6.034 Quiz 3
November 14, 2007

Name	
EMail	

Circle your TA and recitation time, if any, so that we can enter your score in our records and return your quiz to you promptly.

TAs
Mike Klein
Tom Kollar
Akua Nti
Mark Seifter
Hope Lestier

Thu	
Time	Instructor
11-12	Kimberle Koile
12-1	Kimberle Koile
1-2	Kimberle Koile

Fri	
Time	Instructor
12-1	Howard Shrobe
1-2	Howard Shrobe
2-3	Howard Shrobe

Problem number	Maximum	Score	Grader
1	50		
2	50		
Total	100		

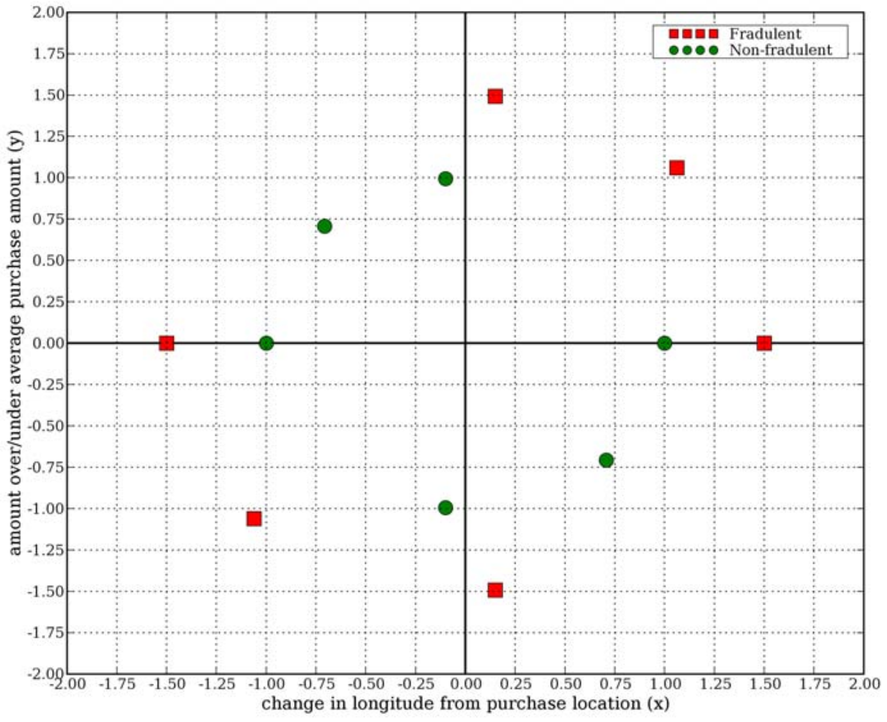
There are 10 pages in this quiz, including this one. After the 10th page, you will find a tear-off sheet with neural net formulas and a spare drawing.

Question 1: Nearest Neighbors and ID-Trees (50 pts)

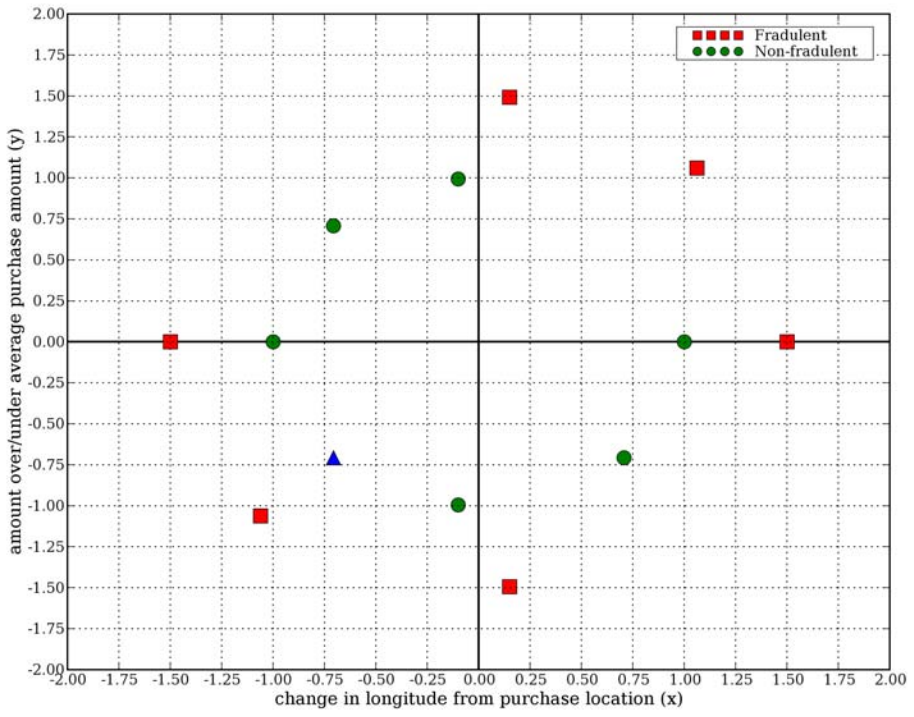
Lucy has been working hard for the credit card companies to detect fraud. They have asked her to analyze a number of classification methods to determine which one is best suited to their problem. The two quantities that they have provided her are the change in longitude from the purchase location to the registered address and the amount that the purchase is over or under the average purchase that the customer usually makes.

Part A: Nearest Neighbors (15 pts)

Lucy decides to use nearest neighbors to solve this problem and plots the fraudulent / non-fraudulent data. Squares are fraudulent and circles are non-fraudulent. Sketch the resulting decision boundary on the figure below.



It is the end of the month and Lucy's boss comes over with new data hot off the presses (the triangle). He wants Lucy to analyze whether or not the new charge is fraudulent.



What is the nearest neighbor classification of the new charge, fraudulent or non-fraudulent?

She's not too sure about this classification and decides to rerun it using k-nearest neighbors for *k*=3 and then for *k*=5. Is the charge fraudulent for these values of *k*?

K= 3:

K= 5:

Part B: Identification-trees (25 pts)

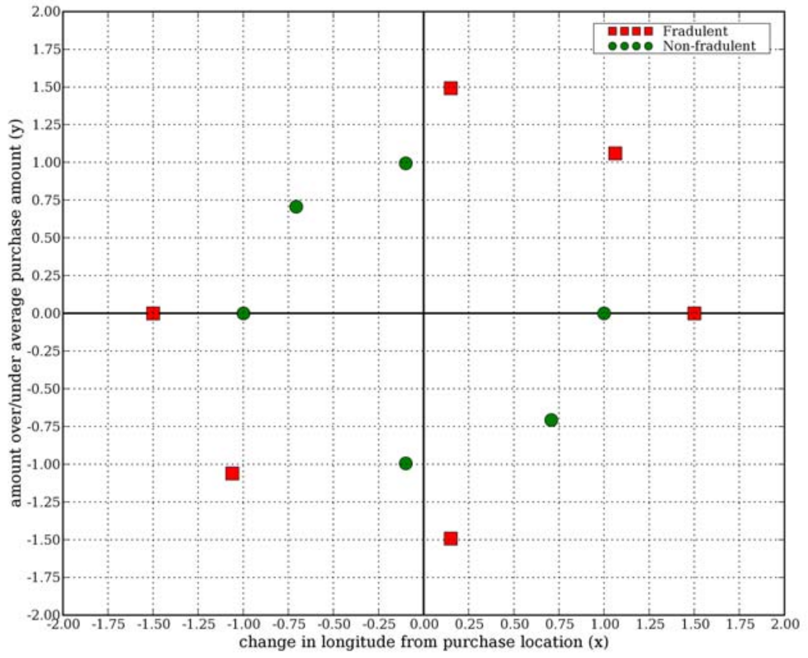
B1 The boundaries (18 pts)

Lucy now decides that she'll try to use identification trees on the data. There are three likely candidates for splitting the data: *x*=0.0, *x*=-1.01 and *x*=1.01. Note that the -1.01 and 1.01 values lie half-way between a square and a circle with nearby x values. Compute the average disorder for the decision boundary *x*=1.01. Your answer may contain logarithms.

Compute the average disorder for the decision boundary *x*=0.0. Again, your answer may contain logarithms.

Which of the two decision boundaries, *x*= 0.0 and *x*=1.01, is added first?

Sketch all of the decision boundaries on the figure below. Assume that *x*= 0.0 and *x*=1.01, in the order you determined above, are the first two decision boundaries selected (this may or may not be true, but assume it is).



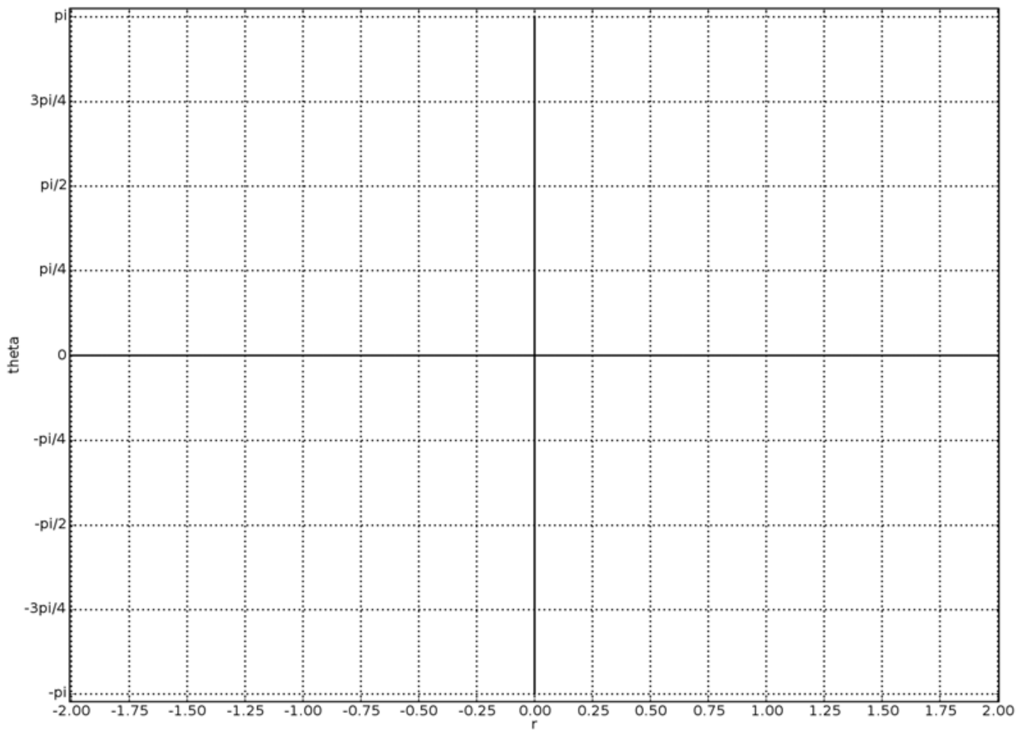
B2 The identification tree (7 pts)

Draw the identification tree corresponding to your decision boundaries.

What is the classification of the new charge (triangle)?

Part C: Polar coordinates (10 pts)

Lucy gets smart and decides to try a different space for each of the points. That is, she converts all of the points to polar coordinates. Sketch the data below. **You may assume that *r* value of each point is very close to a multiple of 0.25 and that the theta value of each point is very close to a multiple of pi/4.**



How many decision boundaries do we need in this case?

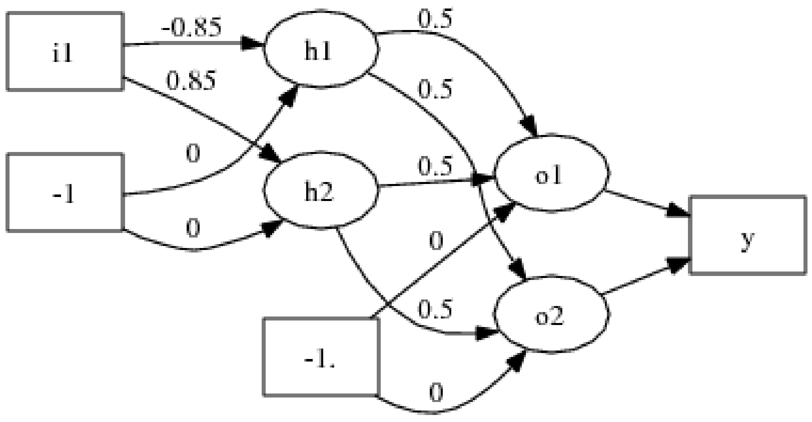
Draw the resulting identification tree and sketch the decision boundary on the graph above.

Question 2: Neural Networks (50 pts)

Part A: Forward Propagation and Backward Propagation (30 pts)

Lucy thinks about attacking the fraud problem with a neural net. She first reviews her knowledge of neural networks with sigmoid activation functions. She comes up with a simple example that has only one input i_1, two hidden nodes h_1 and h_2, and two output nodes o_1 and o_2. There are also two bias inputs labeled -1. The output vector y is $y=[o_1,o_2]$. All of the units are **sigmoid** units, meaning that their output will be given by the sum of the weighted inputs z such that:

$$s(z)=\frac{1}{1+e^{-z}}$$



Don't forget that the update for weights during the learning process is:

$$w_i'=w_i+r\delta_i x_i$$

For this problem assume that r=0.1. **See the tearout sheet if you need to be reminded of the particular updates for the sigmoid activation function.** Note the following table for values of the sigmoid function:

z	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
f(z)	0.60	0.63	0.65	0.66	0.67	0.68	0.69	0.7	0.71	0.72
z	-0.5	-0.55	-0.6	-0.65	-0.7	-0.75	-0.8	-0.85	-0.9	-0.95
f(z)	0.40	0.37	0.35	0.34	0.33	0.32	0.31	0.3	0.29	0.28

Assuming that i_1=1.0, then what are the output values, y_{o_1} and y_{o_2}, of the output nodes o_1 and o_2. In this and other parts of the problem, **you may express your answers in terms of variable-free arithmetic expressions, such as 0.9 × 0.9.**

y_{o_1} :

y_{o_2} :

Now assume that for input i_1=1.0, then the desired y_{o_1} should be 1.0 and the desired y_{o_2} should be 1.0. Compute delta for the output nodes o_1 and o_2, e.g. δ_{o_1} and δ_{o_2}.

. δ_{o_1} :

δ_{o_2} :

Using the deltas that you just computed, what is the delta (δ) for h_1, δ_{h_1}?

δ_{h_1}:

What is the new weight from h_1 to o_1?

New weight:

What is the new weight from i_1 to h_1?

New weight:

Part B: A new neuron activation function (20 pts)

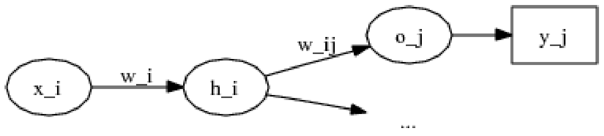
John has done a lot with neural networks and he thinks he can squeeze a little more performance out by replacing his old neuron activation function (the sigmoid) with a new neuron activation function (the hyperbolic tangent). Thus we have the following relationship:

$$y = g(z) = tanh(z)$$

The derivative of this hyperbolic tangent is:

$$g'(z) = 1 - tanh(z)^2$$

Assume that we're still using the normal quality metric: $P = -\frac{1}{2}(y^* - y)^2$. John, knowing he has to derive two updates for a two-layer network (abstractly shown below) starts by deriving the update for the outer layer w_{ij}. **See the tear-off page for more detail on backpropagation with a sigmoid activation function.**



Derive the new update equation for the weight changes and deltas in the outer layer. Ex**press your answer in a form analogous to the summary at the bottom of the tear off sheet.**

.

Now derive the update equation for the inner weight changes and deltas. **Express your answer in a form analogous to the summary at the bottom of the tear off sheet.**

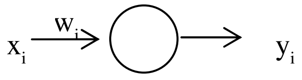
.

Backpropagation Notes

New weights depend on a learning rate and the derivative of a performance function with respect to weights:

(1) $w_i' = w_i + r\frac{\partial P}{\partial w_i}$

Using the chain rule, where y_i designates a neuron's output:



(2) $\frac{\partial P}{\partial w_i} = \frac{\partial P}{\partial y_i}\frac{\partial y_i}{\partial w_i}$

For the standard performance function, where *y** is the final desired output and *y* is the actual final output, $P = -\frac{1}{2}\sum(y^* - y)^2$:

(3) $\frac{\partial P}{\partial y_i} = \frac{\partial}{\partial y_i}(-\frac{1}{2}(y^* - y)^2) = (y^* - y)$

For a neural net, the total input to a neuron is $z = \sum w_i x_i$

(Note that x_i is sometimes written y_i to indicate that in a multilayer network, the input for one node is the output for a previous layer node)

For a sigmoid neural net, the output of a neuron, where z is the total input, is $y = s(z) = \frac{1}{1+e^{-z}}$.

Recall that the derivative $\frac{\partial y}{\partial z} = y(1-y)$.

So for the output layer of a sigmoid neural net:

(4) $\frac{\partial y_i}{\partial w_i} = \frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i} = y(1-y)x_i$

Substituting (3) and (4) into (2):

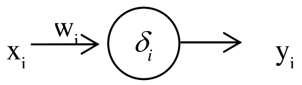
(5) $\frac{\partial P}{\partial w_i} = \frac{\partial P}{\partial y_i}\frac{\partial y_i}{\partial w_i} = (y^* - y)y(1-y)x_i$

Substituting (5) into the weight equation (1):

$w_i' = w_i + r(y^* - y)y(1-y)x_i$

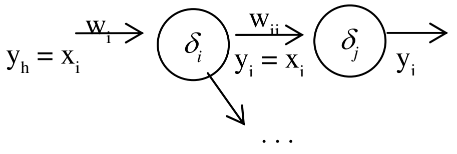
... which can be written in terms of δ, which is the derivative of *P* with respect to total input, $\frac{\partial P}{\partial z}$:

$w_i' = w_i + r\delta x_i$, where $\delta_i = \frac{\partial P}{\partial y}\frac{\partial y}{\partial z} = (y^* - y)y(1-y) = y(1-y)(y^* - y)$



(Note that to change *P*, one only needs to substitute the new $\frac{\partial P}{\partial y}$ into this equation.)

For an inner node, we use a similar equation that takes into account the output y_i for that node, and the δ and *w* values for the next layer, where "next layer" means the neighboring nodes one layer closer to the output layer:

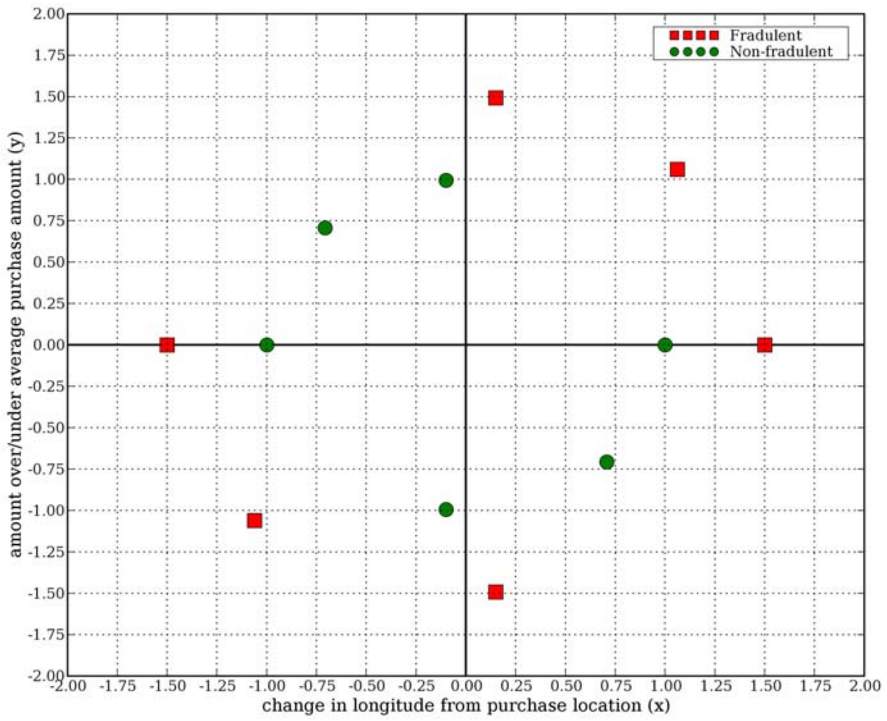


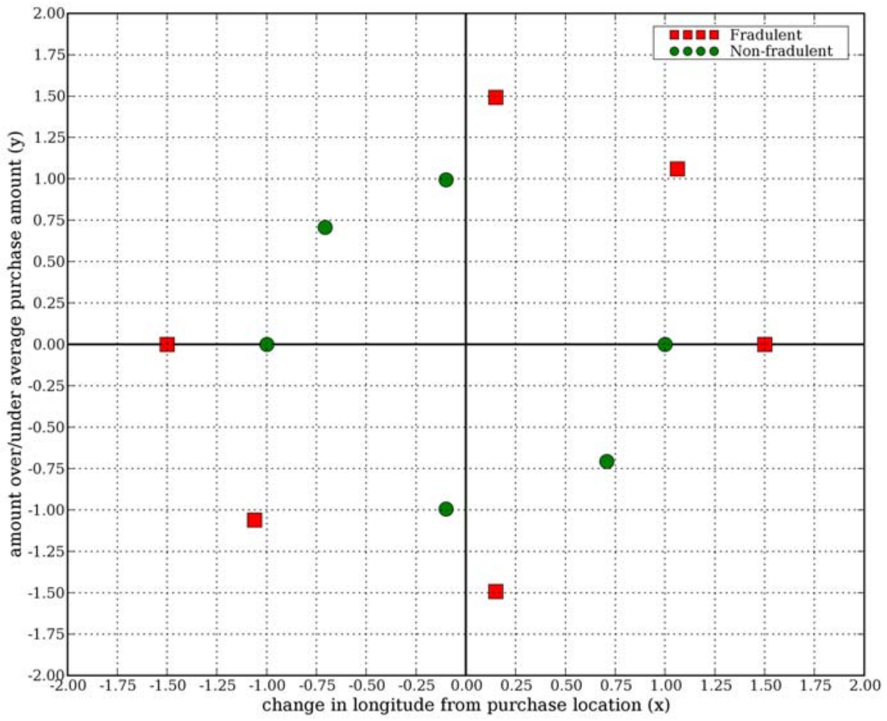
$w_i' = w_i + r\delta x_i$, where $\delta_i = y_i(1-y_i)\sum \delta_j w_{ij}$

and w_{ij} is the weight between layers *i* and *j*.

Summary: For each layer, $w_i' = w_i + r\delta x_i$, where for outer layer $\delta_i = y(1-y)(y^* - y)$

inner layer $\delta_i = y_i(1-y_i)\sum \delta_j w_{ij}$


Fradulent
Non-fradulent
amount over/under average purchase amount (y)
change in longitude from purchase location (x)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
-1.25
-1.50
-1.75
-2.00


Fradulent
Non-fradulent
amount over/under average purchase amount (y)
change in longitude from purchase location (x)