University of Pittsburgh
CS1675 Introduction to Machine Learning
Professor Milos Hauskrecht

Handout 5
February 25, 2021

Problem assignment 5

Due: Thursday, March 4, 2021

In this assignment we investigate the "Pima" dataset. Recall we performed initial exploratory analysis of the Pima dataset in Homework 1. You can download the dataset (*pima.txt*) and its description (*pima_desc.txt*) using the canvas file link. In addition to the complete dataset *pima.txt*, you got *pima_train.txt* and *pima_test.txt* you will need to use for training and testing purposes. The dataset has been obtained from the UC Irvine machine learning repository:
http://www1.ics.uci.edu/~mlearn/MLRepository.html.

Problem 1. Logistic regression model

In this problem we experiment with the logistic regression model. Please perform the following tasks:

- (a) Familiarize yourself with a batch-mode gradient descent function in file *Log_regression.m*, in which all data points are considered at the same time to calculate the negative log likelihood error.
- (b) Implement and submit a program *main_LogReg.m* that:
  - loads the training and testing data
  - uses functions *compute_norm_parameters* and *normalize* functions given to you to normalize the inputs (for both the training and testing data) based on the data in the training set.
  - runs the gradient descent function given to you on the training dataset for 2000 iteration steps (also called epochs)
  - uses the model to make class predictions on both the training and testing data and calculate the following statistics:
    * Confusion matrices for the train and test sets
    * Training and test misclassification errors
    * Sensitivity and specificity of the model on the test set.

For the definitions of misclassification errors, confusion matrix, sensitivity and specificity please see slides from Class 4 on 01/28/2021.

- (e) Experiment with the learning algorithm by changing initial weights, learning schedule, number of epochs. Report training and test misclassification errors for different settings. What was the best result you could get?

Problem 2. Naive Bayes model

The Naive Bayes model defines a generative classifier model in which all features are independent given the class label. In such a case the class-conditional densities over many input variables can be decomposed into a set of independent class-conditional densities, one for every input variable. For example, the conditional probability of an input $\mathbf{x} = \{x_1, x_2, \cdots, x_d\}$ given class 1 in the Naive Bayes model is decomposed as:

$$p(\mathbf{x}|y=1) = \prod_{i=1}^{d} p(x_i|y=1).$$

One important task when using generative classification models is the choice of an appropriate parameterization of class-conditional densities. Typically we do not choose the distributions arbitrarily, instead we want to make a good educated guess. Exploratory data analysis can help us greatly to recognize types of densities that appear to match the data the best.

Problem 2.1. Exploratory data analysis

We have performed the exploratory analysis of the Pima dataset in Homework 1. Here we reuse the programs created there and apply them to study the density models we choose to parameterize our Naive Bayes model.

Part a. Write and submit a program (*main2_1.m*) that:

- Divides "pima.txt" data into two subsets - one with all examples with class "0", and another with all examples with class "1".
- Analyzes examples in two subsets using histograms. Histograms should give you more information about the shape of the distribution of attributes. You can use the function *histogram_analysis.m* for this purpose.

Part b. What distribution/density would you use to fit the values of attributes 1 to 8 in the pima dataset? Choices one typically considers are Binomial, Multinomial, Normal, Poisson, Gamma, exponential distributions.

Problem 2.2. Learning of the Naive Bayes classifier

The learning of the Naive Bayes model corresponds to the estimation of parameters of class-conditional distributions $p(x_i|y=1), p(x_i|y=0)$ for all input components i from data and estimation of class priors $p(y=1)$, $p(y=0)$. Thus, the learning boils down to a number of 'smaller' density estimation problems.

Assume that class-conditional densities for pima dataset have the following form:

- Class-conditionals for inputs [1 5 7 8] take the form of exponential distribution. The exponential distribution is defined as:

$$p(x|\mu) = \frac{1}{\mu} exp\left[-\frac{x}{\mu}\right],$$

  where μ is the parameter. (Exponential distribution is a special case of the Gamma distribution and belongs to the exponential family).

- Class-conditionals for inputs [2 3 4 6] follow univariate normal distributions:

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right],$$

  with mean and standard deviation being the two parameters.

In addition assume that priors on classes follow a Bernoulli distribution:

$$p(x|\theta) = \theta^x(1-\theta)^{(1-x)} \text{ for } x \in \{0,1\}.$$

Part a. Write and submit a program *main2_2.m* that computes and returns the estimates of the parameters of the Naive Bayes model using the training set *pima_train.txt*. The parameters include the prior probability on classes, and parameters of 16 class-conditionals ($8*2=16$), one for every input component and a class label. To fit exponential distibutions use Matlab function *expfit*; to fit normal distributions use function *normfit* (see also Matlab help).

```
%%% example: application of expfit and normfit functions
%%% class_0 : all inputs (x) with label class 0
%%% class_1 : all inputs (x) with label class 1
%%%
%%% fit the exponential class-conditional for input attribute 1 and class 0
%%%  p(x_1|y=0, \mu_0_1)
[exp_0_1_muhat, exp_0_1_muci] = expfit(class_0(:,1));
```

```
%%% fit the exponential class-conditional for input attribute 1 and class 1
%%%  p(x_1|y=1, \mu_1_1)
[exp_1_1_muhat, exp_1_1_muci] = expfit(class_1(:,1));

%%%% fitting of the class-conditional of the second attribute
%%%% with normal distribution
%%% class-condtional for class 0
%% p(x_2|y=0,mu_0_2,sigma_0_2)
[norm_0_2_mu,norm_0_2_sigma,muci_0_2,sci_0_2] = normfit(class_0(:,2));
%%% etc.
```

Part b. List parameters found by your program in the report.

Problem 2.3. Classification with the Naive Bayes model

Once the parameters of the Naive Bayes model are learned (estimated) , the decision about the class for a specific input $\mathbf{x}$ can be made by designing the appropriate discriminant functions. Typically, there are based on class posteriors, thus a classification problems boils down to the problem of comparison of posteriors of classes for $\mathbf{x}$. These are computed through the Bayes rule:

$$p(y=1|x) = \frac{\left[\prod_{i=1}^{d} p(x_i|y=1)\right] p(y=1)}{\left[\prod_{i=1}^{d} p(x_i|y=0)\right] p(y=0) + \left[\prod_{i=1}^{d} p(x_i|y=1)\right] p(y=1)}.$$

Note that in order to make the best posterior choice it is sufficient to compare the following discriminant functions based on log posteriors:

$$g_0(x) = \left[\sum_{i=1}^{d} \log p(x_i|y=0)\right] + \log p(y=0) \quad (1)$$

$$g_1(x) = \left[\sum_{i=1}^{d} \log p(x_i|y=1)\right] + \log p(y=1) \quad (2)$$

Part a. Write and submit a program *main2_3.m* that:

- Calls a function *predict_NB* that predicts class labels for inputs based on class posterior. The discriminant functions you need to use here are given in expressions 1 and 2 and use parameters obtained in Problem 2.2.
- Uses *predict_NB* to compute the misclassification error of the Naive Bayes classifier on both training and test datasets. Report the errors.

- Calculates and reports a *confusion matrix* for the test and training sets.

Part b. In your report include:

- Training and test misclassification errors.
- Confusion matrices for the train and test sets.
- Sensitivity and specificity of the model on the test set.

Part c. Compare and analyze the results for the Naive Bayes classifier with the results for the logistic regression model from Problem 1.