# Assignment (4)

*Artificial Intelligence and Machine Learning (24-787)*

### Due date: 2020/11/06 @ 11:59 pm EST

### File Submission Structure

All the files must be put into one directory andrewID-HW4/. Convert the jupyter notebook to a pdf file. Ensure that the submitted notebooks have been run and the cell outputs are visible: Restart and Run All option in the Kernel menu. Zip the directory andrewID-HW4/ and submit the zip on canvas. If an assignment has theoretical and mathematical derivation, scan your handwritten solution and make a PDF file. Any deviations from the submission structure shown below would attract penalty to the assignment score. Please use Piazza for any questions on the assignment.

**andrewID-HW4/**
**q1.ipynb**
**q1.pdf**
**q2.ipynb**
**q2.pdf**
**q3.pdf**
**q3.ipynb**
**q3.pdf**
**q4.pdf**

## PROBLEM 1

**Topic: PCA without skLearn** **[35 points]**

**a) Load and Standardize- 5 points:** Write a function to load the dataset from q1-data/features.npy and normalize it with mean 0 and 1 standard deviation.

**b) Eigen-decomposition- 5 points:** Write a function that takes standardized dataset as the input. The function should compute the Covariance Matrix and return the sorted eigenvalues in descending order and the corresponding eigenvectors. You can use **np.linalg.eig** to compute eigenvectors.Include the eigenvalues in your report.

**c) Evaluation- 10 points:**Write a function that iterates through dimensionality k from 1 to the number of dimensions in the input features and computes the variance explained on reducing the dimensionality to k. Use the function created in part b. Print the variance explained as a function of k. Include the eigenvalue for each k in your report. Which value of k would you pick and Why?

**d) Visualization- 10 points:** Write a function that projects the original data to a 2-dimensional feature subspace. Load the labels from q1-data/labels.npy and plot the 2-D representation as a scatter plot with labels as legends of the plot. If everything goes well, you'll get a figure like below:

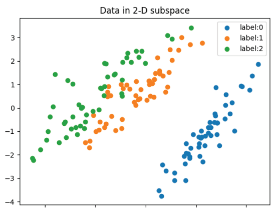

*Fig. 1:* Data in 2D Subspace

**e) Theory- 5 points:**
1. If the number of features is 1000 and the number of data is 10, what will be the dimension of your covariance matrix? Can you suggest what can be changed to improve the performance?

2. Assume you have a dataset with the original dimensionality as 2 and you have to reduce it to 1. Provide a sample scatter plot of the original data (less than 10 data) where PCA might produce misleading results.
Hint: PCA fails to find non-linear structure in data.

## PROBLEM 2

**Topic: Scikit-learn** **[30 points]**

Amino acids are organic compounds that constitute protein in our bodies. In this problem, you are given a dataset which contains data for 20 different types of amino acids. For each type of amino acid, two features are extracted: ionic current in picoamps, and residence time in picoseconds.
Your job is to run various machine learning algorithms on the data using scikit-learn. However, do not use the whole dataset for training, split it into training dataset and testing dataset, accounting for 70% and 30% respectively. You may do this only once and use them for all the models in this problem. Some useful methods for this problem are: **pandas.read_csv**, **sklearn.model_selection**, **train_test_split**, **numpy.meshgrid**, **pyplot.pcolormesh**
For each question that asks to apply a machine learning algorithm, you need to print accuracy on both training and testing datasets in percentage; you also need to plot all the data (training + testing), using only one color, as well as the decision boundary on the same figure. You can use any color as long as it distinguishes the data .

**a) Data Preprocessing- 5 points:** The given data is not very well organized in that it doesn't follow the form of nExamples x (nFeatures + 1). The first row is a list of labels, and the two columns below each label are the data of the two features for that acid type. Load the data and rearrange them so that x of shape (2000, 2) represents the examples and y of shape (2000,) represents the labels. Note: you need to map the string labels to numeric values so y consists of 2000 integers in the range [0, 19]. After you get x and y, split the data into training set and testing set. Save the preprocessed data as a new csv file as you may need that for problem 3

**b) K-Means- 10 points:** Although you have been given the labels, let's pretend we are doing unsupervised learning for the purpose of using KMeans. Train the training set, setting number of clusters to 20. In the first figure, plot the decision boundary and data . Also plot centroids using red stars in the same figure. In another figure, scatter plot all the data using different colors for different classes

**c) Random Forest- 10 points:** Train the training dataset with random forest, leaving the number of estimators as default. In the first figure, plot decision boundary and data . Report accuracy. In the second figure, plot the accuracy on testing set in percentage against the number of estimators which is in range [1, 25].

**d) Analysis- 5 points:** After applying different machine learning algorithms to this dataset, what have you noticed? Concisely write your findings and provide your reasoning. You can think of it from any perspective, such as accuracy, performance, decision boundary shapes and so on.

## PROBLEM 3

**Topic: Neural Net** **[40 points]**

Use the data saved from problem 2a for this question. Use any auto differentiation libraries that you may prefer like Keras, Tensorflow, Pytorch . The problem expects you to implement a Multi Layer Perceptron(MLP) but is open ended in terms of the number of hidden layers, learning rate, activation functions. Choose the hyperparameters such that you get test accuracy above 85%. The train-test split to be used for the dataset is 80 % - 20% which means that you must train your model on 80% data and test it on 20% data. Please note that full credits will be awarded to the submission that has achieved a test accuracy of 85 %. To get you started we will be taking a recitation on Keras with Tensorflow backend on 10/30/2020

# PROBLEM 4

**Topic: Feed Forward and Back Propagation** **[35 points]**
**Submission Instructions**: Please submit a typed pdf file or a clearly hand written scan pdf file for Problem 3.

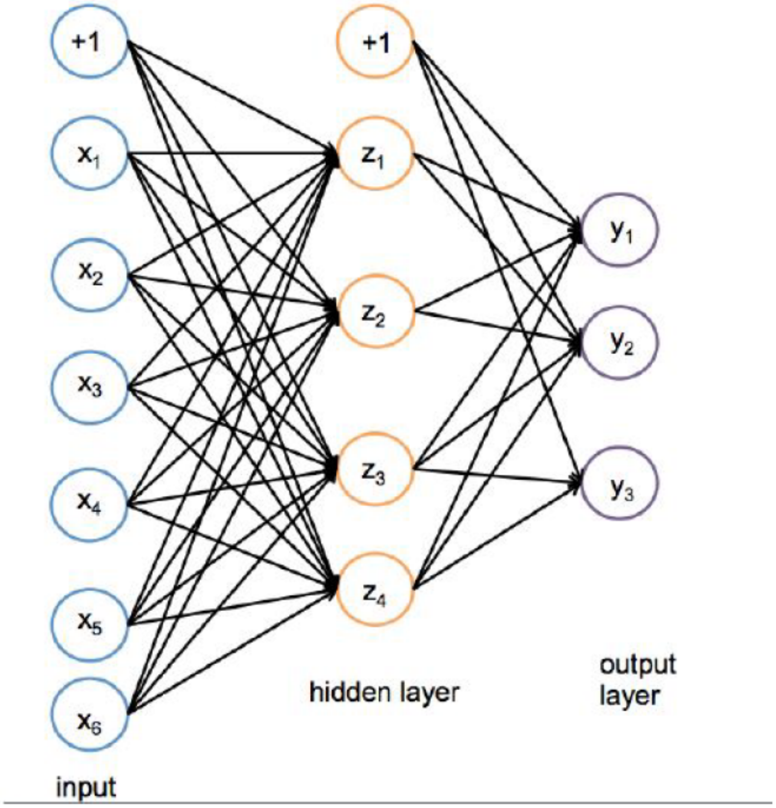

*Fig. 2:* A One Hidden Layer Neural Network

Consider the neural network with one hidden layer. The inputs have 6 features ($x_1$,...,$x_6$), the hidden layer has 4 nodes ($z_1$,...,$z_4$), and the output is a probability distribution ($y_1$, $y_2$, $y_3$) over 3 classes.

We also add a bias to the input $x_0$, as well as to the hidden layer $z_0$ and set them to 1. $\alpha$ is the matrix of weights from the inputs to the hidden layer and $\beta$ is the matrix of weights from the hidden layer to the output layer. $\alpha_{j,i}$ represents the weight going to the node $z_j$ in the hidden layer from the node $x_i$ in the input layer (e.g.$\alpha_{1,2}$ is the weight from $x_2$ to $z_1$ ), and $\beta$ is defined similarly.

We will use a sigmoid activation function for the hidden layer and a softmax for the output layer. Equivalently, we define each of the following:

The input:

$$x = (x_1, x_2, x_3, x_4, x_5, x_6)$$

Linear combination at first (hidden) layer:

$$a_j = \alpha_0 + \sum_{i=1}^{6} \alpha_{j,i} \times x_i, \forall j \in \{1, ..., 4\}$$

Activation at first (hidden) layer:

$$z_j = \sigma(a_j) = \frac{1}{1 + exp(-a_j)}, \forall j \in \{1, ..., 4\}$$

Linear combination at second (output) layer:

$$b_k = \beta_0 + \sum_{j=1}^{4} \beta_{k,j} \times z_j, \forall k \in \{1, ..., 3\}$$

Activation at second (output) layer:

$$\hat{y}_k = \frac{exp(b_k)}{\sum_{l=1}^{3} exp(b_l)}, \forall k \in \{1, ..., 3\}$$

Note that the linear combination equations can be written equivalently as the product of the transpose of the weight matrix with the input vector. We can even fold in the bias term $\alpha_0$ by thinking of $x_0 = 1$, and fold in $\beta_0$ by thinking of $z_0 = 1$.

We will use cross entropy loss, $l(\hat{y}, y)$. If $y$ represents our target output, which will be a one-hot vector representing the correct class, and $\hat{y}$ represents the output of the network, the loss is calculated by:

$$l(\hat{y}, y) = -\sum_{i=1}^{3} y_i \times log(\hat{y}_i)$$

When doing prediction, we will predict the $argmax$ of the output layer. For example, if $\hat{y}_1 = 0.3, \hat{y}_2 = 0.2, \hat{y}_3 = 0.5$ we would predict class 3. If the true class from the training data was 2 we would have a one-hot vector $y$ with values $y1 = 0, y2 = 1, y3 = 0$.

We **initialize the weights** as:

$$\alpha = \begin{bmatrix} 1 & 2 & -3 & 0 & 1 & -3 \\ 3 & 1 & 2 & 1 & 0 & 2 \\ 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & 2 & 1 & -2 & 2 \end{bmatrix}$$

$$\beta = \begin{bmatrix} 1 & 2 & -2 & 1 \\ 1 & -1 & 1 & 2 \\ 3 & 1 & -1 & 1 \end{bmatrix}$$

And weights on the bias terms ($\alpha_{j,0}$ and $\beta_{j,0}$) are **initialized to 1**.

You are given a training example $x^{(1)} = (1, 1, 0, 0, 1, 1)$ with label class 2, so $y^{(1)} = (0, 1, 0)$. Using the initial weights, run the feed forward of the network over this example (without rounding) and then answer the following questions. In your responses, **round to four decimal places (if the answer is an integer you need not include trailing zeros)**. (**Note**: the superscript $(1)$ simply indicates that a value corresponds to using training example $x^{(1)}$)

**Round to four decimal places for all of problem 3**

**1a)** What is $a_1^{(1)}$ and $z_1^{(1)}$ ?
**1b)** What is $a_3^{(1)}$ and $z_3^{(1)}$ ?
**1c)** What is $b_2^{(1)}$ ?
**1d)** What is $\hat{y_2}^{(1)}$ ?
**1e)** Which class would we predict on this example?
**1f)** What is the total loss on this example?

Now **use the results of the previous question** to run backpropagation over the network and update the weights. Use learning rate = 1. Do your backpropagation calculations without rounding, and then in your responses, **round to four decimal places**.

**2a)** What is the updated value of $\beta_{2,1}$?
**2b)** What is the updated weight of the hidden layer bias term applied to $y_1 (e.g. \beta_{1,0})$?
**2c)** What is the updated value of $\alpha_{3,4}$?
**2d)** What is the updated weight of the input layer bias term applied to $z_2 (e.g. \alpha_{2,0})$?
**2e)** Once we've updated all of our weights if we ran feed forward over the same example again, which class would we predict?