of the adaboost algorithm, We know that the upper bound of the training error is given by $\hat{err}(H) \leq \prod_{t=1}^{T} Z_t$.

($Z_t$ is normalization constant.)

Thus, $\alpha_t$ is chosen to minimize $Z_t$.

We know $Z_t = (1-\epsilon_t)\, e^{-\alpha_t} + \epsilon_t e^{\alpha_t}$

taking $\frac{dZ_t}{d\alpha_t} = -(1-\epsilon_t)\, e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 0$

$$\left[-(1-\epsilon_t)\, e^{-\alpha_t} + \epsilon_t e^{\alpha_t}\right] e^{-\alpha_t} = 0 \cdot e^{-\alpha_t}$$

$$-(1-\epsilon_t)\, e^{-2\alpha_t} + \epsilon_t = 0$$



$$e^{-2\alpha_t} = \frac{\epsilon_t}{1-\epsilon_t}$$

$$-2\alpha_t = \log\frac{\epsilon_t}{1-\epsilon_t}$$

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

## Problem #3 – True/False: A Gamblers Ruin (10 Points)

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

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

error on the training data.

**[True/False] (1 point)**: The gradients cannot explode in an RNN, which is a desirable property of the backpropagation through time.

Gradients can explode in RNN.

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

**[True/False] (1 point)**: Classification error is typically the best way to measure the performance of an RNN language model.

usually use perplexity.

**[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.

We can also use unlabled data to train each layer.

**[True/False] (1 point)**: In the context of a adversarial MAB, the term $\gamma \in [0,1]$ controls the trade-off between the estimated reward of the arm and pure exploration.

$$\widehat{p}_i(t) = \gamma \frac{w_i(t)}{\sum_j w_j(t)} + (1-\gamma)\frac{1}{K}$$

The text is correct but the formula is wrong.

$$\hat{p}_c(t) = (1-\gamma)\frac{w_i(t)}{\sum_j w_j(t)} + \frac{\gamma}{K}$$

where $K$ is the number of arms and $w_i(t)$ is the weight of the $i$th arm at time $t$.

$\gamma \approx 1 \rightarrow$ pure exploration

$\gamma \approx 0 \rightarrow$ exploitation

**[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.

A neural network will (likely) find local min.
SVM finds global optimal solution.

**[True/False] (1 point)**: Using a sigmoid activation function in a neural network trained with backpropagation is one way to avoid the vanishing gradient problem.

ReLU reduces vanishing gradient problem.

**[True/False] (1 point)**: A discriminator network, $D$, after enough training in a GAN will always be able to identify if a sample came from the data set or the generator network, $G$.

After enough training, G will generate very realistic data.

**[True/False] (1 point)**: In backpropagation, the only difference between updating a hidden node versus an output node is how the local gradient is calculated.

## Problem #4 – Adaboost (10 Points)

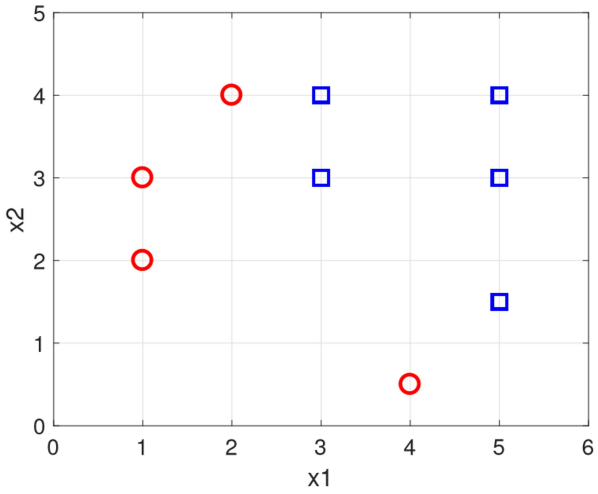



Figure 1: Labeled training points for Problem 2.

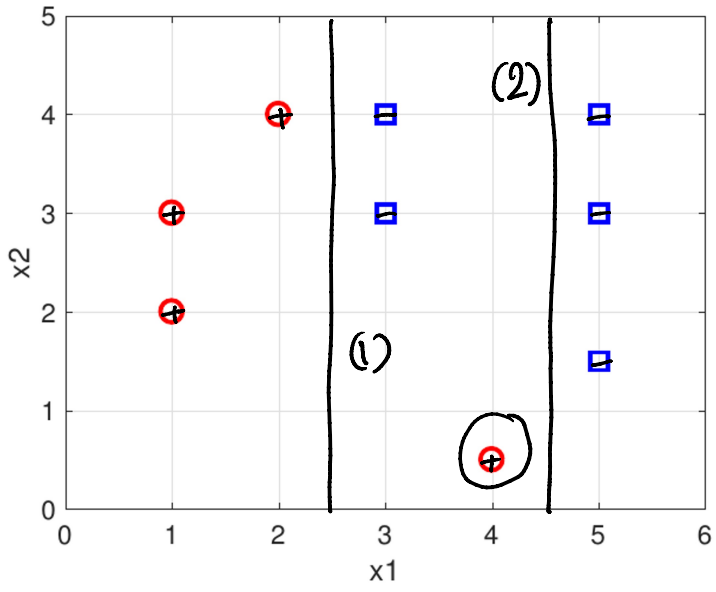
Consider the labeled training points in Figure 1, where + and □ denote positive and negative labels, respectively. We wish to apply AdaBoost with a threshold classifier (i.e., pick an axis then pick a threshold to label the data). In each boosting iteration, we select the threshold that minimizes the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

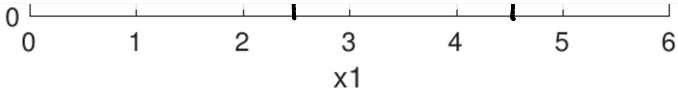
1. In Figure 1, draw a decision boundary on $x_1$-axis (i.e., vertical line) corresponding to the first threshold that the boosting algorithm could choose. Label this boundary (1), and also indicate +/- side of the decision boundary.

figure is below.



(1)








2. In the same figure also circle the point(s) that have the highest weight after the first boosting iteration.

The circled sample has the highest weight.
figure above.

3. What is the weighted error of the first threshold after the first boosting iteration, i.e., after the points have been re-weighted?

$\frac{1}{2}$

4. Draw a decision boundary corresponding to the second threshold using the weights, again in Figure , 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).

Since the circled point has a higher weight, we need to classify it correctly.
figure above.

$\varepsilon_1 = \frac{1}{9}$

$\alpha_1 = \frac{1}{2}\log\left(\frac{1-\frac{1}{9}}{\frac{1}{9}}\right)$

$= 1.03$

$D_2 = \frac{1}{9} e^{-1.03}$ correct

$= 0.0397 \ (\times 8) \Rightarrow 0.063 \times 8$

$D_2 = \frac{1}{9} e^{+1.03}$ incorrect

$= 0.311 \Rightarrow \boxed{0.495} \times 1$

## Problem #5 – Random Short Answer (10 Points)

(SA:1) Describe the process of learning and testing a random forest on a data set with $n$ samples and $p$ features.

1. Create a random sample of the original training data set with replacement aka bootstrapped dataset of size n samples
2. Create a decision tree using the bootstrapped dataset.
3. Repeat step (1) and (2) T times and that will be the number of trees in the forest.
4. To predict a new data point, we run it down all trees created and keep track of the results. The prediction is the class with majority votes.
5. To evaluate random forest, use out-of-bag (OOB) data. The data left in step (1).

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

The appropiate way to train deep neural network is through unsuppervised pre-training and then fine tuning with backpropagation. Some methods that can be used are autoencoders and Restricted Boltzmann Machine (RBM).

(SA:3) Explain the differences between an adversarial bandit and stochastic bandit. Also, describe the concept of regret.

- An adversarial bandit gives more control over exploration and exploitation. Meaning adversary has control of reward distribution and it can also choose the arm that leads to large reward.
- In Stochastic bandit, we converge to a particular arm that gives the best reward based on learning the distribution of the reward.
- The regret is the difference between optimal selection strategy to choose an arm and the strategy we are using.

(SA:4) Explain the difference between backpropagation and backpropagation through time.

The backpropagation is an algorithm that is used to update the weights of neural network by propagating the output error back through the hidden layers (network) in order to minimize the error.

The backpropagation through time is roughly the same algorithm and it is specifically applied to recurrent neural network (RNN). In RNN, the network is unravelled through time, and now the error signal needs to be propagated back through time as well.