Homework 2 -- due Thursday 24.12.2015, 11:59:59

Problem 1: Single Neuron for Classification (15p)

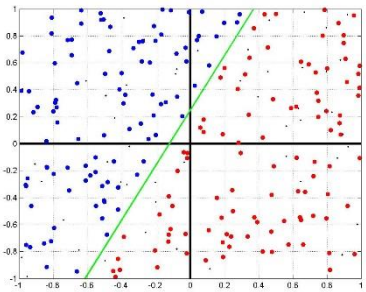
In this problem you shall implement a single neuron that learns to separate two classes. These two classes will be linearly separable.
Your algorithm reads a number of **3-valued training examples**: each such example consists of two inputs ("x" and "y" value) and a desired output value of `+1` or `-1`. The exact number of training examples is unknown, but you can safely assume you will read <= 1000.
At some point your program will find a training example '`0,0,0\n`' , (note the desired output of zero, which is invalid!). This indicates that the training data is completely read, and your program should start training the neuron.

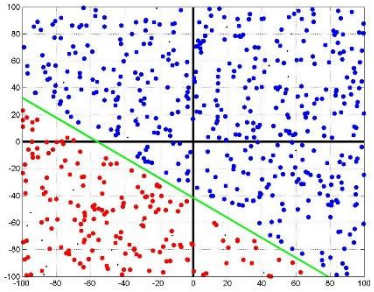
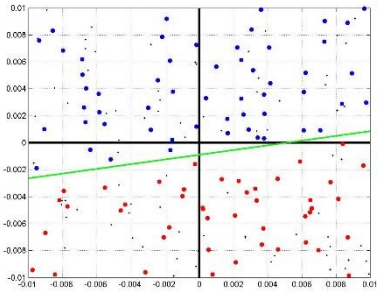
After training, your program continues to read **2-valued evaluation data**: for each such example your program should report the corresponding class (`+1` or `-1`) followed by return ('`\n`') as output.

Hint: A linear output neuron can do this task.
It is however much easier with a tanh neuron, which will classify into +1 and -1 classes with a sharp decission boundary.

Note (1): Do not include the example (0,0,0) in your training set!
Note (2): Your program needs to output the string '`+1\n`' (**with a plus sign**) for the positive class, not just a '`1\n`'.
Note (3): Think about how you compute the "error" of your neuron for training. What is the **neuron** supposed to output?

Hint: Remember the discussion in class about
- initialization of weights (use small random numbers)
- learning rate (suggestion: choose a small constant value)
- normalization of input and output data (assume that we do not query the network outside of the training domain)

Example Input	Visualization	Required Output
testInput10A.txt	 "click" for a large view **blue dots**: class +1	testOutput10A.txt

	red dots: class -1 **black dots**: query points **green line**: "unknown" decision boundary	
testInput10B.txt	 click" for a large view **blue dots**: class +1 **red dots**: class -1 **black dots**: query points **green line**: "unknown" decision boundary	testOutput10B.txt
testInput10C.txt	 click" for a large view **blue dots**: class +1 **red dots**: class -1 **black dots**: query points **green line**: "unknown" decision boundary	testOutput10C.txt

---

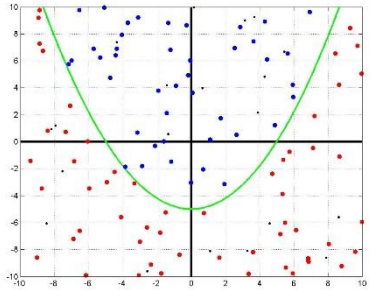
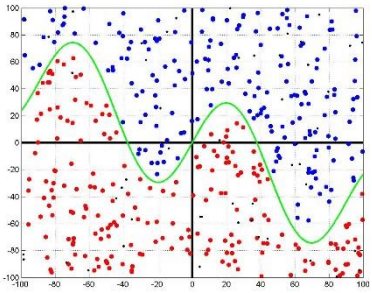
Problem 2: Neuronal Network for Classification (35p)

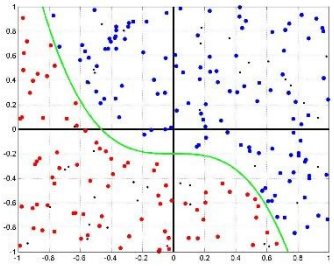
Same scenario as in problem 1, but here **the data is not linearly separable.**
You need to program a multi-layer neural network.
Recommendation: use "tanh" neurons in all your layers.
(same as in problem 1: a linear neuron in output layer can do, but it is easier with a tanh output).

Note (1): Do not include the example (0,0,0) in your training set!
Note (2): Your program needs to output the string '**`+1\n`**' (**with a plus sign**) for the positive class, not just a '**`1\n`**'.

Note (3): Think about how you compute the "error" of your neuron for training. What is the **neuron** supposed to output?

Hint: Remember the discussion in class about
- size of network (number of hidden layers and neurons per layer)
- initialization of weights (use small random numbers)
- learning rate (suggestion: choose a small constant value)
- normalization of input and output data (assume that we do not query the network outside of the training domain)

Example Input	Visualization	Required Output
testInput11A.txt	 click" for a large view **blue dots**: class +1 **red dots**: class -1 **black dots**: query points **green line**: "unknown" decision boundary	testOutput11A.txt
testInput11B.txt	 click" for a large view **blue dots**: class +1 **red dots**: class -1 **black dots**: query points **green line**: "unknown" decision boundary	testOutput11B.txt

testInput11C.txt	 click" for a large view **blue dots**: class +1 **red dots**: class -1 **black dots**: query points **green line**: "unknown" decision boundary	testOutput11C.txt

---

Problem 3: Single Neuron for Regression (15p)

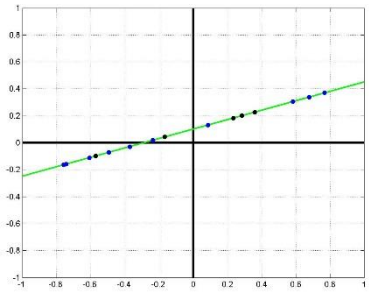
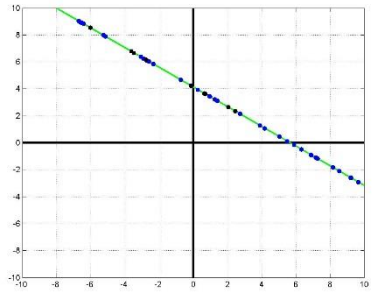
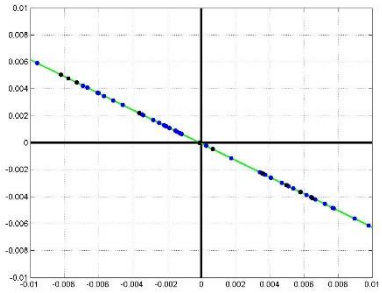
In this problem you shall implement a single neuron that learns to regress (approximate) a function. This function will be linear, so you can use a **single linear neuron**.
Your algorithm reads a number of **2-valued training examples**: each such example consists of one input ("x") and a desired output value ("y"). The exact number of training examples is unknown, but you can safely assume you will read <= 1000.
At some point your program will find a training example '`0,0\n`' , (note that this input might be a possible training point, but we define that (0,0) is invalid!). This indicates that the training data is completely read, and your program should start training the neuron.

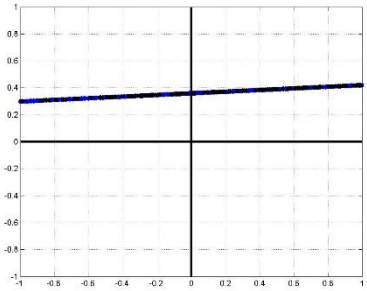
After training, your program continues to read **1-valued evaluation data**: for each such example your program should compute and print the neurons output followed by return ('`\n`').

Note: Do not include the data (0,0) in your training set!

Hint: Remember the discussion in class about
- initialization of weights (use small random numbers)
- learning rate (suggestion: choose a small constant value)
- normalization of input and output data (assume that we do not query the network outside of the training domain)

Example Input	Visualization	Required Output

testInput12A.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput12A.txt
testInput12B.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput12B.txt
testInput12C.txt	 "click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput12C.txt

testInput12D.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput12D.txt

---

Problem 4: Neuronal Network for Regression (35p)

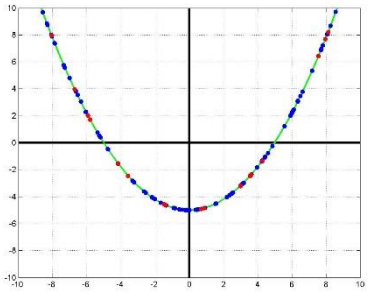
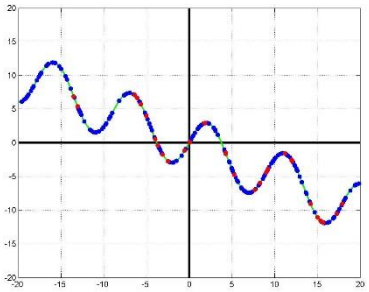
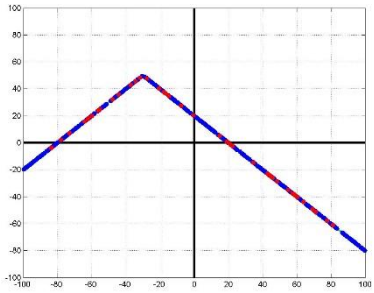
Same scenario as in problem 3, but here **the data is not a linear function.**
You need to program a multi-layer neural network.
Recommendation: use "tanh" neurons in your hidden layer(s) and a linear neuron in your output.

Hint: use a **multi-layer neural network (3-4 hidden layers), a few neurons per layer will do.**

Note (1): Do not include the data (0,0) in your training set!
Note (2): Your network might not be able to learn the data without a remaining error. This is normal. We will tolerate small deviations around the required output, typically within 5% of the overall output range.
Ensure that your network does not need "forever" to learn data. We will terminate your software after 30 seconds, which is plenty of time for learning!

Hint: Remember the discussion in class about
- size of network (number of hidden layers and neurons per layer)
- initialization of weights (use small random numbers)
- learning rate (suggestion: choose a small constant value)
- normalization of input and output data (assume that we do not query the network outside of the training domain)

Example Input	Visualization	Required Output

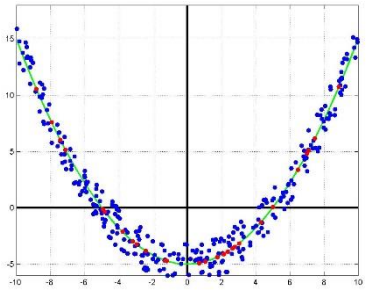
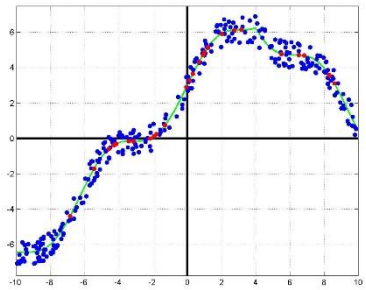
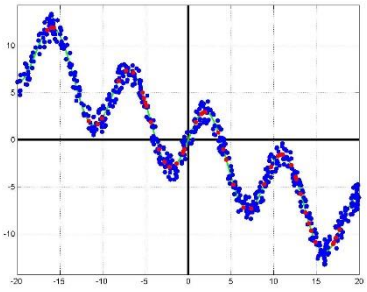
testInput13A.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput13A.txt
testInput13B.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput13B.txt
testInput13C.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput13C.txt

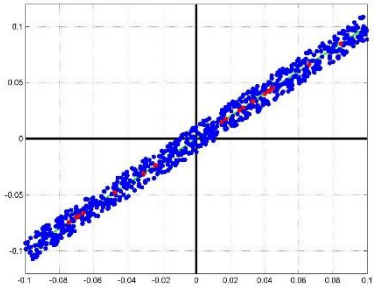
---

Problem 5: Neuronal Network for Regression of Noisy Data (BONUS) (20p)

Exact same scenario as in problem 4 (non-linear data), but here **the training data is noisy!**

You need to use techniques such as Simulated Annealing and/or Early Stopping (separation in training + test data) to solve the bonus assignment.

Example Input	Visualization	Required Output
testInput14A.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput14A.txt
testInput14B.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput14B.txt
testInput14C.txt	 "click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput14C.txt

testInput14D.txt	 click" for a large view **blue dots**: training examples **black dots**: query points **green line**: "unknown" underlying function	testOutput14D.txt