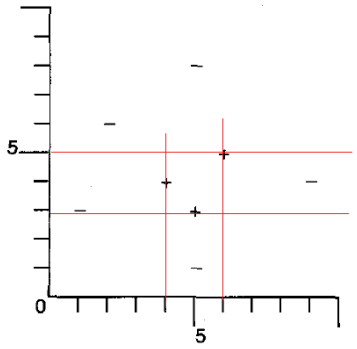
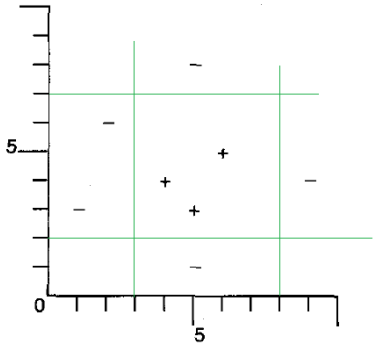
Ahmed Khademzadeh
Machine Learning – Spring 2012
Written Assignment #2

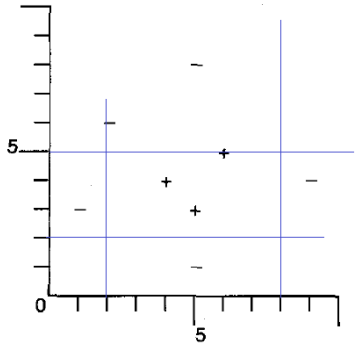
Problem 2.4

a. 4 <= x <= 6, 3 <= y <= 5



b. 3 <= x <= 8, 2 <= y <= 7 it also can be 2 <= x <= 8, 2 <= y <= 5. We can say that the first one is more general in some sense that its area is bigger than the second one. But since none of these include the other one completely we can say that both of them are general boundary in some sense.





c.

a. (7,6) will reduce the version space size either if it is a negative or positive instance. It will make the most specific boundary bigger if it is a positive and will make the general boundary smaller if it is a negative instance.
b. (5,5) will not change the hypothesis space if it is a positive instance. (2,8) also won't change the hypothesis space if it is a negative instance.

d. We can do that with 4 instances. Two positives ((3,2) and (5,9)) for maximizing the specific boundary and two negatives ((2,1) and (6,10)) for minimizing the general boundary.

Problem 2.7

Suppose we have n instances x_1 … x_n, and x_j is the biggest and x_i is the smallest among them. Our hypothesis based on the problem description should be something like to be the most specific one:

a < x < b such that a is the biggest REAL number smaller than x_i, and b is smallest number bigger than x_j. We can't find those REAL numbers, because whatever number we consider for a and b, we will be able to find smaller or bigger than them that satisfies the condition.

Two suggestions for solving the problem:

- Changing the hypothesis form to x_i <= x <= x_j. Now the most specific one will be one with x_i and x_j as the boundaries.
- We can restrict the a and b in (a < x < b) to those that are with some specific number of decimal points (for example 3). This way we will be able to find a, and b.

Problem 2.9

Declare: int[n][2] A; for each attribute we save two values. A[i][1] keeps number of instances having attribute i as true and A[i][2] keeps number of instances having attribute i as false.

```
For each instance I
  For each attribute j of I
    If (j == true) A[j][1]++
    else A[j][2]++;
H = ""
For i = 1 to n
  If (A[i][1] == 0) and (A[i][2] != 0) H += "a_i == false or"
  If (A[i][1] != 0) and (A[i][2] == 0) H += "a_i == true or"

Print H;
```

Problem 3.4

a)

```
AirTemp
   |---Warm
   |     Yes
   |---Cold
         No
```

b) It is one of the hypotheses that we have in version space shown in figure 2.3, and is equivalent to the second hypothesis in the general boundary of the version space.

Sunny Warm Normal Strong Warm Same Yes
Sunny Warm High Strong Warm Same Yes
Rainy Cold High Strong Warm Change No
Sunny Warm High Strong Cool Change Yes
Sunny Warm Normal Weak Warm Same No

- Entropy: 0.9709506
- Gain values in level one of the tree:
  - Sky:0.32
  - AirTemp:0.32
  - Humidity:0.019
  - Wind:0.32
  - Water:0.17
  - Forecast:0.019

Selected Attribute can be each of Sky, AirTemp, and Wind. We select Wind with gain:0.32

If Wind is Weak, then we only have one item of class No. Thus we stop at this leaf. If Wind is Strong, we have two choices (Yes for 3 and No for 1 train item).

- Entropy: 0.8112781
- Gain values in level two of the tree:
  - Sky:0.81
  - AirTemp:0.81
  - Humidity:0.12
  - Water:0.12
  - Forecast:0.31

We selected AirTemp with gain:0.81 and the tree will be as following:

```
Wind
   |---Strong
   |      AirTemp
   |          |---Warm
   |          |      Yes
   |          |---Cold
   |                 No
   |---Weak
          No
```

(d)

Sunny Warm Normal Strong Warm Same Yes

Problem e

i. We have three instances, any number of them can be Yes. Thus we have $2^3 = 8$ different possible hypotheses.
   - Our instances:
     - x=(sunny, high), y=(rainy, high), and z=(cloudy, high)
   - Our possible hypotheses:
     - h0: {no, no, no} ~ No yes ~ { }
     - h1: {no, no, yes} ~ {z=yes}
     - h2: {no, yes, no} ~ {y=yes}
     - h3: {no, yes, yes} ~{y=yes, z=yes}
     - h4: {yes, no, no}
     - h5: {yes, no, yes}
     - h6: {yes, yes, no}
     - h7: { yes, yes, yes}

ii. There are 8 different semantically different hypothesis:
   - h0 = x'y'z' ~ y'z'x' ~ z'y'x' ~ x'z'y' ~ y'x'z' ~ z'x'y' (6 syntactically different but semantically equal hypotheses).
   - h1 = x'y'z
   - h2 = x'yz'
   - h3 = x'yz
   - h4 = xy'z'
   - h5 = xy'z
   - h6 = xyz'
   - h7 = xyz

iii. H2 = {i1 = (?, ?, ?), i2 = (0,?, ?) ~ (?, 0, 0) ~ (0, ?, ?) ~ … ~ (0, 0, 0)} . Thus we only two different hypotheses. One "(?, ?, ?) ~ h7" includes all the instances, and the other "(0, 0, 0) ~ h0" includes none of them.

iv. Based on the section iii, H2 doesn't include h1 through h6 of H1.

Programming Assignment Discussion

Iris dataset
Corruption percent: 0%

validation	Tree accuracy on training	Tree accuracy on test	Pruned ruleset on training	Pruned ruleset on test
0%	100%	92%	N/A	N/A
10%	100%	94%	96.6%	96%
30%	100%	94%	95.7%	96%

Tree accyracy on training data is always 100% which is what we expect, because it is created based on training data and we also hadn't any depth restriction or any other kind of restriction on hypotheses space.

Tree accuracy on test is not 100% which is what we expect because we might have not seen similar to some instances in train data.

Accuracy of pruned ruleset on train data is not 100% because we have changed what we have learned from training data and thus it is not 100%.

Accuracy of pruned ruleset on test data is higher that accuracy of tree on test data which is base on the notion that we have removed the overfitting problem from learned rules.

Iris dataset
Validation percent: 30%

Corruption	Tree accuracy on training	Tree accuracy on test	Pruned ruleset on training	Pruned ruleset on test
0%	100%	94%	95.7%	96%
5%	100%	94%	91.4%	96%
10%	100%	80%	85.7%	96%
15%	100%	72%	81.4%	96%
20%	100%	70%	74.2%	96%
25%	100%	70%	68.6%	90%
30%	100%	62%	62.8%	90%
35%	100%	54%	62.8%	96%

Tree accuracy is 100% alwas on training data for the same reason stated above.

Tree accuracy on test data is decreasing with the increase of corruption, because of the more bad data we have in our training data that causes learning wrong concept or overfitting which finally causes to have decrease in accuracy of learning.

Accuracy of pruned ruleset on training data decreases with the increase of corruption. It is because pruning removes the overfitting negative effect. Thus the rules are not very fit to the noises present in training data and we have a decrease in accuracy.

Accuracy of pruned ruleset on test data don't decrease with the increase of corruption. The reason as stated above is that the pruning causes removing the overfitting problem which is the effect of having corrupted data. Thus removing the overfitting (effect of corrupted data) our learner will perform good on our test data.