1) Explain Naive Bayes classifier & Bayesian Relief Networks

Naive Bayes classifier:

→ Naive Bayes classifier algorithm is a supervised learning algorithm, which is based on Bayes theorem & used for solving classification problems.

→ The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values & where the target function $f(x)$ can take on any value from some finite set V.

→ A set of training examples is provided, and a new instance is presented, described by the tuple of attribute values $(a_1, a_2, \ldots a_n)$.

→ The learner is asked to predict the target value, for this new instance

→ The Bayesian approach to classifying the new instance is to assign the most probable target value V_{MAP}, given the attribute values $(a_1, a_2 \ldots a_n)$ that describe the instance.

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \; P(v_j | a_1, a_2 \ldots a_n)$$

→ By Bayes theorem:

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \; \frac{P(a_1, a_2, \ldots a_n | v_j) \, P(v_j)}{P(a_1, a_2 \ldots a_n)}$$

$$= \underset{v_j \in V}{\operatorname{argmax}} \; P(a_1, a_2 \ldots a_n | v_j) \, P(v_j)$$

→ It is easy to estimate each of the $P(v_j)$ simply by counting the frequency with which each target value v_j occurs in the training data.

→ However, estimating the different $P(a_1, a_2 \ldots a_n | v_j)$ terms is not feasible unless we have a very, very large set of training data.

→ Naive Bayes classifier is based on simplifying assumption that the attribute values are conditionally independent given the target value.

*) Naive Bayes classifier : $V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod P(a_i/v_j)$

Bayesian Belief Networks:

→ The naive Bayes classifier makes significant use of assumption that the values of the attributes $a_1, \ldots a_n$ are conditionally independent given the target value v.

⇒ This assumption dramatically reduces the complexity of learning the target function. When it is met, the naive Bayes classifier outputs the optimal Bayes classification.

→ However, in many cases this conditional independence asumption is clearly overly restrictive.

→ A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities.

→ In contrast to the naive Bayes classifier, which assumes that all the variables are conditionally independent given the value of the target variable, Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables.

→ It provide an intermediate approach that is less constraining than the global assumption of conditional independence made by the naive Bayes classifier, but more tractable than avoiding conditional independence assumptions altogether.

→ In general, a Bayesian belief network describes the probability distribution over a set of variables.

→ Consider an arbitary set of random variables $Y_1 \ldots Y_n$, where each variable Y_i can take on the set of possible values $V(Y_i)$.

→ We define the joint space of set of variables Y to the cross product $V(Y_1) \times V(Y_2) \times \ldots V(Y_n)$.

→ The probability distribution over this joint space is called the joint probability distribution.

→ The joint probability distribution specifies the probability for each of the possible variable bindings for the tuple $(Y_1, \ldots Y_n)$.

2) What is instance based learning. Explain KNN algorithm with an example.

→ Instance based learning methods simply store the training examples instead of learning explicit description of the target function.

→ It is also known as memory-based learning or lazy-learning. The time complexity of this algorithm depends upon the size of training data. The worst-case time complexity of this algorithm is $O(n)$, where n is number of training instances.

→ Instance-based learning includes nearest neighbor, locally weighted regression & case-based reasoning methods.

→ A key advantage of lazy learning is that instead of estimating the target function once for the entire instance space, these methods can estimate it locally & differently for each new instance to be classified.

→ Instance based learning is a supervised classification learning algorithm.

KNN Algorithm:

→ KNN is called Lazy learner as it doesnot learn much in training.

→ Here it checks all the nearest neighbors & consider the unknown input as major neighbors which are present in that set

→ The nearest neighbors of an instance are defined in terms of Euclidean distance

$$Dist(X, Y) = \sqrt{\sum_{i=1}^{D} (X_i - Y_i)^2}$$

→ For example there are 2 subjects A & B, the result will be pass or fail in the exam.

S. No	A	B	Result
1	4	3	Fail
2	6	7	Pass
3	7	8	Pass
4	5	5	Fail
5	8	8	Pass

→ This is a table of 5 students along with their marks & result

→ Now based on the table a new instance arrived whose marks in A is 6 & marks in B is 8 & given K = 3 (in KNN)

i) $\sqrt{(6-4)^2+(8-3)^2} = \sqrt{29} = 5.38$

ii) $\sqrt{(6-6)^2+(8-7)^2} = 1$

iii) $\sqrt{(6-7)^2+(8-8)^2} = 1$

iv) $\sqrt{(6-5)^2+(8-5)^2} = \sqrt{10} = 3.16$

v) $\sqrt{(6-8)^2+(8-8)^2} = 2$

Here we can see that (ii) & (iii) have same value, so they are nearest neighbours & the next nearest neighbour is 2. So we had identified 3 neighbours

→ We got 3 neighbours (ii), (iii) and (v) & all those 3 neighbours are passed So based on this classification we can conclude that the new instance A=6, B=8 has passed the exam.

→ If we can't classify the result in the first step then we again need to find the Euclidean distance between the K clusters & choose the nearest neighbour.

neighbour.

3) Discuss the method of learning using locally weighted linear regression.

→ Linear regression is a supervised learning algorithm used for computing linear relationships between input(X) & output(Y).

→ The basic assumption for a linear regression is that the data must be linearly distributed.

→ We can even apply for non-learly distributed data & it is called as locally weighted regression

→ locally weighted linear regression is a non-parametric algorithm, that is, the model does not learn a fixed set of parameters as is done in ordinary linear regression

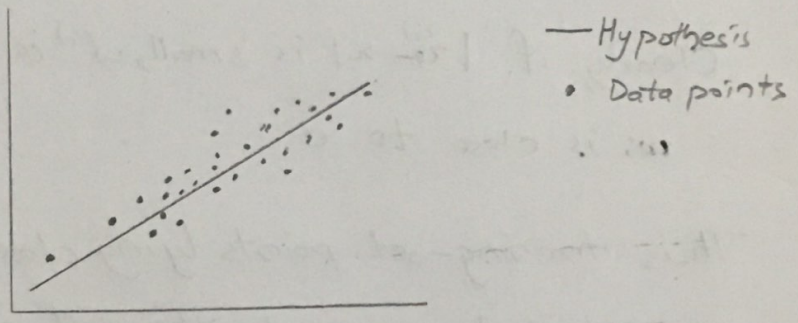
→ The steps involved in ordinary linear regression are:

Training phase: Compute θ to minimize the cost.

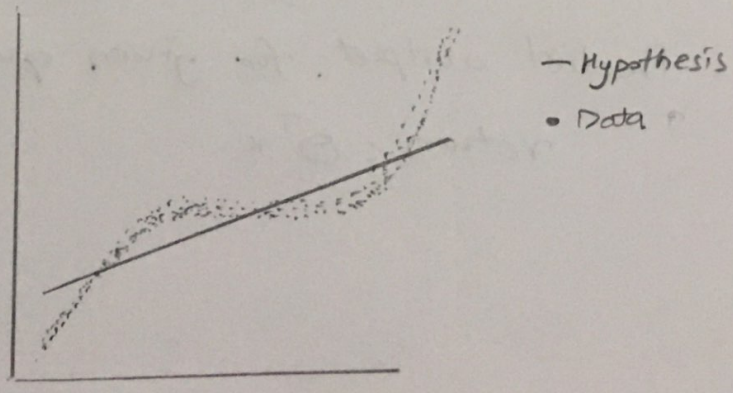
$$J(\theta) = \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2$$

Predict output: for given query point x,

return : $\theta^T x$



→ As evident from image below, this algorithm cannot be used i.e, linear regression for making predictions when there exists a non-linear relationship between X & Y. In such cases, locally weighted linear regression is used



→ So the modified cost function is

$$J(\theta) = \sum_{i=1}^{m} w^{(i)} (\theta^T x^{(i)} - y^{(i)})^2$$ where, $w^{(i)}$ is a non-negative "weight" associated with training point $x^{(i)}$

→ For $x^{(i)}$'s lying closer to the query point x, the value of $w^{(i)}$ is large, while for $x^{(i)}$'s lying far away from x the value of $w^{(i)}$ is small.

→ A typical choice of $w^{(i)}$ is : $w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$

where, τ is called bandwidth parameter & controls the rate at which $w^{(i)}$ falls with distance from x.

Clearly, if $|x^{(i)} - x|$ is small, $w^{(i)}$ is close to 1 & if it is large w_i is close to 0.

Thus training-set points lying closer to the query point x contribute more to the cost $J(\theta)$ than the points lying far away from x

Steps involved in locally weighted linear regression are:

Compute θ to minimize the cost

$$J(\theta) = \sum_{i=1}^{m} w^{(i)} (\theta^T x^{(i)} - y^{(i)})^2$$

Predict output: for given query point x,

return : $\theta^T x$

For example, Consider a query point $x^{(1)}$ and $x^{(2)}$ be two points in the training set such that

$x^{(1)} = 4.9$ & $x^{(2)} = 3.0$

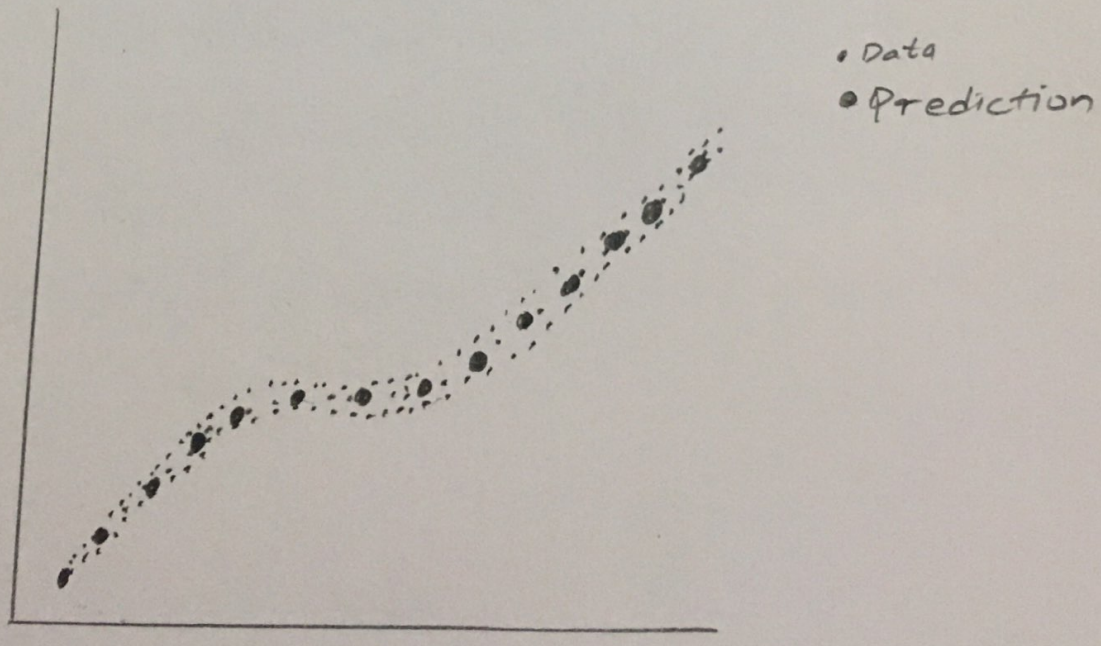
Using the formula $w^{(i)} = \exp\left(\frac{-(x^{(i)} - x)^2}{2\tau^2}\right)$ with $\tau = 0.5$

$$w^{(1)} = \exp\left(\frac{-(4.9-5.0)^2}{2(0.5)^2}\right) = 0.9802$$

$$w^{(2)} = \exp\left(\frac{-(3.0-5.0)^2}{2(0.5)^2}\right) = 0.000335$$

So, $J(\theta) = 0.9802 * (\theta^T x^{(1)} - y^{(1)}) + 0.00035 * (\theta^T x^{(2)} - y^{(2)})$

→ Thus the weights falls exponentially as the distance between x & $x^{(i)}$ increases & so does the contribution of error in prediction for $x^{(i)}$ to the cost.



4) Describe the ~~the~~ genetic algorithm steps using population fitness function & other necessary data.

→ Genetic Algorithms is to search a space of candidate hypothesis to identify the best hypothesis.

→ In Genetic Algorithms the "best hypothesis" is defined as the one that optimizes a predefined numerical measure for the problem at hand called the hypothesis fitness

→ The algorithm operates by iteratively updating a pool of hypotheses called the population

Prototypical Genetic Algorithm:

- Initialize Population: P ← generate p hypotheses at random
- Evaluate: for each h in P, compute fitness(h)
- While $\max_h$ Fitness(h) < Threshold do
- Create a new generation P_{New}
    - *) Select
    - *) Crossover
    - *) Mutate
    - *) Update
    - *) Evaluate
- Return the hypothesis from P that has the highest fitness

→ Once the initial generation is created, the algorithm evolves using following operators.

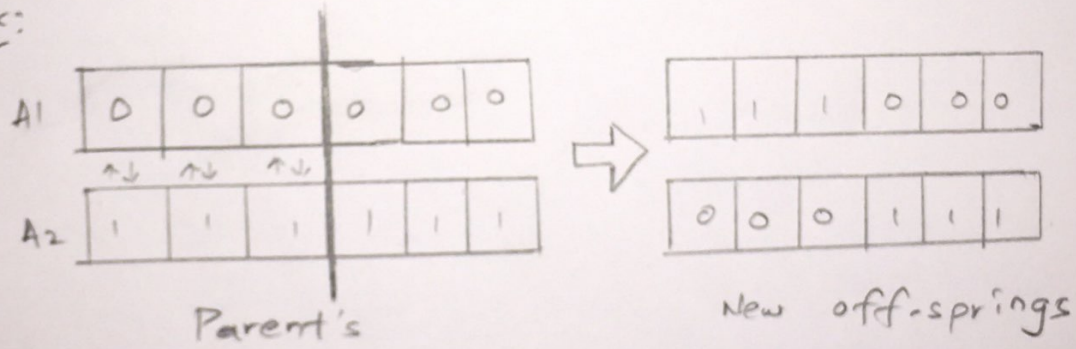
i) Select:

The idea is to give preference to the individuals with good fitness scores & allow them to pass their genes to successive generations

ii) Crossover:

Crossover is the most significant phase in genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.

Ex:



→ In the above example if we take crossover point as 3, then offsprings are created by exchanging the genes of parents among themselves until the crossover point is reached.

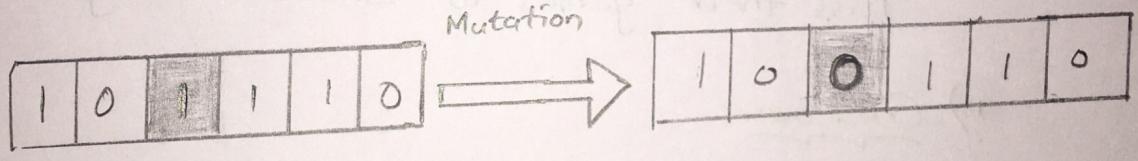
→ Then add all offsprings to P_{New}.

iii) Mutate:

Choose m·l. of P_{new} with uniform probability.

For each, invert one randomly selected bit in its representation

Ex:



iv) Update: $P \leftarrow P_{New}$

v) Evaluate: For each p in P_{New}, compute fitness (p)

5) Explain different types of models of evolution & learning

Models of evolution:

- Lamarckian Evolution:

→ Lamarck was a scientist that evolution over many generations was directly influenced by the experiences of individual organisms during their lifetime.

→ He proposed that experiences of a single organism directly affected the genetic makeup of their offspring.

→ This is an alternative conjencture, because it would presumably allow for more effecient evolutionary progress than a generate-and-test process that ignores the experience gained during an individual's lifetime.

- Baldwin Effect:

→ Although Lamarck evolution is not an accepted model of biological evolution, other mechanisms have been suggested by which individual learning can alter the course of evolution.

→ One such mechanism is called Baldwin effect, after J.M. Baldwin (1986), who first suggested the idea.

→ The Baldwin effect is based on following observations:

*) If a species is evolving in a changing environment, there will be evolutionary pressure to favor individuals with the capability to learn during their lifetime. For example, if a new predator appears in the environment, then individuals capable of learning to avoid the predator will be more successful than individuals who cannot learn. In effect, the ability to learn allows an individual to perform a small local search during its lifetime to maximize its fitness.

Models of Learnings:

Foil:

→ FOIL is the natural extension of SEQUENTIAL-COVERING & LEARN-ONE-RULE to first order rule learning.

→ FOIL learns first order rules which are similar to Horn clauses with 2 exceptions:

- Literals may not contain function symbols
- Literals in body of clause may be negated

→ LIKE Sequential-Covering, foil learns one rule at time & removes positive examples covered by the learned rule before attempting to learn a further rule.

- Sequential Covering Algorithms:

→ Sequential Covering algorithms for learning rule sets based on the strategy of learning one rule, removing the data it coveres, then iterating the process.

→ Propositional logic does not include variables & thus cannot express general relations among the values of the attributes.

→ Ex1: In propositional logic, you can write:

- If (father = Bob) ^ (Name = Bob) ^ (Female = True) then Daughter = True.

This rule applies only to a specific family!

→ Ex2: In first-order logic, you can write:

- If Father(y,z) ^ Female(y), Then Daughter(x,y)

This rule applies to any family!

→ Both approaches to learning are useful as they address different types of learning problems.

6) What is Brute force MAP hypothesis. How it is related to concept learning.

Bruteforce MAP:

→ For each hypothesis h in H, calculate the posterior probability

$$P(h/D) = \frac{P(D/h)\,P(h)}{P(D)}$$

→ Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{map} = \underset{h \in H}{\operatorname{argmax}}\ P(h/D)$$

→ Bruteforce MAP learning algorithm must specify values for P(h) & P(D/h).

→ P(h) & P(D/h) must be chosen to be consistent with the assumptions.

i) The training data D is noise free.

ii) The target concept c is contained in the Hypothesis space H.

iii) We have no a prior reason to believe that any hypothesis is more probable than other.

→ With the assumptions:

$$P(H) = \frac{1}{|H|} \quad \forall h \text{ in } H$$

$$P(D/h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases}$$

→ So, the values of $P(h/D)$ will be:

$$P(h/D) = \frac{0 \cdot P(h)}{P(D)} = 0, \text{ if } h \text{ is inconsistent with } D$$

$$P(h/D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)} = \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

if h is consistent with D

$$P(h/D) = \frac{1}{|VS_{H,D}|}$$

→ $P(D) = \frac{|VS_{H,D}|}{|H|}$ because

- the sum over all hypotheses of $P(h/D)$ must be one and the number of hypotheses from H consistent with D is $|VS_{H,D}|$, or
- we can derive $P(D)$ from the theorem of probability and the fact that the hypotheses are mutually exclusive. (i.e, $(\forall i \neq j)(P(h_i \wedge h_j) = 0)$

$$P(D) = \sum_{h_i \in H} P(D|h_i) P(h_i) = \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + 0 \cdot \sum_{h_i \notin VS_{H,D}}$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} = \frac{|VS_{H,D}|}{|H|}$$

Relation to concept learning:

→ In concept learning we have an instance X, hypothesis space H, training examples D, The find S algorithm finds the most specific hypothesis from $VS(H, D)$.

→ Assume fixed set of instances $\langle x_1, x_2 \dots x_m \rangle$

→ Assume D is the set of classifications $D = \langle (x_1), \dots (x_m) \rangle$

Choose $P(D/h)$

- $P(D/h) = 1$ if h is consistent with D
- $P(D/h) = 0$ otherwise

Choose $P(h)$ to be uniform distribution

$$P(h) = \frac{1}{|H|} \quad \forall h \text{ in } H$$

then

$$P(h/D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$