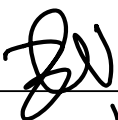


## ECE523: Engineering Applications of Machine Learning and Data Analytics

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

Name: Bader Jeragh

Signature: 

Date: 22/4/2020

**Instructions:** There are five problems. Partial credit is given for answers that are partially correct. No credit is given for answers that are wrong or illegible. Write neatly.

Problem 1: \_\_\_\_\_

Problem 2: \_\_\_\_\_

Problem 3: \_\_\_\_\_

Problem 4: \_\_\_\_\_

Problem 5: \_\_\_\_\_

Total: \_\_\_\_\_

## Quick note about answers submitted for the exam

This exam is a take home. You may use your notes and textbook to help you answer the questions on the exam; however, you are not allowed to use help from other students or the *internet* (this includes Greg's video lectures). Any violation of that rule is a severe breach in trust and academic integrity. Finally, you must answer each of the questions in your own words. This means that you should not be copying and pasting from my lecture or scribe notes to answer the questions.

*Good luck!*

## 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:

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

- 1) 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:

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

- 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.

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

$$\Delta w_{ji}^{(t)} = -\eta \frac{\partial E}{\partial w_{ji}}$$

Differentiating error with respect to  $w_{ji}$ :

$$- E(n) = \sum_{j=1}^L E_j(n)$$

$$- \frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_i(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$- \text{gradient: } \delta_j(n) = \frac{-\partial E(n)}{\partial y_j(n)} \cdot \frac{\partial y_i(n)}{\partial v_j(n)}$$

Thus, the update rule for  $w_{ji}(t+1)$  is:

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

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

The  $(1 - 2\eta\lambda)$  term keeps the weights closer to 0, and this prevents overfitting.

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

$$\phi(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \quad (\text{Should've done this sooner but oh well})$$

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

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

$$\left[ \left( \frac{1}{u(x)} \right)' = \frac{u'(x)}{u(x)^2} \right]$$

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

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

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

$$\therefore \phi'(x) = \frac{e^x}{(e^x + 1)^2}$$

Second,

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

$$= \frac{e^x}{e^x + 1} - \left[ \left( \frac{e^x}{e^x + 1} \right)^2 \right] = \frac{e^x}{e^x + 1} - \frac{e^{2x}}{(e^x + 1)^2}$$

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

$$\therefore \phi(x)(1 - \phi(x)) = \frac{e^x}{(e^x + 1)^2}$$

Thus,

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

→ final answer

## Problem #2 – GANs (10 Points)

Describe what a generative adversarial network is and how they can be used.

A generative adversarial network when two neural network models, called generative and discriminative, compete against each other. The generative model generates fake data while the discriminative model tries to distinguish the real data from the fake data. This continues until the fakes are indistinguishable from the originals. Generative adversarial networks can be used to produce images that are close to reality, such as homes, natural structures like mountains, and clothes.

### Problem #3 – Random Short Answer (10 Points)

(SA:1) In the context of a Multi-Arm Bandit, describe the difference between regret and pseudo-regret. Why is it generally easier to work with pseudo-regret than regret? Explain the differences between an adversarial bandit and stochastic bandit. Feel free to support your response with equations.

Regret is the maximum reward from the optimal strategy machine minus the reward you gained from your selection strategy. The equation for regret is:

$$\max_{i \in [K]} \left\{ \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right\}$$

Pseudo-regret is the expected regret over multiple trials,

and the equation for pseudo-regret is:

$$\max_{i \in [K]} E \left[ \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right]$$

Pseudo-regret is generally easier to work with because the machine that gives the most reward will be likely be different from trial to trial, so it can be hard to analyze regret.

An adversarial bandit gives more control over exploration and exploitation while the stochastic bandit uses the upper confidence to choose the machine with the best reward.

(SA:2) Describe the two step optimization approach used in the  $k$ -means clustering algorithm.

We want to find  $r_{ik}$  and  $\mu_k$  in:  $\min \sum_{i=1}^n \sum_{k=1}^K r_{ik} \|x_i - \mu_k\|_2^2 = \min J$

General algorithm:

1) Minimizing  $J$  with respect to  $r_{ik}$  and having  $\mu_k$  be fixed  
We do this by  $r_{ik} = \begin{cases} 1, & \text{if } \arg\min \|x_i - \mu_k\|_2^2 \\ 0, & \text{otherwise} \end{cases}$

2) Minimizing  $J$  with respect to  $\mu_k$  and having  $r_{ik}$  be fixed  
We do this by:  $\mu_k = \frac{\sum_{i=1}^n r_{ik} x_i}{\sum_{i=1}^n r_{ik}}$

(SA:3) In class, we discussed bounding the risk of a model using Hoeffding's inequality and the union bound. Describe why we are used the supremum in the bound below.

$$\mathbb{P} \left( \sup_{f \in \mathcal{F}} \left| \widehat{R}_n(f) - R(f) \right| \geq \epsilon \right) \leq 2N \exp(-2n\epsilon^2)$$

We don't want  $f$  to be near infinite or infinite, so we use the supremum to take the smallest value that is bigger than all  $f$  in  $\mathcal{F}$ .

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

Backpropagation algorithm modifies the weights of the neural networks to minimize error of output.

Backpropagation through time is the roughly the same algorithm applied to a recurrent neural network. You unfold the network through time, which has the inputs and outputs. The backpropagation algorithm is used to find the gradient of the cost.

(SA:5) We discussed semi-supervised learning, and in particular, self-training and co-training. Describe the differences between them and provide a real-world example for each one of the algorithms. *Note that you will not get credit if you answer with an example we provided in class. Think outside the box!*

Semi-supervised learning uses both annotated and un-annotated data to classify data.

Self-training uses more of the unlabeled data in an iterative process. For example, you take pictures of lions, annotate a small subset of those images, and train on the annotated subset.

Co-training is when you use two classifiers on two (ideally independent) features of the data, so as they can correct each other. For example, one could classify lions based on the shape of the animal and another based on the color.



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

- (1) ~~True~~/False] (1 point): Semi-supervised learning is guaranteed to improve the performance of a classifier that is trained using the unlabeled data.
- (2) True/~~False~~] (1 point):  $k$ -means clustering places data into groups; however, the groups do not necessarily mean the cluster is a class.
- (3) [True/~~False~~] (1 point): SMOTE generates data that lie outside the convex hull of the minority class data, which is why SMOTE is so effective.
- (4) ~~True~~/False] (1 point): Passive-Aggressive online learning will never perform and update to  $\mathbf{w}_t$  if a data  $\mathbf{x}$  is correctly classified.
- (5) [True/~~False~~] (1 point): The VC-dimension allows us to bound on risk even when  $\mathcal{F}$  is infinite.
- (6) [True/~~False~~] (1 point): The gradients cannot explode in an RNN, which is a desirable property of the backpropagation through time.
- (7) [True/~~False~~] (1 point): Classification error is typically the best way to measure the performance of an RNN language model.

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

(9) **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,  $G$ .

(10) **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 #5 – Recurrent Neural Networks (10 Points)

Answer the following questions about recurrent neural networks:

1. Describe the backpropagation through time algorithm and why it is different than the standard backpropagation algorithm.
2. In the context of a language model, why is using a class of words needed to train the model?
3. Outside of the language model, provide an example where they can be used.

1) Same as 3.4 ...

Backpropagation algorithm modifies the weights of the neural networks to minimize error of output.

Backpropagation through time is the roughly the same algorithm applied to a recurrent neural network. You unfold the network through time, which has the inputs and outputs. The backpropagation algorithm is used to find the gradient of the cost.

- 2) Because words are the fundamental building blocks of sentences, and they are necessary for languages. Without recognizing words, communication in that language becomes nigh-impossible.

- 3) An example of recurrent neural networks being used in application may include robotics, such as controlling motors.

## Scratch Paper (not graded)