ECE1513H - Winter 2020 **Final**
University of Toronto - LEC0101 (Prof. Papernot) **April 8**

Please read all instructions carefully. The structure of this take-home exam differs from previous assessment formats used in this course. Follow carefully all instructions to avoid being penalized unnecessarily during grading.

Honor code. Due to the circumstances, this exam is being conducted remotely. The goal of this final is for you to put together the concepts we learned throughout the semester. For this reason, I ask that you complete the exam alone. You are free to use your lecture notes (i.e., this is an open book exam) but please do not use external resources (in particular the Internet). If a request to regrade a specific question is later received, the instructor may reassess any of the questions in the exam in addition to the specific question for which the request was received. Points may be lost if the instructor grades a question lower than TAs originally did.

Instructions. You have 24 hours to turn in your exam. **Given the extended amount of time that you have to turn in your exams, any exam received after 5.30pm Eastern time on April 9 will not be graded.** Please plan accordingly to have your file uploaded in time. To facilitate grading, please follow the following guidelines:

- PDF will be the only accepted format to upload your answers.
- You should submit a single PDF with all of your answers in the same order than this handout.
- Each page should be easy to read and oriented properly.
- If you decide to handwrite your exam rather than typeset it, ensure your handwriting is readable otherwise TAs will have the discretion to not grade your answer.
- Clearly state any questions that you skip by writing down the question number along with "I skip this question"
- Graphs produced should be clearly interpretable. Include labels on axes and a legend.
- Attach a python script (.py) for the questions that require handing in code.

Exam goal. This exam is structured as a series of coherent ML problems where you will go from data visualization all the way to evaluating the ML model you created. I hope this will further motivate you to take the exam seriously, given that it will help you acquire methodologies and skills that are in high demand on the job market.

Exam structure. The exam contains 20 questions worth a total of 50 points. I strongly encourage you to complete questions and problems in the order they are asked in this document. Some of the later questions will be missing context if you skip the early questions. **Note that some of the questions are voluntarily open-ended and there may be several correct answers.** The goal of these questions is to encourage you to leverage what we covered in the course so far through lectures and assignments to tackle new problems.

Problem 1 - Data visualization We will consider the UCI ML Breast Cancer Wisconsin dataset. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. You can download the dataset using the function:

`sklearn.datasets.load_breast_cancer`

Unless specified otherwise, questions in Problems 1-3 below refer to the UCI ML Breast Cancer Wisconsin dataset when the word "dataset" is used.

1. (5 points) We would like to visualize this dataset in 2D. To do this, we will use PCA. Implement PCA yourself using the singular value decomposition function included with numpy: `numpy.linalg.svd`. Your PCA function should take in an array containing a dataset and the number of components to be kept by PCA, and return the matrix $U_{[1:k]}$ we covered in class. Hand-in the code for full credit.

2. (1 point) Which of the two pre-processing steps we covered in the lecture on PCA need to be applied to this dataset before we can run the PCA algorithm on this dataset?

3. (3 points) Project the data onto $k = 2$ principal components (1pt), plot the resulting representation of the dataset (1pt), and comment on the graph by explaining the meaning of each coordinate of a point (1pt). Hand-in the code and figure output for full credit. Justify in your answer which data you passed as the input to PCA.

Problem 2 - Clustering

1. (5 points) Implement κ-means yourself. Your function should take in an array containing a dataset and a value of κ, and return the cluster centroids along with the cluster assignment for each data point. You may choose the initialization heuristic of your choice among the two we saw in class. Hand-in the code for full credit.

   From now on, we use κ instead of k to refer to the number of centroids in the clustering algorithm to avoid any ambiguity with the k notation used previously to refer to the number of principal components in PCA.

2. (3 points) Run the κ-means algorithm for values of κ varying between 2 and 7, at increments of 1. Justify in your answer which data you passed as the input to the κ-means algorithm (1pt). Plot the distortion achieved by κ-means for values of κ varying between 2 and 7, at increments of 1 (2pt). Hand-in the code and figure output for full credit.

3. (1 point) If you had to pick one value of κ, which value would you pick? Justify your choice.

4. (1 point) Using results from above on PCA, project the centroids obtained by the κ-means algorithm, for the value of κ you chose at the previous question, on $k = 2$ principal components.

5. (2 points) Update the graph you produced previously to visualize the dataset with PCA by adding the κ centroids (1pt) and coloring each data point based on the cluster it is assigned to by κ-means (1pt). Hand-in the code and figure output for full credit.

Problem 3 - Linear classification

1. (1 point) Propose a split to partition the dataset into training data and test data.

2. (1 point) Can you come up with a reason for why we do not need validation data in addition to test data in this setting?

3. (4 points) Prove that the parameter $w^\star$ that minimizes the mean squared error of a linear regression model $g(x) = xw$ where x is a row vector and w a column vector is $w^\star = (X^T X)^{-1} X^T Y$. In the solution, we used the vectorized notation, so X and Y are the dataset of inputs and labels. In the model definition, x is a single input and $g(x)$ the model's prediction on this single input.

4. (1 point) Using numpy, compute the numerical value of $w^\star$ for our dataset. Hand in code and the vector values with your answer.

5. (1 point) Turn the linear regression model into a classifier by thresholding the application of a sigmoid to the output of the linear regression model. Report the accuracy your classifier achieves on your training and test sets. Hand in code and the accuracy values with your answer.

6. (2 points) Update the graph you produced in Problem 1 to visualize the dataset with PCA. Visualize the predictions of the algorithm by reflecting the model's prediction with the color of each point on the graph. Hand-in the code and figure output for full credit.

Problem 4 - Neural networks

1. (2 points) We covered multiple optimizers throughout the semester. Some of them are *adaptive*. For instance, Adam adapts the learning rate throughout training, for updates made to each parameter, by using this variant of the gradient descent update rule:

$$w \leftarrow w - \alpha \frac{\hat{m}}{\sqrt{\hat{v}} + \varepsilon} \tag{1}$$

   where α is the learning rate, $\hat{m}$ an estimator of the first moment (mean) of the gradients, $\hat{v}$ an estimator of the second moment (uncentered variance) of the gradients, and ε a small constant to avoid division by zero. In light of this, do you think it is important to fine-tune the learning rate for Adam? Justify your answer.

2. (2 points) We would like to train a neural network on a large dataset. If our machine only has one GPU accelerator, what is the best variant of gradient descent to use: batch gradient descent, stochastic gradient descent, or minibatch stochastic gradient descent? Describe how you would set any hyperparameter that is **specific** to the variant you choose. Justify your answer.

3. (4 points) To speed-up wall-clock time, we are given access to multiple GPUs. These GPUs can be used in parallel. If our machine has multiple GPU accelerators attached to it, what is the best variant of gradient descent to use: batch gradient descent, stochastic

gradient descent, or minibatch stochastic gradient descent? Describe how you would allocate the data between GPUs and how you modify the gradient descent algorithm to synchronize intermediate computations performed by each GPU. Also describe how you would set any hyperparameter that is **specific** to the variant you choose. For the purpose of this question, you can assume for simplicity that the GPU gives you an efficient implementation of the forward and backward passes needed in back-propagation: you send training examples to the GPU and it returns average gradients of the loss with respect to model parameters. These gradients can then be used to update model parameters. Justify your answer.

Problem 5 - Generalization

1. (5 points) Someone describes to you the following process for training a feedforward neural network on MNIST:

   - Split the training set in 3 equal splits D_1, D_2, D_3.
   - Split the test set in 3 equal splits T_1, T_2, T_3.
   - Consider a feedforward neural network with a single hidden layer of 5 neurons. Set momentum to 0 and train the neural network with varying learning rates on D_1. Select the learning rate value α that maximizes the performance of the neural network on T_1.
   - Using the learning rate α from the previous step, train the feedforward neural network again with varying momentum rates on D_2. Select the momentum rate η which maximizes the performance of the neural network on T_2.
   - Using the learning rate α and momentum rate η, train the feedforward neural network again with varying number of neurons on the hidden layer on D_3. Select the number of neurons which maximizes the performance of the neural network on T_3.

   What do you think about this training process? Justify your answer.

2. (2 points) A confusion matrix A_{ij} is such that component (i, j) indicates the number of test points from class i for which the classifier predicted class j. When would such a way to measure performance be preferable to reporting the accuracy of the classifier? Justify your answer.

3. (4 points) You are given a classification dataset $(X, Y) = \{(x_i, y_i) : i \in 1..n\}$ and you make a proposal for a machine learning algorithm. To convince your friend that it is a good learning algorithm, you want to measure its bias and its variance. Come up with an algorithm for measuring bias (1pt) and an algorithm for measuring variance (1pt). Justify your two algorithms (1pt each).

*
* *