Problem Set 5
Due on Friday, November 11, 2016 at 11:55 pm

How to Submit
Create one zip file (.zip) of your code and submit it via `ilearn.ucr.edu`. This zip file should have a single function named `q1.m` that runs all of the examples. `q1.m` should have a single parameters, `numrep`, whose semantics are explained below. It should also contain the code necessary to run these samples. Finally, it should contain a single pdf file (q1.pdf) that has your plots.

Do not supply any directories in your zip file. Each file should (in comments) list

- Your name
- Your UCR student ID number
- The date
- The course (CS 229)
- The assignment number (PS 5)

Question 1 [10 points]
Summary:

- Implement a procedure to train a neural network.
- Train it on the "class2d" dataset, supplied in a file with $y \in \{0, 1\}$, because that is more natural for neural networks.
- Plot the results.

All non-linearities should be signoid functions (including the output). It should be trained with cross-entropy (the same loss function used in class for neural-network binary classification).

Select the step-size (η) as follows.

- Start η as 1.
- After each step, check the total error (including regularization term).
- If this error goes up, undo the step and multiply η by 0.5.
- If this error does not go up, multiply η by 1.05 and stop if the error did not improve by more than 10^{-8} times its previous value.

Pick the weights initially at random from the interval $[-1, 1]$. Be sure to have a "bias" term for each layer (whose out-going weights should still be regularized like all other weights).

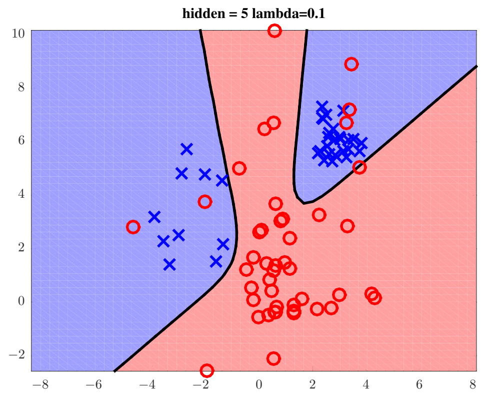
You should use batch gradient descent. Online or mini-batches may work better, but they are harder to debug and to grade, so for this assignment, please stick with batch gradient descent.

Additionally, you should run with "random restarts" (that is run to completion starting at different random weights), `numrep` times and select the weights that produce the smallest total loss.

Vary three aspects of your training:

- Number of layers: 2 or 3 (that is, 1 or 2 layers of hidden units)
- Number of hidden units per hidden layer: 1, 5, 20 (same in all hidden layers)
- Amount of weight-decay or regulariztion (λ in the slides from class): 0.001, 0.01, or 0.1

That totals 18 different experiments. For each experiment plot the resulting decision surfaces and the training data together. Set `numrep` to 5 for your plots. Please give the plot a title that makes is clear which experiment it is. Below is an example plot from a 2-layer, 5 hidden units per layer, result.



Please use `plotclassifer` to plot the decision surface. Just be certain to have the last two arguments be 0.5 and 0 to correspond to neural networks (and not SVMs).

You code will take a while to run to convergence. Collect all of your plots (see `help print` on how to output them as a pdf or similar), with titles (see `help title`) in a single pdf document. Submit this document and your code according to the instructions above.

Suggestions: Create separate functions to run forward propagation, backward propagation, and gradient computations (each of which depends on the previous). This will help in debugging. Plot the weights or the decision surface after each iteration (or set of iterations) while debugging. Construct your propagation functions and the like to handle an entire dataset at a time. For example, on forward propagation, do not loop over each data point, propagating it forward on its own. Instead, push all data points forward as a group to the next layer. This will greatly increase the speed of your program. It will take tens of thousands of iterations (at least) to converge on some of the examples.