Solutions to Problem set 5

1 Logistic regression

(a) Code
(b) Code
(c) Code
(d) If we use $2/\sqrt{i}$ learning rate, after 2000 steps. Missclassification error of the train set is: 0.306122. Missclassification error of the test set is: 0.270742.

Training confusion matrix:

<table>
<thead>
<tr>
<th>Predict</th>
<th>Target0</th>
<th>Target1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>263(TN)</td>
<td>89(FN)</td>
</tr>
<tr>
<td>1</td>
<td>76(FP)</td>
<td>111(TP)</td>
</tr>
</tbody>
</table>

Testing confusion matrix:

<table>
<thead>
<tr>
<th>Predict</th>
<th>Target0</th>
<th>Target1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>121(TN)</td>
<td>22(FN)</td>
</tr>
<tr>
<td>1</td>
<td>40(FP)</td>
<td>46(TP)</td>
</tr>
</tbody>
</table>

Sensitivity = 0.6765. Specificity = 0.7516.
The learning curve is shown in Figure 1.

e Here you are required to generate a single layer neural network (perceptron) and using it to perform logistic regression—first running it with default settings for 2000 epochs (update steps), and then experimenting with other settings. For demonstrative purposes, we show two such experiments here. Figure 2 shows MMEs for a single layer perceptron using the logistic sigmoid function and gradient descent backpropagation training; figure 3 modifies this architecture by replacing gradient descent backprop with conjugate gradient. Both were run for 2000 epochs. Other things one can vary are initial set of weights, stopping criterion, type of sigmoid functions. Experiments performed here were with randomly chosen weights.
Figure 1: One initial weights, $\alpha = \frac{2}{\sqrt{i}}$, 2000 steps.

Figure 2: Single layer NN, logistic function, 2000 epochs, gradient descent.
Figure 3: Single layer NN, logistic function, 2000 epochs, conjugate gradient descent.

Error for single layer NN, with logistic function and gradient descent:
\[ MME_{\text{train}} = 0.2263 \]
\[ MME_{\text{test}} = 0.2140. \]

Error for single layer NN, with logistic function and conjugate gradient descent:
\[ MME_{\text{train}} = 0.2226 \]
\[ MME_{\text{test}} = 0.2096. \]

2 Naive Bayes model

This problem consisted of three parts. In the first, you were to run a histogram analysis on the Pima dataset and propose distributions appropriate for each of the features in that dataset as well as pick out and argue for two features which you found promising. Below are shown each histogram for features 1 ... 8, with the distribution appropriate for it. Note that judgements as to how to interpret these histograms were, for the purposes of this assignment, somewhat but not entirely subjective. In particular, features 3 and 4 both had significant outliers which may have influenced some to argue for bimodal distributions rather than unimodal and normal, which is how we treat them here.
2.1 Exploratory data analysis

In addition to the histogram analysis, the first part of the assignment required you to measure the maximum and minimum values, means, and standard deviations for all features. Between these metrics and the histograms themselves you were to choose and argue for 2 features which looked promising to you. Reasonable answers were those which pointed to features which showed different behaviors given the class. Features which behaved the same regardless of class should be candidates for discard, or should be weighted lower.

(a) Max, Min, Std, and Mean values for Pima’s eight features are given in the table below.

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>17.0000</td>
<td>199.0000</td>
<td>122.0000</td>
<td>99.0000</td>
<td>846.0000</td>
<td>67.1000</td>
<td>2.4200</td>
<td>81.0000</td>
</tr>
<tr>
<td>min</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0780</td>
</tr>
<tr>
<td>mean</td>
<td>3.8451</td>
<td>120.8945</td>
<td>69.1055</td>
<td>20.5365</td>
<td>79.7995</td>
<td>31.9926</td>
<td>0.4719</td>
<td>33.2409</td>
</tr>
<tr>
<td>std</td>
<td>3.3696</td>
<td>31.9726</td>
<td>19.3558</td>
<td>15.9522</td>
<td>115.2440</td>
<td>7.8842</td>
<td>0.3313</td>
<td>11.7602</td>
</tr>
</tbody>
</table>

(b) Histograms for each feature showing the most appropriate distribution to match it with:
Figure 6: Feature 3, normal distribution.

Figure 7: Feature 4, normal distribution.

Figure 8: Feature 5, exponential distribution.

Figure 9: Feature 6, normal distribution.
2.2 Learning of the Naive Bayes classifier

(a) Here you were required to implement and run a Naive Bayes generative model classifier on some pre-split Pima datasets, providing your classification errors and confusion matrix.

(b) The parameters of Naive Bayes learned were:

<table>
<thead>
<tr>
<th>Class</th>
<th>Prior</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Attr4</th>
<th>Attr5</th>
<th>Attr6</th>
<th>Attr7</th>
<th>Attr8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_0$</td>
<td>0.629</td>
<td>3.242</td>
<td>109.6</td>
<td>67.53</td>
<td>19.73</td>
<td>67.72</td>
<td>30.31</td>
<td>0.4164</td>
<td>31.10</td>
</tr>
<tr>
<td>$\mu_1$</td>
<td>0.371</td>
<td>4.710</td>
<td>141.4</td>
<td>70.19</td>
<td>22.94</td>
<td>103.7</td>
<td>35.26</td>
<td>0.5491</td>
<td>37.12</td>
</tr>
<tr>
<td>$\sigma_0$</td>
<td>-</td>
<td>-</td>
<td>26.23</td>
<td>18.67</td>
<td>14.58</td>
<td>-</td>
<td>7.73</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\sigma_1$</td>
<td>-</td>
<td>-</td>
<td>33.67</td>
<td>21.62</td>
<td>17.83</td>
<td>-</td>
<td>7.33</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

2.3 Classification with the Naive Bayes model

(a) Code

(b) Error Breakdown:

\[ NBErr_{train} = 0.2393 \]

Training confusion matrix:
\[
\begin{pmatrix}
289 & 79 \\
50 & 121
\end{pmatrix}
\]

\[NBErr_{test} = 0.2271\]

Test confusion matrix:

\[
\begin{pmatrix}
138 & 29 \\
23 & 39
\end{pmatrix}
\]

Sensitivity = 0.5735. Specificity = 0.8571.

(c) The Naive Bayes model performs better overall compared to the logistic regression model.

3 SVM

This problem required running the pre-provided programs for SVM, reporting the same sorts of results as shown for B. For this experiment we used a cost of 1. The results were:

\[SVMErr_{train} = 0.2319\]

Train confusion matrix:

\[
\begin{pmatrix}
299 & 40 \\
85 & 115
\end{pmatrix}
\]

\[SVMErr_{test} = 0.1965\]

\[
\begin{pmatrix}
142 & 19 \\
26 & 42
\end{pmatrix}
\]

Some of you experimented with the cost constant (the cost from crossing the margin) in your experiments, which produced somewhat different results. As it turns out, this is undesirable with the Pima dataset: The minimal error comes at a cost of 1, and essentially does not change at all about 10, as the following graph shows. Note that the x-axis is graphed in log base 10 scale.
Figure 12: SVM Errors shown at various costs ranging from 0.01 to 100.

4 ROC analysis

AUC can be computed by:

\[
AUC = \frac{1}{N-1} \sum_{x=0}^{N-1} \frac{1}{2} |X_{i+1} - X_i|(Y_{i+1} + Y_i)
\]

where, \(X_i = 1 - SP_i\) and \(Y_i = SN_i\).

(a) The ROC curve of the Logistic regression model is shown in Figure 13. AUC = 0.7380.

(b) A common mistake here was in calculating a probability for thresholds other than 0.5. For a threshold of 0.5, it is sufficient to compare \(g_1 > g_0\). If you look carefully at the way \(g_0\) and \(g_1\) are defined, however, they are just the logarithmic probability, missing a normalizing constant. Therefore, to convert these to a probability, use:

\[
p_1 = \frac{\exp(g_1)}{\exp(g_0) + \exp(g_1)}
\]

This happens to be equivalent to \(\frac{1}{1 + \exp(-(g_1 - g_0))}\), or the sigmoid function using the difference between \(g_1\) and \(g_0\) as input.

The ROC curve of the Naive Bayes model is shown in Figure 14. AUC = 0.8148.
Figure 13: Logistic regression model

Figure 14: Naive Bayes model

Figure 15: SVM
(c) AUC = 0.85

(d) Comparing the results we clearly see the SVM model leading to the best AUC on the test set, outperforming both the Naive Bayes and the logistic regression model. The second best is the Naive Bayes model.