Fall 2021—Assignment 4

(Un)Supervised Learning

Using the provided dataset, implement a multi-layer perceptron for classification via supervised learning, as well as the unsupervised k-means and AGNES clustering algorithms. For this assignment, you will use python3 and (optionally) the numpy library for vector/matrix-based scientific computation in python.

The assignment file you will submit is assignment4.py. Please complete the methods within the `class` definitions provided. You are free to add additional helper methods and classes. All the code will be run using the run_assignment4.py file. You can modify this file for your own testing, but you can **only upload the assignment4.py file** to Brightspace, so make sure all your code is in that file. You can expect that any variable in the "run" file could be changed during evaluation though its data type will remain the same. For example, the shape of the data could change (i.e. there may be a different number of points or features). If you **don't hardcode values**, you should be fine.

During evaluation, we will let the "run" file run for up to 1 minute, but you should really aim for under 10 seconds total (assuming an average modern laptop). Using numpy arrays is not required, but it is recommended. Numpy provides many vector operations that you can run over data (often replacing costly for-loops with parallelized one-line operations). These can help keep your code clean and simple while massively improving performance. The numpy documentation may be a helpful reference.

Your code should not print anything to the console when you submit your assignment.

Data

The attached csv file contains all the data. The run file handles importing it and converting it to numpy arrays. A description of the dataset is in the run file.

Algorithms

Multi-Layer Perceptron (MLP)

Much of the MLP is already implemented for you. Please look through the code and try to understand it. What happens if you comment out the line that shuffles the dataset before training? The MLP class calls a fully connected layer ("FCLayer") and a Sigmoid layer. You need to implement the functions implementing the forward and backward passes for each. The forward pass is for prediction and the backward pass is for doing gradient descent. The backward-pass function takes the previous gradient as input, updates the layer weights (for FCLayer) and returns the gradients for the next layer. Please follow the example from the slides. MLP will be trained with the learning rate and loss function (Mean Squared Error) provided. You must update the weights, but don't change the shape or type of the numpy array. Also don't modify the MLP class in your final submission. The number of steps are defined in the run file, for each step update the model on a single datapoint.

K-Means:

Use Euclidean distance between the features. Use a maximum number of iterations, t. Choose a k value and use k-means to split data in k clusters. The k value is provided to the k_means class. Please implement the train method, which should return an n-element array (with n the number of data points in X) with the cluster id corresponding to each item. The distance function is provided for you and you can assume all data is continuous. Ties can be broken arbitrarily.

AGNES:

Use the Single-Link method (distance between cluster a and b=distance between closest members of clusters a and b) and the dissimilarity matrix. For this exercise, use k as the number of clusters. Stop when number of clusters == k. The k value is provided to the AGNES class. Use Euclidean distance between the features. The distance function is provided for you and you can assume all data is continuous. Ties can be broken arbitrarily.