Nearest Neighbor, Decision Trees, Neural Networks

Question 1.

a. Decision Trees. Because the target function is in disjunctive normal form, that's exatly what rules Decision Tree algorithm can generate. Besides we have enough data.

b. Nearest Neighbour. Because the expected value of $\sum x_i$ for both classes is fixed, the dimension of this problem can be reduced to 1, and it's linear separable. Nearest Neighbour(distance measured by $\sum x_i$) is better due to the small size of the dataset.

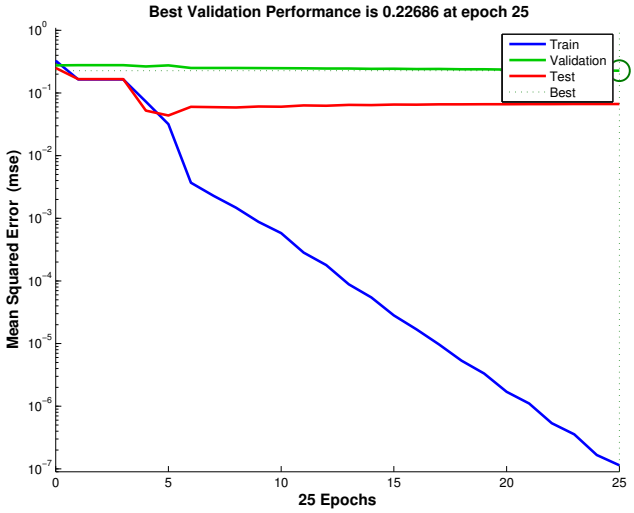
c. None. $\|\mathbf{x}\|$ is 100, so there are 2^{100} instance of $\mathbf{x}$ with different permutation of 0 and 1. Thus instance-based approach Nearest Neighbour is hit by curse of dimensinality. The same with Decision Trees, because to approach the function in disjunctive normal form the complexity is the same. How about the magical Neural Networks? Maybe it will work with 100 layers, but we should consider data we have and overfitting. This function is hard to learn, because each feature is independent, and you can gain none information by individual feature untill you check all the features.

d. Neural Networks. The function looks complex, but all the formula can be represent by Neural Networks, because the feature is binary, $x_i * x_j$ equal to $x_i \; and \; x_j$, $x_i/(x_j + \epsilon)$ can also be represent with x_0 and hidden layer.

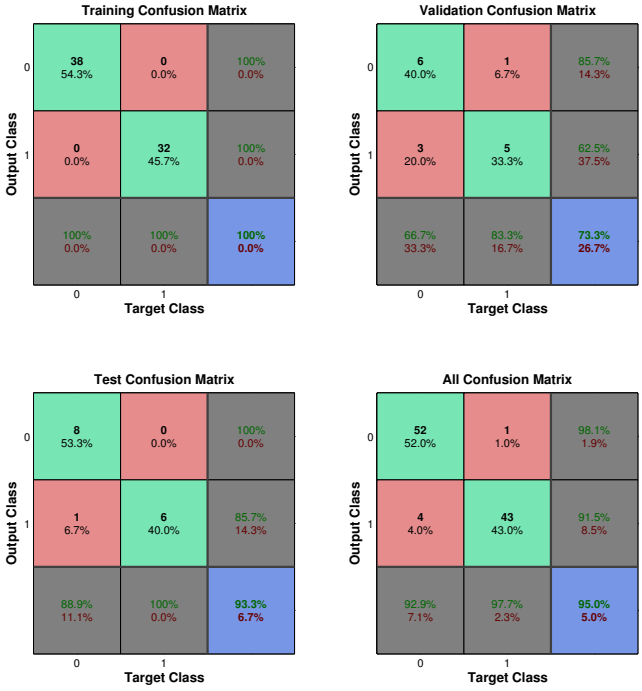
For b and d, I also make experiments(using Matlab) comparing the performance of Nearest Neighbour and Neural Networks, because I think both classifiers can work with the given function and data. For b, the original Nearest Neighbour(without using $\sum x_i$ as distance measurement, because obiviously the NN described in b can work, I don't think it needs to be test) achieve 50% percision, while Neural Networks(with 10 neurons and one hidden layer) achieve 93% percision(as figure 1 shows). For d, my experiment show Nearest Neighbour achieve percision around 70%, while Neural Networks achieve 100% percision on both training and test set(both have 10000 instances). I think the reason behind this result is that I do not add ϵ(noise) in the data. I randomly generate the data, then assign instance with target value by exatly the target function. In this case, the result means Neural Networks can perfectly represent the target function, while nearest neighbour do have an effect, but due to it's an instance-based approach, it can not represent the target function very well.

Question 2.

The dataset shown in figure 2:



(a) Error of Neural Network on b.



(b) Confusion Matrix of Neural Network on b.

Figure 1: Some result of my tiny experiment.

x_1	x_2	y
1	1	1
1	2	1
2	2	1
-1.25	-1.75	1
-1	-1	0
-1	-2	0
-2	-2	0
1.25	1.75	0

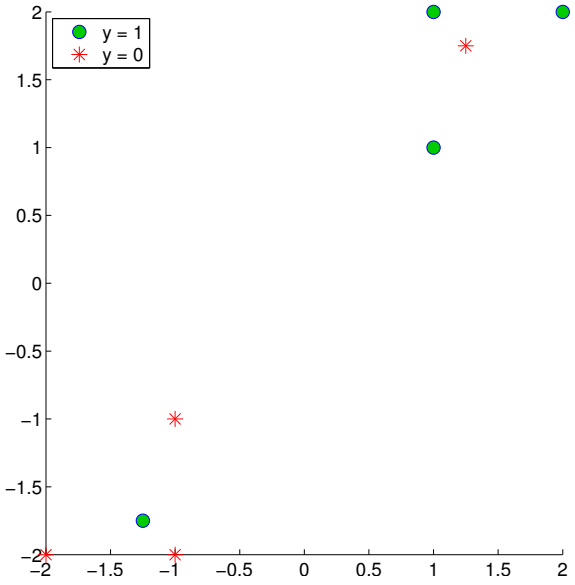


Figure 2: The dataset for Question 2

Perceptrons

Question 3.

See figure 3.

Optimization

Question 4.

Mutation and crossover.

Question 5.

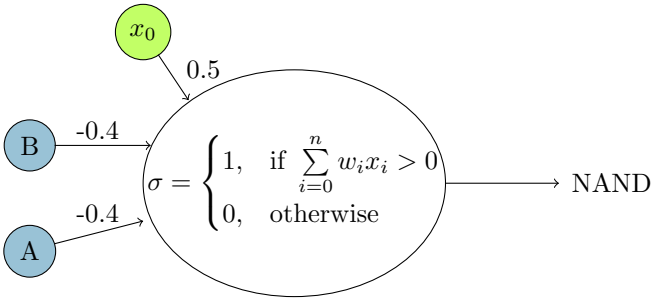


Figure 3: NAND Perceptron

a. The optimization problem is:

$$\arg\min_{x} f(x)$$

where $f(x)$ is a sawtooth wave function as figure 4 shows (in discrete form, we can think this is a digital signal).

The sawtooth(triangle also) wave has properties that every local minima or

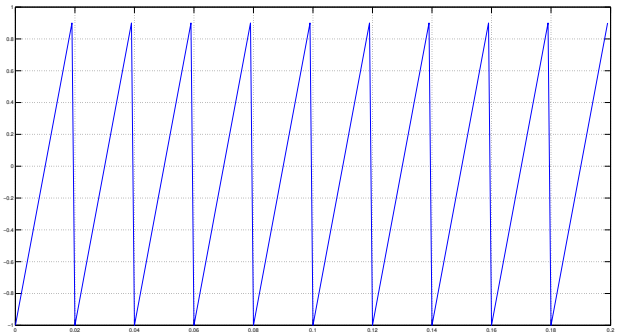


Figure 4: Sawtooth wave.

maxima is also global minima or maxima and it's non differentiable in some points. Gradient Descent can not handle non differentiable function, and to find a local minima Hill Climbing has a better efficiency and more reliable than Gentic Algorithms.

So Hill Climbing is a better choice for this problem.

b. Find the minimal of the graph (generated by data interpolation), and we know it approximates to $f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x}^2 + \mathbf{y}^2 + \mathbf{z}^2)$ as figure 5 shows.

For this problem, Gradient Descent is better, since the function is continous and convex. And it may converge faster than Hill Climbing, because this is a 3-dimension problem, in each iteration hill climbing will try to make change

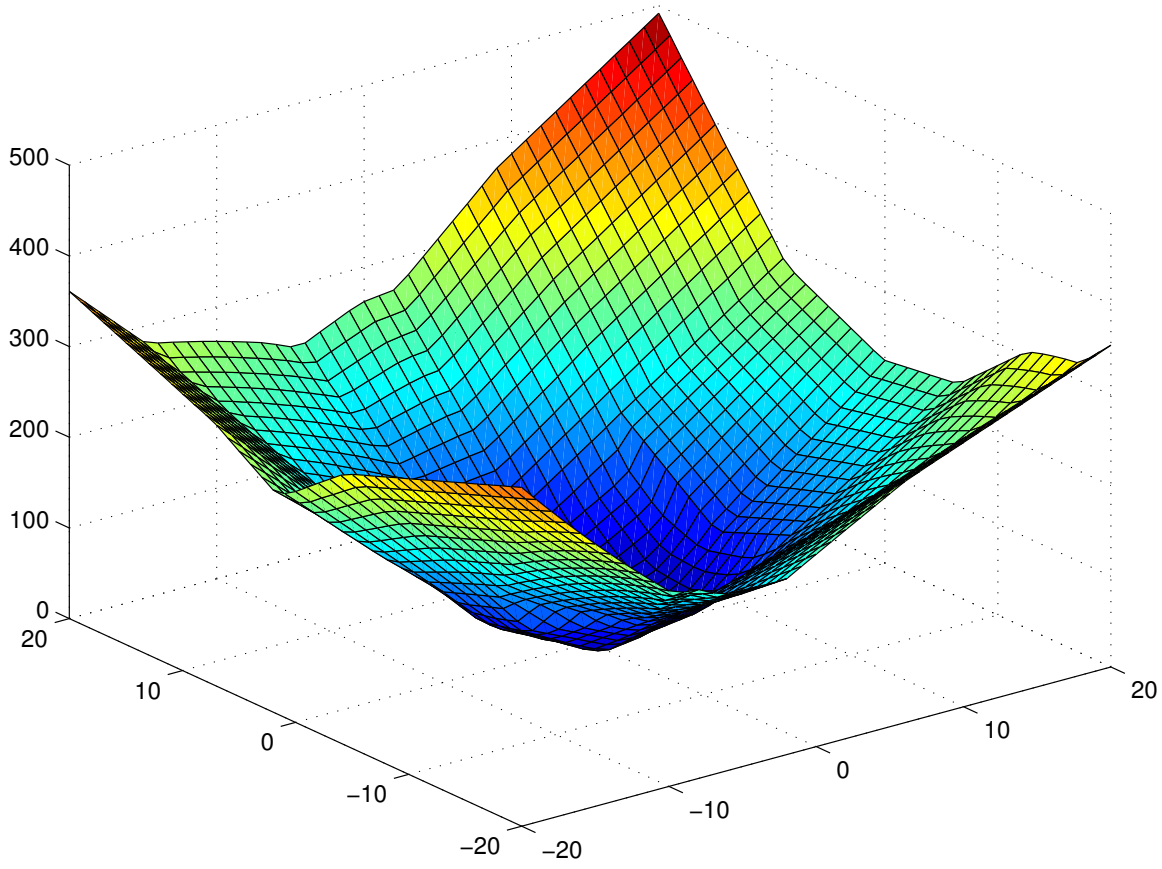


Figure 5

in one dimension, while gradient descent will go forward on the direction with steepest gradient.

c. Also an unconstrained optimization problem: find the minimal of $f(x)$. But here $f(x)$ is non-convex and has saddle point(s), so both Hill Climbing and Gradient Descent will not work well. We can suppose this problem occur when we want to use manifold learning for feature selection. Thus encoding the feature vector is not difficult for Genetic Algorithms.

   So for this problem, Genetic Algorithms is better.

Bayes Nets and Statistical Estimation

Question 6.

a. See figure 6.

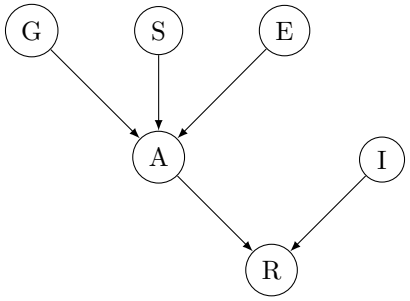


Figure 6: Bayes Net.

b. $1 + 1 + 1 + 2^3 + 2^2 = 3 + 8 + 4 = 15$

c. $2^6 - 1 = 63$

d. Yes, because E is not a descendant of G and vice versa, and both G and E do not have parent in the graph, so they are uncorrelated.

e. Yes, because if we know A, E and G is d-separated by A, so E and G are conditional independent given A.

Question 7.

a.

$$\begin{aligned} P(C) &= \sum_i \sum_j P(C|A_i, B_j)P(A_i, B_j) \\ &= \sum_i \sum_j P(C|A_i, B_j)P(A_i)P(B_j) \\ &= 0.35 + 0.35 + 0.7 * 0.3 * 0.5 + 0.3 * 0.3 * 0.5 \\ &= 0.85 \end{aligned}$$

b. $9/12 = 0.75$