University of Pittsburgh
CS1675 Introduction to Machine Learning
Professor Milos Hauskrecht

Handout 2
September 7, 2017

Problem assignment 1

Due: Thursday, September 14, 2017

Problem 1.

Install Matlab on your computer or access it in one of the CSSD labs.

Problem 2. Exploratory data analysis

In this problem we will explore and analyze the dataset *pima.txt* provided on the course web page. To do the analysis you will need to write short programs. Keep the code you write for future problem sets.

The *pima.txt* is described in the file *pima_desc.txt* The dataset consists of 8 attributes and a binary attribute defining the class label, the presence of diabetes. Data entries are organized in rows such that attributes come first and the class label is last. Answer the following questions with the help of Matlab:

- (a) What is the range (minimum and maximum value) for each of the attributes?
- (b) What are the means and variances of every attribute.
- (c) Calculate and report correlations between the first 8 attributes (in columns 1-8) and the target class attribute (column 9). Use Matlab's corrcoef function to do the calculations. What is the attribute with the highest (positive) correlation to the target attribute? Do you think it is the most or the least helpful attribute in predicting the target class? Explain.
- (d) Calculate all correlations between 8 attributes (using the corrcoef function). Which two attributes have the largest mutual correlation in the dataset?
- (e) Assume we want to predict a target class given all attributes. What do you think, does it help or not in prediction to have 2 attributes that are fully correlated? Explain.

While the analysis using basic statistics as performed above conveys a lot of information about the data and lets us make some conclusions about the importance of attributes or

their relation, it is often very useful to inspect the data visually and get more insight into various shapes and patterns they hide. In the following we will inspect the data using histograms and 2D scatter plots.

- (f) **Histogram analysis** gives us more information about the distribution of attribute values. Write (and submit) a Matlab function *histogram_analysis* that takes the data for an attribute (as a vector) and plots a histogram with 20 bins using Matlab's hist function. Analyze attributes in the data using the function. Answer the following questions. Which histogram resambles most the normal distribution? In your report show at least two histograms, including the choice you picked as the most normally distributed attribute.
- (g) **2D Scatter plots** plots let us inspect the relations between pairs of attributes. Write (and submit) a function *scatter_plot* that takes pairs of values for two attributes and plots them as points in 2D (use Matlab function scatter to do the plot). Analyze the pairwise relations between 8 attributes in the pima dataset using the scatter plot function. Answer the following questions. Is there a scatter plot that indicates possible linear dependency between two variables? Select two random scatter plots and include them in the report. With every plot include the corresponding attribute names.

Problem 3. Data preprocessing

Before applying learning algorithms some data preprocessing may be necessary. To practice Matlab we will write programs for two possible preprocessing tasks: normalization and discretization of continuous values.

- (a) Write (and submit) a function *normalize* that takes an unnormalized vector of attribute values and returns the vector of values normalized according to the data mean and standard deviation. The normalized value should be:

$$x_{\text{norm}} = \frac{x - \mu_x}{\sigma_x}.$$

  where x is an unnormalized value, μ_x is the mean value of the attribute in the data and σ_x its standard deviation. Test your function on attribute 3 of the pima dataset. Report normalized values of the attribute 3 for the first five entries in the dataset.
- (b) Write (and submit) a function *discretize_attribute* that takes a vector of attribute values, a number k (number of bins) and assings each value to one of the k bins. Bins are of equal length and should cover the range of values that is determined by the min and the max operations on the vector. Every bin is given a numerical label such that the smallest value is in bin 1 and the largest attribute value is in bin k. The bin label represents the result of discretization. Test your function on attribute 3 of the pima

dataset. Assume we use 10 bins. Report new (discretized) values of the attribute 3 for the first five entries in the dataset.

Problem 4. Data set splitting

In this problem we practice (a) splitting of the dataset along an attribute value and (b) a random splitting of the dataset into the training and testing sets.

- (a) Split *pima.txt* data into two data subsets - one that includes only examples with class label "0", the other one with class "1" values. Calculate and report the mean and standard deviation of each attribute in these two subsets. Hint: try to use Matlab's function *find* to split the data.
- (b) Write (and submit) a function *divideset1* that takes the dataset (represented as a matrix) and the probability p_{train} of selecting the data entry (a row in the matrix) into the training set. The function should return two non-overlapping datasets: the training and testing data, such that every entry is selected to the training set randomly with probability p_{train}. Test your *divideset* function on the pima dataset. Run the function 20 times with probability $p_{\text{train}} = 0.66$ and report the average length of the training dataset.
- (c) If your code to part b is correct, you should see some variation in the size of the training sets. Write (and submit) a function *divideset2* that takes the dataset (represented as a matrix) and the probability p_{train}, and returns two non-overlapping datasets: the training and testing data, that mimic closely the distribution defined by p_{train}. Basically, your *divideset2* function should decide first on the number of examples that will go into training and test sets and after that choose randomly examples that will go into each set. The algorithm, if you run it, should always give you different training and test sets but their sizes should stay the same for the same dataset.