Practice Final Exam Solutions

19 December 2012

You have up to 2 hours, 30 minutes. You may not use text book or notes. You may refer to an online spreadsheet in those problems that provide a URL.

1. For each statement below, fill in the blank with the *best* term from the following list. Some terms might be used more than once; some might not be used at all.

   • crossover • entropy • Euclidean distance • Manhattan distance • mutation
   • sigmoid function • step function • supervised learning • tournament selection
   • unsupervised learning

   (a) Euclidean distance is a method for calculating distances between points, based on the square root of the sum of the squared differences along each dimension.

   (b) The step function governs the activation of a (simulated) neuron, where the value jumps from zero to one at some threshold.

   (c) tournament selection is a way to choose individuals from a population that will probabilistically select those with higher fitness.

   (d) In unsupervised learning, the learning algorithm is not provided with a correct classification for the data. Instead, it tries to cluster and find patterns.

2. In a genetic algorithm with bit representations, which of the following strings **cannot** be the result of a *crossover* between X and Y?

   X = 1 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0
   Y = 0 1 1 1 0 1 0 1 1 1 0 0 0 1 0 0

   (a) 1 1 0 1 0 1 0 0 1 1 1 0 0 1 0 0

   (b) 0 1 1 1 0 1 0 0 1 1 1 0 0 0 1 0 This one, because of that "1" in the next-to-last position.

   (c) 1 1 0 1 0 1 0 1 1 1 0 0 0 1 0 0

   (d) 0 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0

3. Which machine learning techniques are most suitable for classifying continuous, real-valued data, such as measurements?

   (a) Decision Tree Induction (ID3) — No, ID3 on its own works only with discrete data. There are various extensions to handle continuous data, by breaking it into ranges, but we didn't cover those.

   (b) Back-Propagation Neural Network — Yes, BPNN is great for classifying continuous data.

   (c) Genetic Algorithm — No, GA is mostly for optimizing parameters, not for classifying. It's also a bit weak for continuous data.

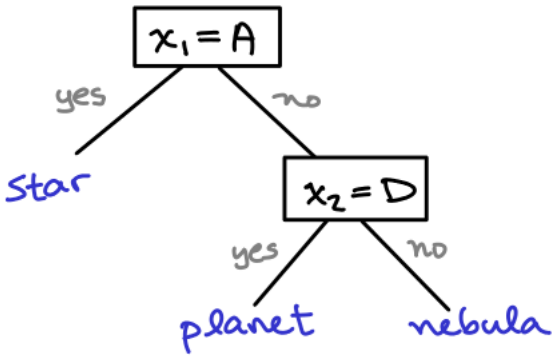
   (d) k-Nearest Neighbors — Yes, kNN is great for classifying continuous data, based on computing distances.

4. Suppose you were hired by the *Space Telescope Science Institute* in Baltimore. This is the organization that collects scientific data from the Hubble Space Telescope. They want you to create an intelligent program that can examine pictures of space objects and classify them as *planets, stars,* or *nebulae.* They already have subroutines that can extract four features (named x_1–x_4) from the images. They provide you with these sample data:

ID	x_1	x_2	x_3	x_4	classification
1	A	D	H	L	star
2	A	E	H	K	star
3	A	F	G	L	planet
4	A	F	H	K	star
5	A	F	H	L	star
6	B	D	G	K	planet
7	B	D	H	K	nebula
8	B	E	G	K	planet
9	C	D	G	L	planet
10	C	E	G	L	planet
11	C	E	H	L	nebula

You can view these examples in a Google Spreadsheet at `http://bit.ly/162galaxy` — I suggest trying View » List on the menu.

(a) Below is a colleague's attempt to design a decision tree for these data. Which records does it misclassify?



Misclassifies #3 as a star. Misclassifies #7 as a planet. Misclassifies #8, #10 as nebula.

(b) Use the telescope data and my entropy calculator at `http://bit.ly/162entropy` to determine the *best* initial test, and then produce a decision tree that will correctly classify all 11 records.

Consider all the tests, and the number that end up in each class to compute entropy, then choose the minimum one. Notation is number of stars/ planets/ nebula.

```
x1=A: 4/1/0 vs 0/4/2 = [5(.722) + 6(.918)]/11 = .829
x1=B: 0/2/1 vs 4/3/1 = [3(.918) + 8(1.41)]/11 = 1.28
x1=C: 0/2/1 vs 4/3/1 = same, 1.28
x2=D: 1/2/1 vs 3/3/1 = [4(1.5) + 7(1.45)]/11 = 1.47
x2=E: 1/2/1 vs 3/3/1 = same, 1.47
x2=F: 2/1/0 vs 2/4/2 = [3(.918) + 8(1.5)]/11 = 1.34
x3=G: 0/5/0 vs 4/0/2 = [5(0) + 6(.918)]/11 = 0.501  ===BEST
x4=K: 2/2/1 vs 2/3/1 = [5(1.52) + 6(1.46)]/11 = 1.49
```

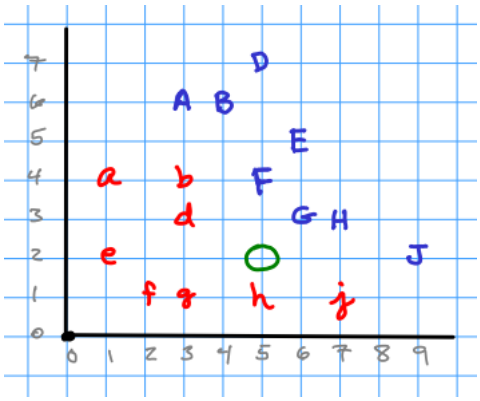
So, `x3=G` is obviously the best test, so we start there. To the left, if the answer is yes, then the class is planet.

Otherwise, when the answer to the first test is no, we could repeat this whole process on the 6 remaining records. But if you look at them, and maybe try sorting by class, you'll see right away that the test `x1=A` now perfectly divides them, into stars on the left and nebula on the right.

(c) Using the decision tree you developed in the previous part, classify these two additional records:

ID	x_1	x_2	x_3	x_4	classification
12	A	D	G	L	planet
13	B	F	H	L	nebula

5. Below is a Cartesian coordinate system containing 8 points classified as blue (upper-case letters), and 8 points classified as red (lower-case).



Your task is to classify the green circle at (5, 2) using a K *nearest-neighbors* algorithm. To keep the numbers simple, use Manhattan (city-block) distance.

(a) What classification (red/blue) will be chosen when K = 1 (meaning the algorithm considers just the single nearest neighbor).

The single closest neighbor is `h`, which is the only point at distance 1. It is red, so we classify the circle as red.

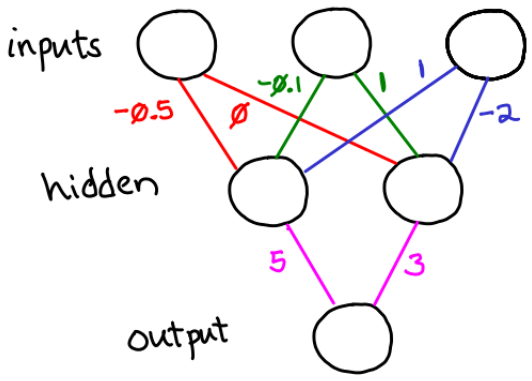
(b) What classification will be chosen when K = 3 (meaning the three nearest neighbors *vote* on the classification).

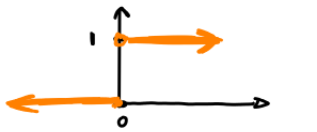
Let's compute distances of ALL the points, and then put them in order.

```
1  h  <- 1 neighbor chooses red/lowercase
2  F
2  G  <- 3 neighbors have 2 blue, 1 red so: blue/upper
3  H
3  d
3  g
3  j  <- 7 neighbors have 3 blue, 4 red, so: red/lower
4  E
4  J
4  b
4  e
4  f
5  B
5  D
6  A
6  a
```

(c) What classification will be chosen when K = 7 ?

6. Below is a multi-layer perceptron with three input neurons, two hidden neurons, and one output. Each neuron takes the sum of its weighted inputs, and applies a step function (as shown) to determine its own activation.





activation function

f(x) = if x > 0 then 1 else 0

(a) What is the output activation when the three inputs (left to right) are `0 1 0` ?

```
H0 = f(0 * -0.5 + 1 * -0.1 + 0 * 1)
   = f(0 + -0.1 + 0)
   = f(-0.1)
   = 0

H1 = f(0 * 0 + 1 * 1 + 0 * -2)
   = f(0 + 1 + 0)
   = f(1)
   = 1

Out = f(H0 * 5 + H1 * 3)
    = f(0 * 5 + 1 * 3)
    = f(3)
    = 1
```

(b) What is the output activation when the three inputs (left to right) are `1 0 1`?

```
H0 = f(1 * -0.5 + 0 * -0.1 + 1 * 1)
   = f(-0.5 + 0 + 1)
   = f(0.5)
   = 1

H1 = f(1 * 0 + 0 * 1 + 1 * -2)
   = f(0 + 0 + -2)
   = f(-2)
   = 0

Out = f(H0 * 5 + H1 * 3)
    = f(1 * 5 + 0 * 3)
    = f(5)
    = 1
```

7. In class and on assignment 6, we used a genetic algorithm to solve a *knapsack problem,* where we decide which objects to pack into a finite container to maximize the value. That was a 0-1 knapsack, where you either take an object or not. A generalization of this problem considers taking some discrete number of objects of a particular type.

item	count	value($)	weight(kg)
A	6	2	3
B	2	3	5
C	3	4	6

In this example, we can pack **up to 6** instances of object A (worth $2 each, and weighing 3kg each), and so on for B, C. However, our container has a weight limit of 25kg.

Use 7 bits to represent solutions to this problem: 3 bits for the number of 'A' objects, 2 bits for 'B', and 2 bits for 'C'. We'll define fitness as the monetary value, or zero if the selection is over the weight limit, or uses more than the number of available objects in each category.

What is the fitness of each bit string below?

(a) `(0 0 1) (0 1) (1 1)` with 1 A, 1 B, 3 Cs, the weight is 3 + 5 + 3*6 = 26, which is over weight. Fitness = 0.

(b) `(0 1 0) (0 0) (1 0)` with 2 As, 0 Bs, 2 Cs, the weight is 2*3 + 0 + 2*6 = 18, which is fine. Fitness = 2*$2 + 0 + 2*$4 = $12.

(c) `(0 1 1) (1 1) (0 1)` with 3 As, 3Bs, 1 C, we have an error because there are not 3 Bs available. So Fitness=0.

(d) `(1 0 0) (0 1) (0 1)` with 4 As, 1 B, 1 C, the weight is 4*3 + 5 + 6 = 23, which is fine. Fitness = 4*$2 + $3 + $4 = $15. So this is the best fitness found so far.