

Solutions to Problem set 6

This assignment called for performing experiments and comparison of the single layer perceptron, various multi-layer neural network structures and the Support Vector machine on the pima dataset. The program `divideset.m` was provided for producing a random test/training split.

Problem 1 required generating 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 ?? shows MMEs for a single layer perceptron using the logistic sigmoid function and gradient descent backpropagation training; figure ?? 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.

Error for single layer NN, with logistic function and gradient descent:

$$MME_{train} = 0.2263$$

$$MME_{test} = 0.2140.$$

Error for single layer NN, with logistic function and conjugate gradient descent:

$$MME_{train} = 0.2226$$

$$MME_{test} = 0.2096.$$

Problem 2 called for the same procedure as part B, but incorporating a hidden layer. Here we show results for a single hidden layer of 2, 3, 5, and 10 units respectively. All experiments were performed with randomly initialized weights. What would happen if all the weights were set uniformly?

The first experiment (Figure ??) was on 2 hidden units trained for 2000 epochs using gradient descent. The errors obtained were:

$$MME_{train} = 0.31911$$

$MME_{test} = 0.30131$. Relatively large errors are caused by the fact that learning multilayer network involves more weights as compared to the single layer architectures. Training of multiple weights typically causes larger weight fluctuations and slower improvements (weights on the hidden layer and output layer need to be properly aligned). Also the threat here is the convergence to a local optima. To check this we repeated the above experiment for 5000 epochs (Figure ??). The errors obtained at the end were: $MME_{train} = 0.2486$

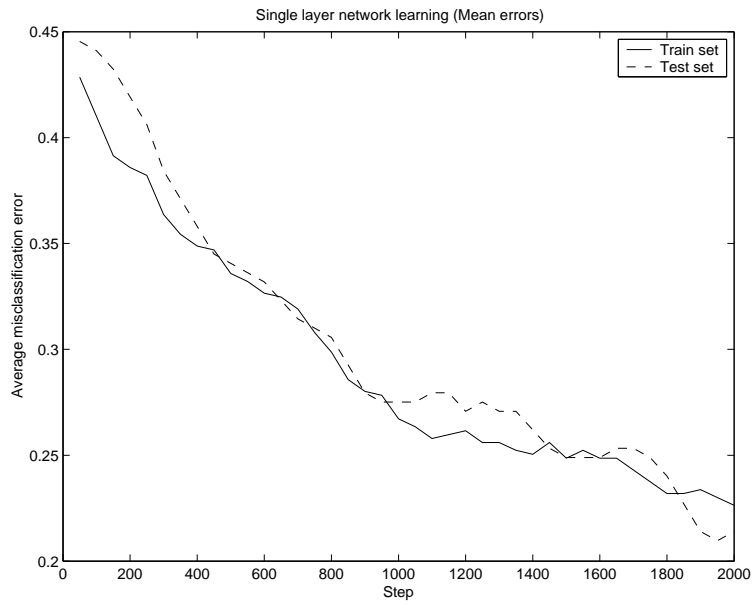


Figure 1: Single layer NN, logistic function, 2000 epochs, gradient descent.

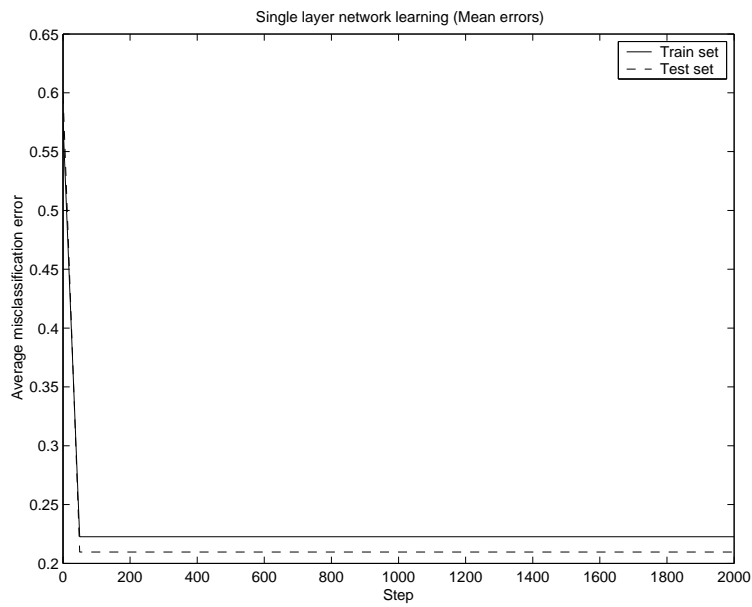


Figure 2: Single layer NN, logistic function, 2000 epochs, conjugate gradient descent.

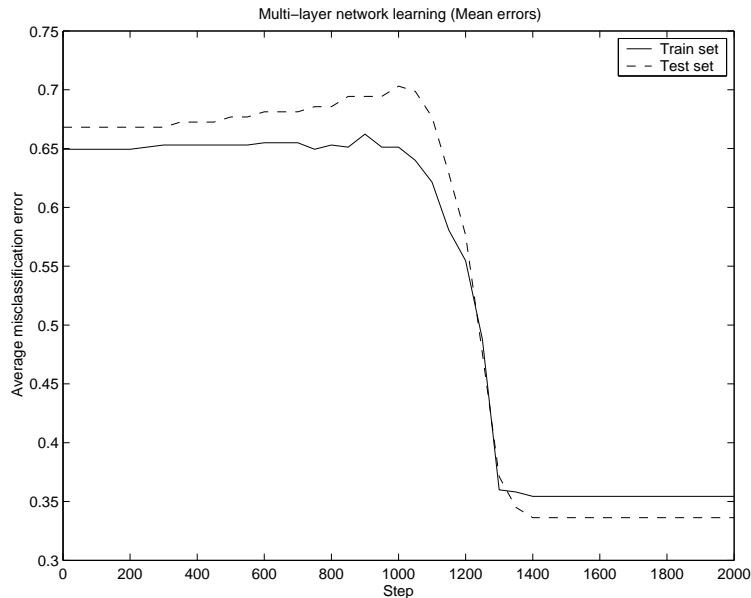


Figure 3: Hidden units = 2, 2000 epochs.

$MME_{test} = 0.2271$; showing significant improvements. Extending the experiment to 10000 epochs we were able to get: $MME_{train} = 0.2245$

$MME_{test} = 0.2096$. Note also that the learning method is very sensitive to the initial set of weights. Figure ?? illustrates the errors for 2000 epochs with different initial weights. In this case we observe quite different results. In general, learning of multilayer networks is: very sensitive to initial set of weights and also the method may converge to a local optima.

Figures ??,??,?? illustrate the performance of multilayer neural networks with 3, 5 and 10 units in the hidden layer. Corresponding errors at the end of 2000 epochs are as follows.

Three hidden units:

$$MME_{train} = 0.31911$$

$$MME_{test} = 0.30131$$

Five hidden units:

$$MME_{train} = 0.23562$$

$$MME_{test} = 0.29694$$

Ten hidden units:

$$MME_{train} = 0.2449 \quad MME_{test} = 0.30131$$

Problem 3 required running the pre-provided programs for SVM, reporting the same sorts

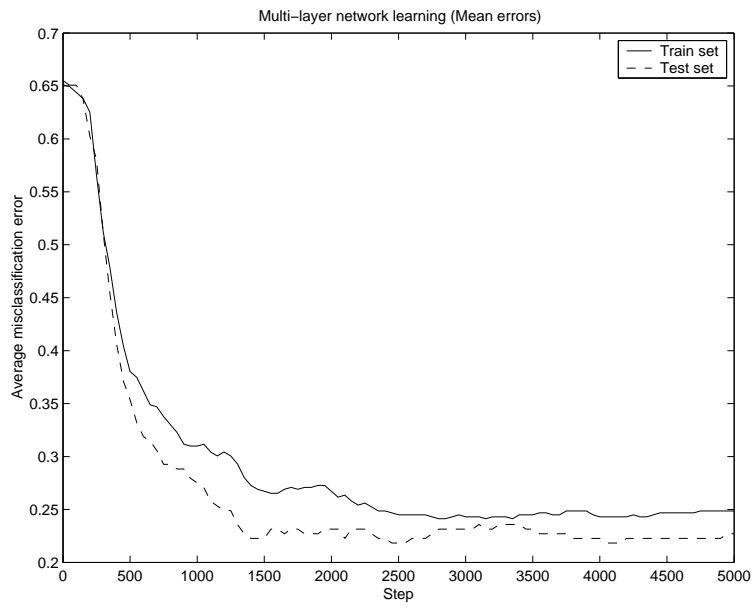


Figure 4: Hidden units = 2, 5000 epochs.

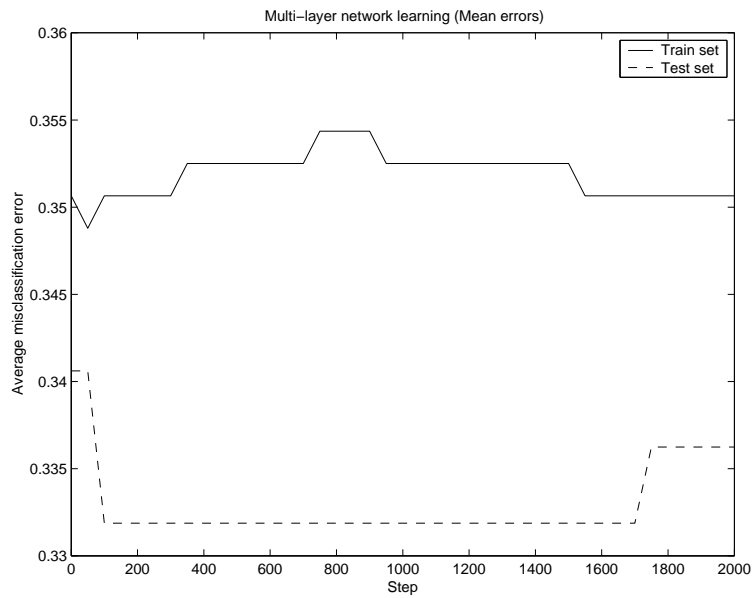


Figure 5: Hidden units = 2, 2000 epochs.

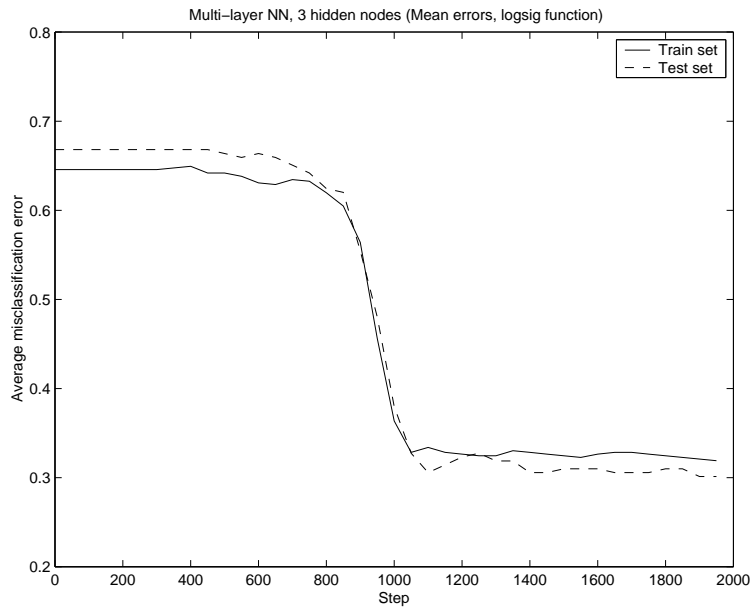


Figure 6: Hidden units = 3, 2000 epochs.

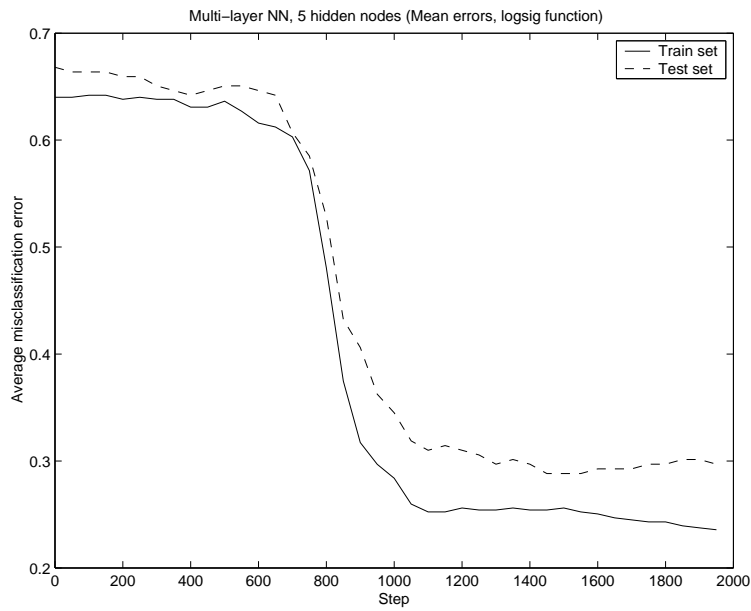


Figure 7: Hidden units = 5, 2000 epochs.

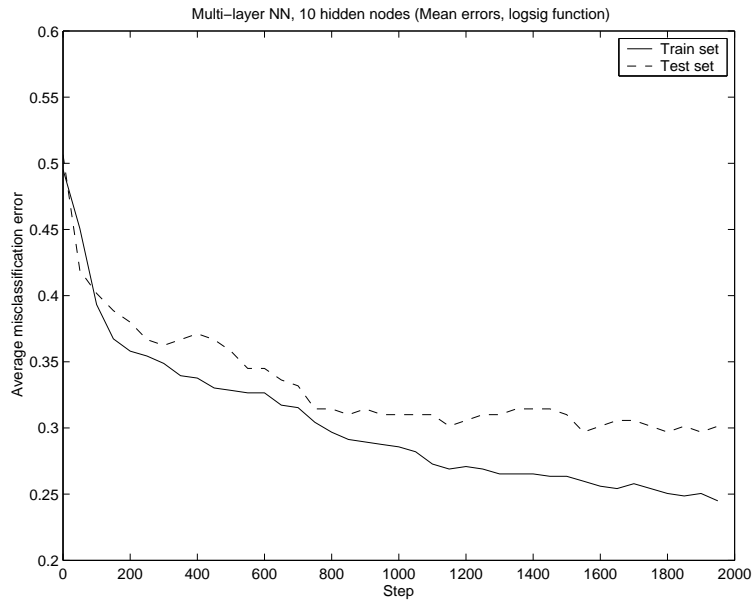


Figure 8: Hidden units = 10, 2000 epochs.

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.

Problem 4 The final experiment was something different: It called for performing 30 experiments, averaging their results, and using these to compare SVM and Naive Bayes

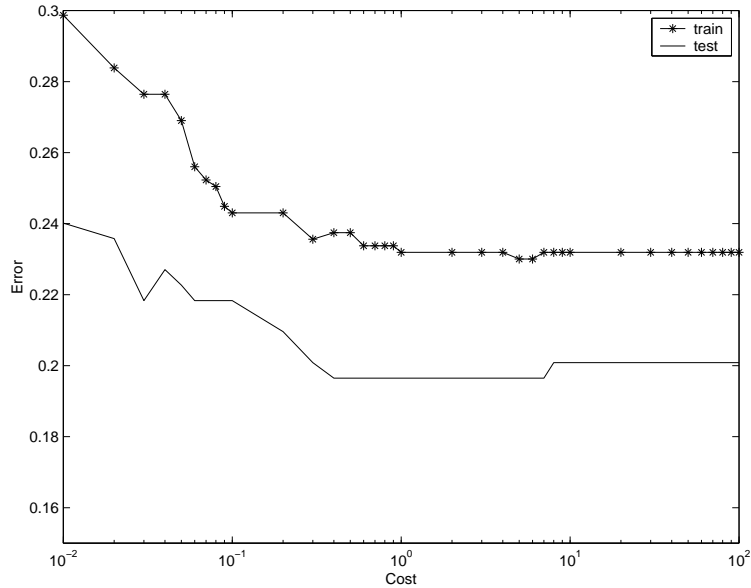


Figure 9: SVM Errors shown at various costs ranging from 0.01 to 100.

approaches. The point of the experiment was to overcome any differences brought about by using different splits, which could be quite significant.

A fairly typical picture of performance follows.

	Train		Test	
	Mean	Dev	Mean	Dev
Logistic Regression	0.3464	0.0190	0.3457	0.0338
Neural Network (2 hidden)	0.3392	0.0258	0.3405	0.0339
Support Vector Machine	0.2212	0.0103	0.2314	0.0236

Training set performance

LR	Predicted		MNN (2)	Predicted		SVM	Predicted	
Actual	0	1	Actual	0	1	Actual	0	1
0	0.6194	0.0288	0	0.5847	0.0635	0	0.5776	0.0706
1	0.3176	0.0343	1	0.2758	0.0761	1	0.1505	0.2013

Testing set performance

LR	Predicted		MNN (2)	Predicted		SVM	Predicted	
Actual	0	1	Actual	0	1	Actual	0	1
0	0.6265	0.0313	0	0.5895	0.0683	0	0.5776	0.0812
1	0.3144	0.0278	1	0.2722	0.0700	1	0.1502	0.1920

To evaluate the capability of the learner to generalize on the problem the average error is a better measure for this purpose than the single split error. The reason for this is that a single split can be simply “lucky” or “unlucky” in the sense that, for example, the split omits or overemphasizes some property of the data.

For this data set the support vector machine learner seems to be the best candidate. It has the best mean errors, while the test and train errors are close (no over fitting). The standard deviation of misclassification error indicates that it is most stable learner of the three.