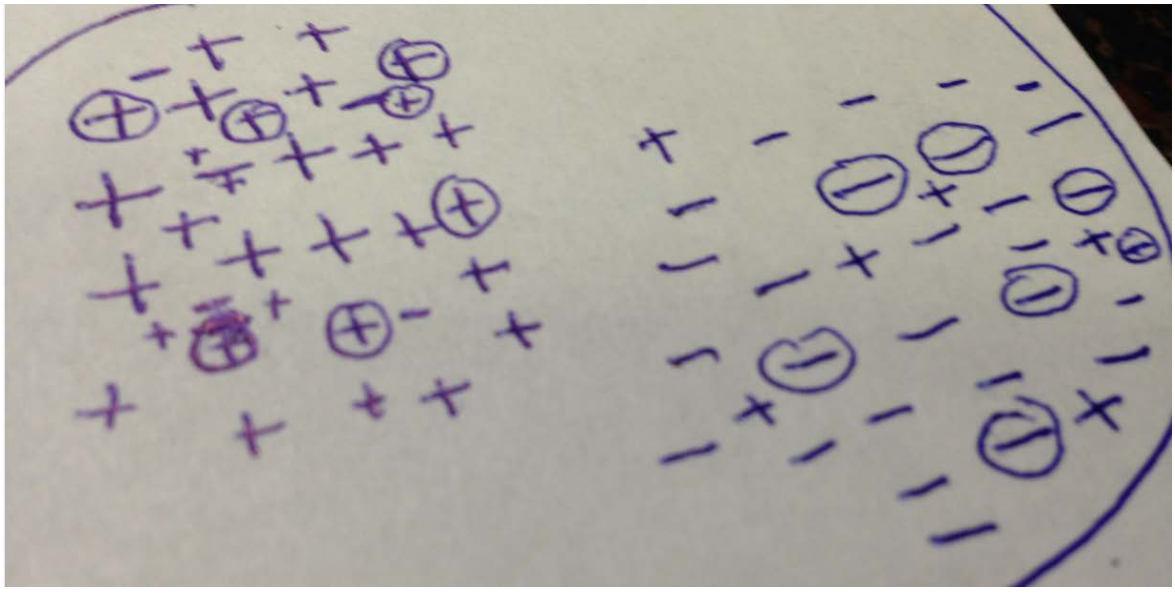
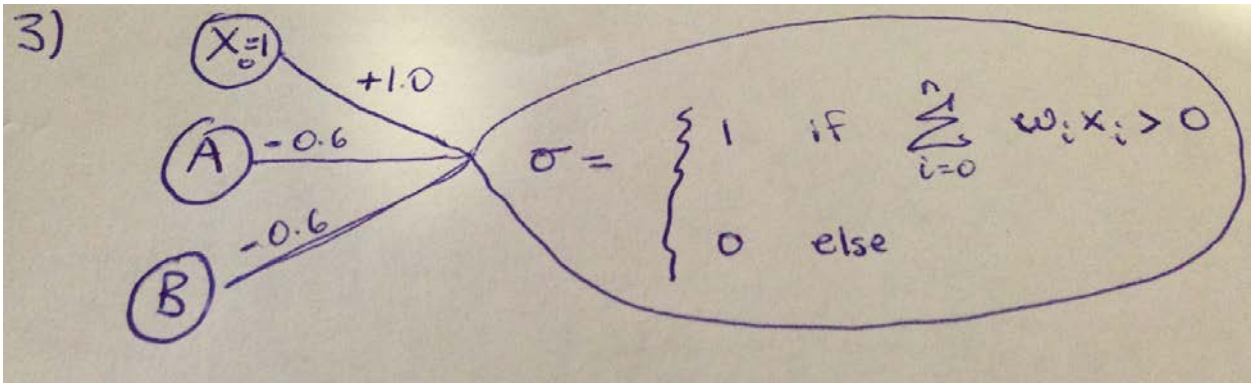
ML Pset 3

Nishant Subramani

1.
    a. Decision trees; This is because the function to be learned is in disjunctive normal form, a form indicative for decision tree performance. Two paths will be created given by each of the entities in the disjunctive normal form function ((x1 AND x3 AND –x7) and (x45 AND –x67 AND x90)) to indicate if an example goes through each of them, it will be marked as a one, otherwise a 0.
    b. Nearest neighbors; This is because similar examples will be created based on this conditional probability of 0.55 vs 0.45 depending on the outcome. An example's closest neighbors then will be those created from this same distribution and thus every predictor/parameter is important in its classification and thus nearest neighbors is the best strategy for this prediction.
    c. None; This is because the task and function to be learned seems to be a counting problem rather than a learning problem, each feature individually is not relevant, but only the aggregated count. Thus, Y cannot be learned specifically by any of the above functions and will try to be fit to some other form given by the inputs.
    d. Neural Nets; This is because a distinctive linear equation exists for y. Neural networks do a phenomenal job of being able to model linear functions which this can be transformed to.
2. A dataset with a significant percentage of outliers will tend to perform poorly in 1-nearest neighbor classification but the k = 3 k-nn classification will account for that by its voting by the three nearest neighbors. A space that has + and – outcomes is placed in a geometric space. The circled ones are the test set that is being tested, while the uncircled are the training set for the model. The correct label is given by what is in the circle. More than half of the circled test set would be predicted incorrectly by the 1-nn algorithm, while they would almost all be predicted correctly by the 3-nn algorithm.



3.



4. Crossing-Over; No where in local beam search are two states combined in some way by some linear combination to where a state is mutated such that it becomes a combination of two good states, while in genetic algorithms this is definitely the case, states in one generation combine in ways such that crossing-over materializes and states have more diversity.

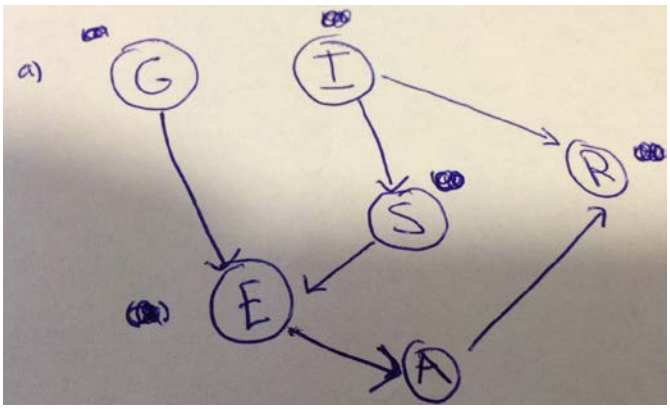
5.

   a. Genetic Algorithms are used when the equation or function to model or predict is unknown and thus the optimal solution to this unknown equation cannot be ascertained by purely mathematical means. Trying to identify the optimal walk for a robot that is tasked for a complicated task such as playing robosoccer will benefit from genetic algorithms because it uses the principles of natural selection with randomization to slowly improve as time wears on and does not rely on learning a linear function or finding an extrema.
   b. Gradient descent will be used optimally with a large number of input variables in a function that has a positive second derivative (if descent) and a negative second derivative (if ascent) across its domain. The function would hopefully be double differentiable as gradient descent works best in that case. A task such as linear regression utilizes gradient descent such that the cost function of the beta parameters for linear regression is optimized using gradient descent. Usually this cost function is optimized by MSE (mean squared error) and gradient descent performs superbly on this task as this is a function that is double differentiable and has a non-changing sign of second derivative.
   c. Hill climbing is a local search method that starts with some arbitrary solution to the problem and incrementally changes a single parameter such that the solution gets better. An ideal problem for hill climbing would be one that has only one local optima, the global optima. Conversely if any optima are desired, hill climbing works well. It tends to converge at local optima, however. One way to get around this is random-restart hill climbing. Let's assume we have patient data with 100 predictors for each patient and want to create a graph that models the data exactly (dependencies and associations). We want to ascertain the true structure of the directed acyclic graph ideally represents,

but the search space is too large, so we must use random-start hill-climbing to find the best solution. Using a large number of random restarts will guarantee to find a pretty decent solution.

6.

a.



b. **14** (1 for G, 1 for I, 2 for S, 4 for E, 4 for R, 2 for A)

c. **63** (2^num_nodes – 1; 2^6 – 1)

d. There is a G and E are connected by an edge. Thus they are dependent.

e. Since there is an edge between G and E, regardless of what nodes are in the evidence, G and E are always dependent on each other. E is **conditionally dependent** of G given A.

7.

a. Originally n=10, so based on
n_a = P(A) * n = 0.7 * 10 = 7
n_not_a = (1-P(A))*10 = 0.3*10 = 3
n = n_a + n_not_a
Now n_a += 2, so n_a = 9 and n_not_a = 3
Thus P(A) in MLE is 9/12 = **0.75**

b. P(C) = P(C|A,B) * P(A) * P(B) + P(C|A, ~B) * P(A) * P(~B) + P (C|~A,B) * P(~A) * P(B) + P(C|~A,~B) * P(~A) * P(~B)
= 1*0.7*0.5 + 1*0.7*(1-P(B)) + 0.7 * (1-P(A)) * 0.5 + 0.3*(1-P(A))*(1-P(B))
= 0.35 + 0.7 * 0.5 + 0.35 * 0.3 + 0.3 * 0.3 * 0.5
= 0.35 + 0.35 + 0.105 + 0.045
= **0.85**