CS5361/6361 Machine Learning
Fall 2024
Arrays – Exercise 1

Write the following functions. See the commented out part of the program for expected results. Also, make sure your functions don't modify the array (if any) received as parameter.

1. Write the function is_square(X) that receives an array X and determines if X is square. An array is square if it has two dimensions and they are the same size (that is, it has the same number of rows and columns).

2. The numpy function np.max(X) returns the maximum value in array X. Write the function max_loop(X) that uses a loop to find the maximum value in array X, where X may have any number of dimensions.

3. The numpy function np.argmax(X) returns the index where the maximum value in array X is stored. Write the function argmax_loop(X) that uses a loop to find the argmax in 1D array X.

4. The numpy function np.sum(X) returns the sum of the elements in array X. Write the function sum_loop(X) that uses a loop to compute the sum of the elements in array X, where X may have any number of dimensions.

5. Write the function diagonal(X) that receives a square array X and returns a 1D array containing the elements in the diagonal of X (that is [X[0,0], X[1,1], ...).

6. Write the function count_digits(X) that counts the number of times each of the numbers 0,1,…,9 appears in an array. The function should receive an array of integers X of any dimensionality and return a 1D array of length 10, where the first element in the array is the number of times 0 appears in X, the second element is the number of times 1 appears in X, and so on.

7. The numpy function np.dot(x1,x2) receives two 1D arrays x1 and x2 and returns their dot product. Write the function dot(x1,x2) that computes the dot product of x1 and x2 without using the numpy function.

8. Write the function euclidean_dist(x1,x2) that receives 1D arrays x1 and x2 and returns the Euclidean distance from the point represented by x1 to the point represented by x2.

9. Write the function manhattan_dist(x1,x2) that receives 1D arrays x1 and x2 and returns the Manhattan Euclidean distance from the point represented by x1 to the point represented by x2.

10. Write the function accuracy(p,y) that receives 1D arrays of integers p and y, where (p.shape == y.shape) and returns the accuracy of a classifier whose predictions are given by p and where the correct classification is given by y.

11. Write the function mse(p,y) that receives 1D float arrays p and y, where (p.shape == y.shape) and returns the mean-squared error of a model whose predictions are given by p and where the correct target value is given by y.

12. Write the function select_features(X,n) that receives 2D array X representing a dataset and an integer n and returns a subset of X that contains all instances in X but only the features with highest variance (you may use the np.var numpy function).

13. Write the function select_instances(X,f,t) that receives 2D array X representing a dataset, an integer f representing a feature (or column number) in X and a floating point value t and returns a subset of X containing only the instances where feature f is larger than t.

14. Write the function nearest_neighbor(X,x) that receives a 2D array X of shape (n,c) and a 1D array x of shape(c,) and returns the index of the row in X that is most similar (according to Euclidean distance) to x.

15. Write the function nearest_neighbors(X1,X2) that receives 2D arrays X1 and X2, where (X1.shape[1] == X2.shape[1]) and returns a 1D array N of shape X2.shape[0], where N[i] is the instance in X1 that is most similar to X2[i] according to Euclidean distance. Do this using at most one loop; for extra credit do it with no loops.