Homework assignment # 2

Due:

IMPORTANT NOTE #1 When you are asked to explain something *concisely*, *clearly*, and/or *briefly*, *please do exactly that.* Lengthy answers not to the point will not earn you extra credits but will cost you more time. Also, *please write neatly and legibly.* Poor penmanship that is hard to decipher will not earn you any sympathy points.

IMPORTANT NOTE #2 You can dicuss the problems with your fellow students, TAs and instructor, but all answers and codes written down and turned in must be your own work.

These notes apply to all assignments in this class.

Caveats: The concept of dimension reduction is an important one in machine learning but will not be covered in detail in lecture. This assignment will be explained by the TAs in discussion sessions and for you to self study. For example, consult Duda, Hart and Stork, Sec. 3.8 on Component Analysis and Discriminants.

The concept of classification (or telling different classes apart) is often important in visualization. If you have classes represented by feature vectors of length k and $k > 3$, it is difficult to visualize how class samples are placed in the high-dimensional space and where the separating hyperplane should be drawn.

Often times, you would like to "map" or "project" such high-dimensional features into a lower-dimensional space (most commonly one or two dimensions) so that you can observe how, or if it is at all possible, to separate these class samples.

Many such dimension-reduction techniques exist. We will examine here the simplest one which maps high-dimensional feature vectors linearly onto a one-dimensional space (or a line). The technique is represented mathematically as (for the i-th sample)

$$g^i = \sum_{j=0}^{k-1} w_j f_j^i + b$$

where $f^i = (f_o^i, \cdots, f_{k-1}^i)$ is the original k-dimensional feature vector for sample i, $w = (w_o, \cdots, w_{k-1})$ are the mapping coefficients, b is the mapping bias, and g^i is the new 1-D feature vector.

Not all choices of w_j and b will result in a "good" mapping. Goodness is often defined to be function of class separation after mapping. That is, in the lower-dimensional space, we still want samples from different classes to be well separated if at all possible. For example, in Figure 1 where a 3-D data set, comprising class samples from three classes, is mapped to a plane. Then it should be evident that W_1 mapping on the left preserves class separation much better than W_2 mapping on the right.

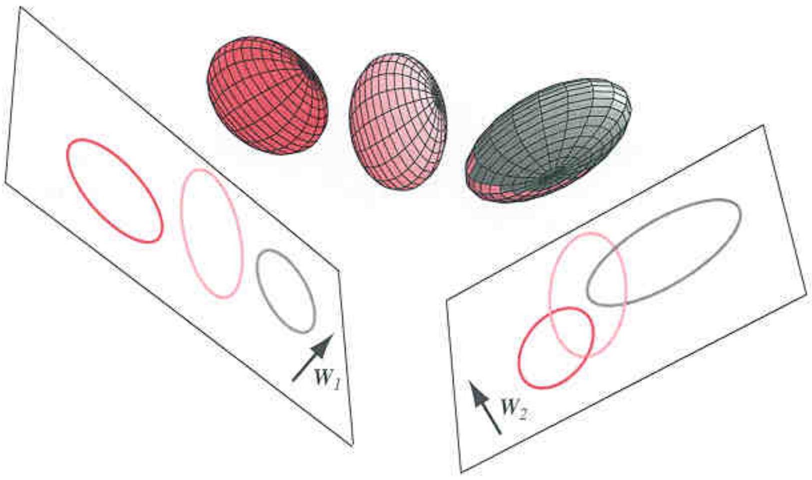

Figure 1: Different linear projections can produce different amounts of separation of class samples.

For a 2-class problem, the linear mapping to best preserve class separation is achieved by Fisher linear discinimant function. The optimization criteria is that after the mapping, the classes should still be well separated. One way to measure the separation is to measure the distance between the class means, or we would like the class means D after mapping or projection to be as large as possible

$$D = |w \cdot m_1 - w \cdot m_2]$$

where

$$m_1 = \frac{1}{|C_1|} \sum_{i \in C_1} f^i$$
$$m_2 = \frac{1}{|C_2|} \sum_{i \in C_2} f^i$$

We will derive such a formulation step-by-step below:

(1). One problem with this reduction (mapping) operation is that when different components in the original feature vectors are correlated, the mapping can become erratic. Based on what we learned about correlated features in regression, what do you think is a good mechanism to combat instability of solution due to feature correlation? However, note that you cannot stabilize the solution by setting w to zero like in regression.

(2). Write down the mathematical formulation of such a "stabilizing" formulation and solve the equations. Can you explain your solution succinctly but clearly?

(3.) However, keeping the projected means apart is not enough, as the data can spread out unevenly along different dimensions (or axes). So an additional constraint is often used to require that the class data do not spread out after the projection. Now define the sample variance after projection as *scatter* (basically, class variance in the projected 1D space). Give the mathematical expression of the scatter.

(4.) Intutively speaking, a good mapping is one that (a) keeps the projected means apart and (b) keeps the projected scatters small (i.e., different class members are separated apart while the same class members remain close together after mapping). How do you combine the two requirements together (large separation of projected means and small data variances or scatters for each class)? **Hint:** Fisher uses a quotient expression that you can find in Duda, Hart and Stork.

(5.) Read section 3.8 in Duda, Heart and Stork. Then summarize in your own words how Fisher's linear discriminant function chooses the projection direction $\mathbf{w}$ (a picture here will be helpful)

$$\mathbf{w} = \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

(6). Another way to reduce feature dimensions is to use SVD — the "magic" transformation that will pick the most important feature dimensions for you with one function call in Linpack (please review your linear algebra or check out Gilbert Strang's Youtube videos on SVD). Explain why SVD is not a good dimension-reduction technique as far as classification problems are concerned.

Programming exercise

The programming part of the assignment is for you to extend/modify the linear regressor in assignment #1 based on iterative stochastic gradient descent (SGD) with regularization into a logic regressor. You must use Python for this assignment (so the reader can grade your assignment on machines in CSIL), and you must implement the iterative solver yourself (not calling an existing package). Again, your solver must support 2D grid search with one dimension being the learning rate α and the other dimension being the regularization weight λ. The input/output parameters of your logistic regression function and the phases of operation should follow those in hw#1.

For comparison purpose you should run through the same data with an SVM. You do NOT need to implement SVM yourself, but can call a public-domain one (e.g. in scikit-learn). The exercise is to compare the run time and performance of a logistic regressor based on SGD with the SVM based on maximizing margin.

We will experiment with two data sets on quality of red wine and white wine `https://archive.ics.uci.edu/ml/datasets/wine+quality`. There are 11 input parameters and one output parameter. The input parameters are: 1 - fixed acidity, 2 - volatile acidity, 3 - citric acid, 4 - residual sugar, 5 - chlorides, 6 - free sulfur dioxide, 7 - total sulfur dioxide, 8 - density, 9 - pH, 10 - sulphates, and 11 - alcohol. The output variable (based on sensory data) is quality (score between 0 and 10).

Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g., there is no data about grape types, wine brand, wine selling price, etc.).

To simplify the classification task, we will group all red wines and white wines into three categories only: excellent (those with a quality score at or above 7), poor (those with a quality score at or below 4) and median (all others). Note that the grouping results in highly unbalanced classes: there are a lot more median wines than excellent and poor ones. Also, the grouping creates a multi-class classification problem. Your "work-horse" should still be a binary classifier. Then you will use a strategy (one versus all, or one versus one) on top of your binary classifier to get the final class label.

Both the white-wine and red-wine data will be partitioned into training and test sets. You will be provided with five different partitions per each wine category (samples are randomly assigned into training and test sets). You should use only the training sets for training and then validate the accuracy on the corresponding test sets. You should provide a table or a plot of the run time and accuracy of the five runs of your programs as well as the average run time and accuracy for both logistic regressor and SVM under both training and testing configurations.

One last piece of the puzzle is how we define accuracy. There are actually many valid definitions, and we will use the following (not unique): If you imagine a 3×3 matrix where (i, j) entry records how many samples in class i are classified as samples in class j (this is often called the confusion matrix), then mis-classified samples are those off-diagonal entries. So accuracy can be defined as (total number of diagonal entries) / (total number of entries in the matrix). Again, other accuracy definitions exist (e.g., precision vs. recall, false positives vs. false negatives, and area under the curve) and will be covered later in lectures.

Btw, there is no reason to cheat by training on the test sets: as your classifiers will be retrained and graded on a different training-testing partition.