Middle East Technical University　　Department of Computer Engineering

CENG 499

Introduction to Machine Learning

Fall 2015-2016

Homework 1

Due date: Nov 28, 2015, Sat, 23:55

1 Objectives

To familiarize yourselves with the fundamental concepts of statistical learning through implementing Bayesian classifiers that utilize **Maximum Likelihood Method** to estimate parameters of class conditional densities and class prior probabilities, which are then combined to yield class posteriori densities forming the grounds for optimal **Bayesian Decision Making** under particular **loss** schemes. To practice the traditional steps **Machine Learning** methodology and to learn the basics of algorithm implementation, **data visualization**, and **classifier evaluation** on **MATLAB/Octave** framework.

2 Specifications

This homework comprises of two main tasks, Bayesian decision making under zero-one loss scheme and under arbitrary risk matrices, which will be described in detail subsequently.

Training data for this task is included in the file named "**hwdata3d.mlm**", whose first row carries information on the number of attributes, the number of classes, and the number of instances, respectively. Following lines include the values of the first, second, third attributes and the index of the class the instance belongs to. You need to input this file by proper MATLAB functions such as `fscanf` into your own choice of variables to initiate the computation.

Change the default format of the MATLAB workspace into long fixed-decimal format to avoid possible numerical errors.

2.1 PART A: Classification Using Quadratic Discriminant Function

Assume that the given data set is generated by a process that obeys multivariate normal distribution for each class. Estimate parameters of class-conditional densities by using the Maximum Likelihood Method. Estimate class prior probabilities by computing the relative frequency of class members within the whole data set.

You should write a MATLAB **function** that computes the quadratic discriminant for provided classes according to the following formula.

$$g_i(x) = -\frac{1}{2}(\mathbf{x}-\mu_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_i) - \frac{1}{2}\log|\mathbf{\Sigma_i}| + \log P(\omega_i)\,, \quad \forall i \in [1, C]. \tag{1}$$

in which $\mathbf{x}$ is a 3-d vector in the space spanned by real numbers, $(\mu_i, \mathbf{\Sigma}_i)$ are estimated distribution parameters of the class indexed by i, namely the ω_i, and $P(\omega_i)$ is the prior probability associated with this class.

Your function should be free of loops, and should operate on all data entries at one pass. In other words, you must write your code in a **vectorized** fashion so that your function accepts some data design matrix $\mathbf{X}$ populated with the instances of the whole data set, instead of where resides a 3-D vector in equation (1).

Having computed discriminative function outputs, determine the class of each data entry in the given set according to the following decision rule.

$$Assign\ \mathbf{x}\ to\ \omega_i\ \ iff\ \ g_i(\mathbf{x}) > g_j(\mathbf{x})\ ,\ \forall j \neq i. \tag{2}$$

Again, implement the assignment procedure without using repetitive structures.

At this point, determine the **accuracy** as the percentage of correct guesses over all guesses and also output the **number of incorrect guesses**.

Visualization: Now, you should visualize your data set. First, plot each data instance discriminated by distinct color or shape features based on their true class labels. Then, you should mark incorrect guesses by particular features of the plotting utility you adopt to use. Observe how your data points spread throughout the 3-D space. You may use `plot3` function for this task.

On top of your data set, plot decision boundaries for each class tuple that satisfy the following equation.

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0\ ,\ \forall i \neq j. \tag{3}$$

Usually, equation (3) does not have an easily computable analytic solution. Therefore, it is advised to use `isosurface` and `isonormals` functions by providing manually selected intervals in which candidate solution vectors of equation (3) are anticipated to exist.

These intervals come in three dimensions, so a nested-loop of size three seems to be required. However, for vectorized evaluation of class discriminative functions on vectors residing inside given intervals, you can always use `meshgrid` or `ndgrid` functions.

Having applied these built-in MATLAB functions, combine the obtained output matrices in the same fashion as you had organized your data design matrix so that they can serve as inputs to the discriminant function you have implemented. For that, you may make use of `reshape` command or default indexing operations. Then, you can safely get the response values out of the discriminants in question.

You may also use MATLAB function handles to generalize the computation for each separating surface.

An example image is included below.

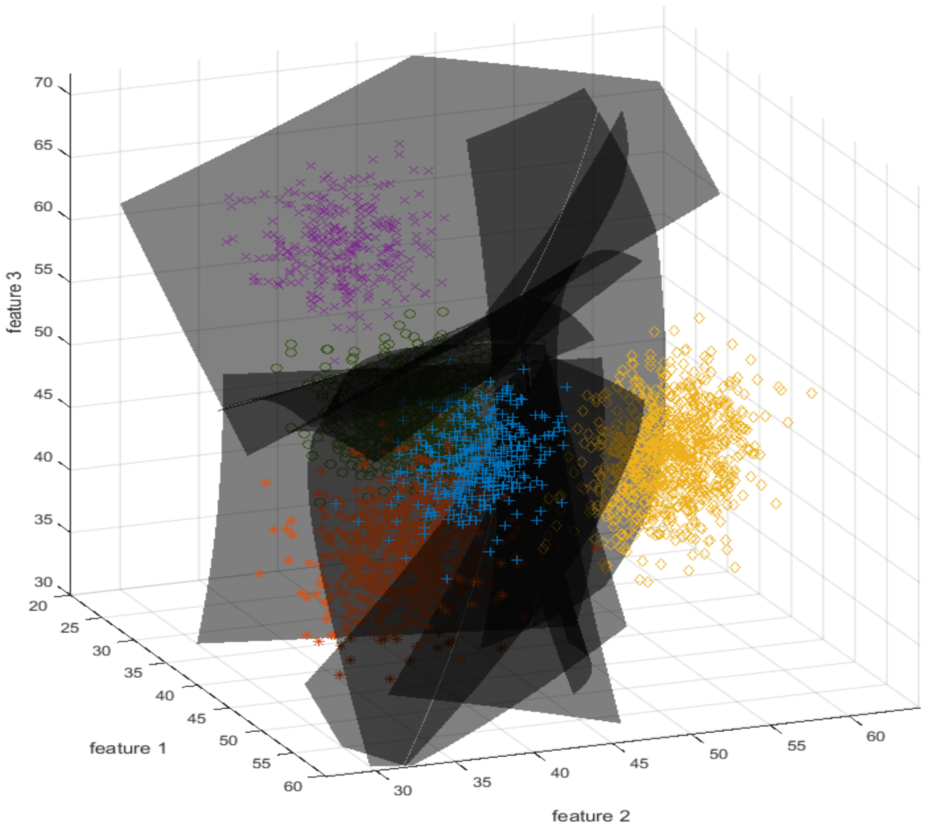

Figure 1: *Decision boundaries are depicted as transparent black hyperquadratic surfaces in this case.*

Note that this is merely a two-dimensional figure, a projection of the 3-D figure captured at a particular viewpoint. Your script should output a MATLAB figure for manual visual inspection during the evaluation. In your report, comment on **why** the instances were separated the way they were.

2.2 PART B: Classification Under Risk

Having implemented the MAP classifier in PART A, now it is time to incorporate arbitrary **risk matrices** into the problem by changing our decision-making rule. Recall that class-conditional densities have the following analytic formula under multivariate (d features such that $d > 1$) Gaussian distribution assumption.

$$p(\mathbf{x}|\omega_i) = \frac{1}{2\pi|\mathbf{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_i)\right], \forall i \in [1, C]. \tag{4}$$

First, implement a MATLAB function that computes the class posteriori densities openly using the Bayesian formula that combines class prior probabilities and class-conditional densities in equation (4) as follows.

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{\Sigma_{j=1}^{C} p(\mathbf{x}|\omega_j)P(\omega_j)}. \tag{5}$$

Use vectorization techniques for the implementation of equation (5) in MATLAB. Your function is expected to compute posteriori probabilities for all classes given the whole training data set at once. You may compute normalization by the marginal density of $\mathbf{X}$, the data design matrix, after the calculation of the nominator of the Bayes' rule.

Bayesian decision rule under risk regimes requires a variation of minimum-error rule in MAP classifier. Loss scheme can be represented by a CxC matrix $\mathbf{R} = [R_{ij}]$ in MATLAB such that $R_{ij} := \lambda(\alpha_i|\omega_j)$. In other words, some data vector $\mathbf{x}$ is observed whose true nature is ω_j, yet action α_i, namely assigning $\mathbf{x}$ to class ω_i is carried out and by doing so some loss quantifiable to the predefined degree R_{ij} is incurred.

You should modify the decision-making procedure in such a way to minimize the total incurred loss. For this purpose you must create a third function for our task that implements the class discriminant in accordance with the subsequent equation.

$$h_i(\mathbf{x}) = -\Sigma_{j=1}^{C}\lambda(\alpha_i|\omega_j)p(\omega_j|\mathbf{x}). \tag{6}$$

This function of yours should compute discriminant function outputs for all classes given the training data set at one pass. Surely, it should use the outputs obtained upon appliance of the previous function.

Now, since we negated the sum of incurred loss for all classes, the assignment rule stays the same, only this time by using the new discriminants:

$$Assign\ \mathbf{x}\ to\ \omega_i\ \ iff\ \ h_i(\mathbf{x}) > h_j(\mathbf{x})\ ,\ \forall j \neq i. \tag{7}$$

To put things in motion, please assign instances in your design matrix to classes predicted by the rule in equation (7), assuming that your **first** risk matrix is set as `R1 = double(~eye(numClasses))`. Compute the accuracy again as a percentage of correct guesses over data set cardinality and also output the number of incorrect guesses. Compare obtained results with the classifier in PART A.

Determine yet **another** risk matrix of your own choice `R2` such that it will result in a change in the separating surface that discriminates two of your favorite classes in this task. Again output the accuracy and wrong guesses. Also visualize the change using `subplot` in a 1x2 grid. You may include only the instances of these two classes for your plot. How did the behavior of the classifier change?

Include all of your findings, comments, and figures in the **homework report**, whose expected basic characteristics are indicated in Submission section.

3 Restrictions and Tips

- Do not use any available MATLAB repository files without referring to them in your report.
- Toolbox function use is not allowed in this homework. Do not use any ML-related toolbox. However, you may cross-check your results utilizing toolboxes. Your homework submission must not include any high-level toolbox function.
- If you encounter trouble regarding vectorization, first try to implement the tasks by using loops. Vectorization is required, since typical ML projects deal with massive amounts of data. If at the end you fail to vectorize your code, submit the current version. Although vectorization appears to be essential in MATLAB programming, since this is not a MATLAB course, unvectorized versions will only result in a minor decline (at most 10%) in the grade you are going to receive.
- You may want to pay attention to difference between biased and unbiased estimators of the covariance matrix.
- For plotting you can encounter various options in the MATLAB repository. Try searching how to plot $f(x, y, z) = 0$, a function of three variables, in MATLAB. Don't forget to cite if you use these codes with/without modification.
- Implementation should be of your own. Readily-used codes should not exceed a reasonable threshold within your total work. In fact, except for plotting you shouldn't need any code on repositories. Consequently, try to devise all these algorithms as if it is the first time on earth you want to form a communication between basic MATLAB environment and fundamentals of statistical learning.
- Don't forget that the code you are going to submit will also be subject to manual inspection.

4 Submission

- **Late Submission:** No late submission is accepted.
- Your scripts and function files together with a 3-to-4 pages long report focusing on theoretical and practical aspects you observed regarding this task should be uploaded on COW before the specified deadline as a compressed archive file whose name should be <<**student_id**>_**hw1**> preceding the file extension.
- The archive must contain **no directories** on top of MATLAB/Octave scripts and function files.

5 Regulations

1. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations.
2. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.