CSE 517a: Homework #4

Due on Tuesday, April 21st, 2015

2:00pm

Contents

Programming

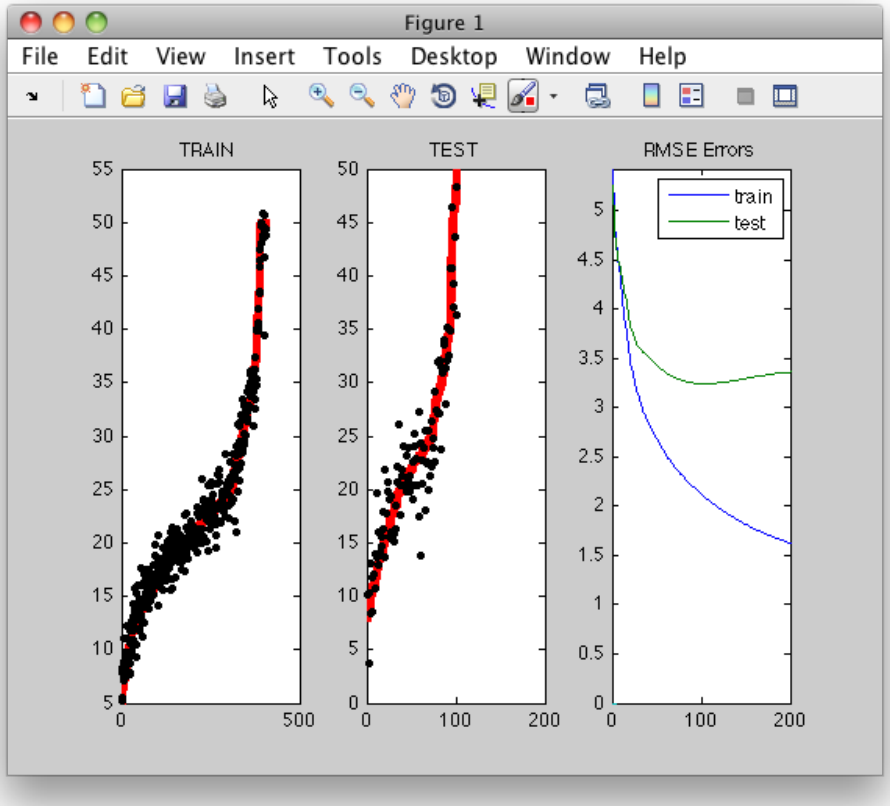

Figure 1: The results of the boston housing data set. *Left:* The predicted house-price (black) and the true price (red) of the training data. *Middle:* The predicted prices of the test data. *Right:* The root-mean-squared error (RMSE) of train (blue) and test (green) as a function of gradient steps.

In this assignment you will implement a Neural Network. First, perform an "svn update" in your svntop directory. Then Download the boston housing data set

`http://www.cse.wustl.edu/~kilian/cse517a2015/hw4/boston.mat`

This data set contains housing prices as targets and community statistics as features.

As always, make sure to add a test for every function you implement.

1. Implement the function *preprocess.m*. This should make the training data set zero-mean and each feature should have standard deviation 1. Make sure to apply exactly the same transformations to the test data set. (HINT: Each input vector should be transformed by $\vec{x}_i \rightarrow U\vec{x}_i - m$, where u can be a diagonal $d \times d$ matrix.)

2. (*optional*) Ideally you would like the input features to be de-correlated. The correlation matrix should be diagonal (in this case even the identity matrix). One way to do this is to project the data onto the PCA principal components (which we will still cover later in the course). You can get the transposed projection matrix by calling $pcacov(xTr')$. Make sure to apply PCA *after* you subtracted off the mean.

continued on next page...

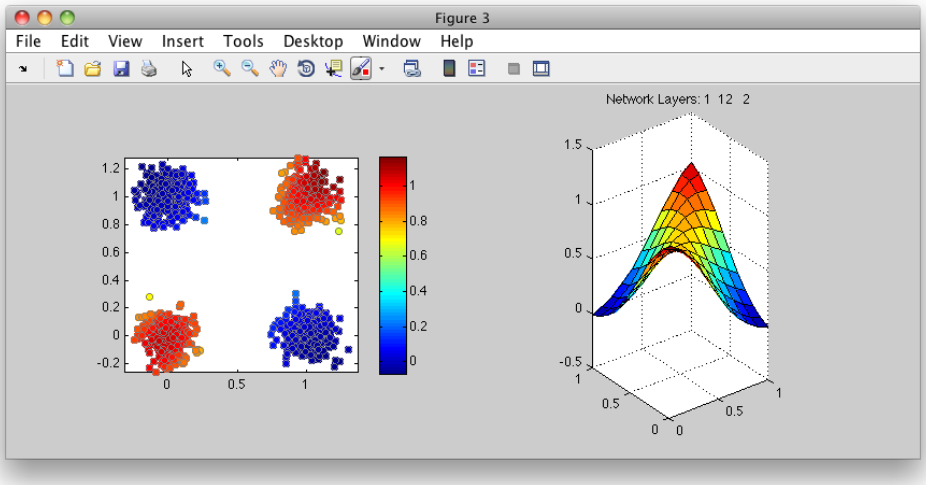

Figure 2: Results on a xor data set (with noise). *Left:* The prediction of the training data set. *Right:* The predictions of the surface within the unit square.

3. Implement the function *ffnn.m* which computes the loss and gradient of a particular feed-forward neural net for a data set xTr, yTr. All weights are passed in as a single vector w. The neural net structure is passed in as an array, where e.g. $wst = [1, 5, 12, 17]$ would mean a neural network with one output, and two hidden layers with 5 and 12 nodes and an input dimensionality of 17. With any data set x and y should be able to train the neural net with your *grdescent* function from the previous homework.

```
f=@(w) ffnn(w,x,y,wst);
w=grdescent(f,w,0.1,100,1e-8);
```

Alternatively, you can download

`http://learning.eng.cam.ac.uk/carl/code/minimize/minimize.m`

which is a slightly more sophisticated implementation of gradient descent (some people claim it is among the top-ten best pieces of code ever written) and call:

```
[w,l]=minimize(w,'ffnn',100,x,y,wst);
```

Use the script *testhw4.m* to test if your implementation is correct. Once you pass this test, you can evaluate your neural net and call the functions *xordemo* and *bostondemo*. How are the results affected when you change the network structure (or you switch to *minimize.m*)?