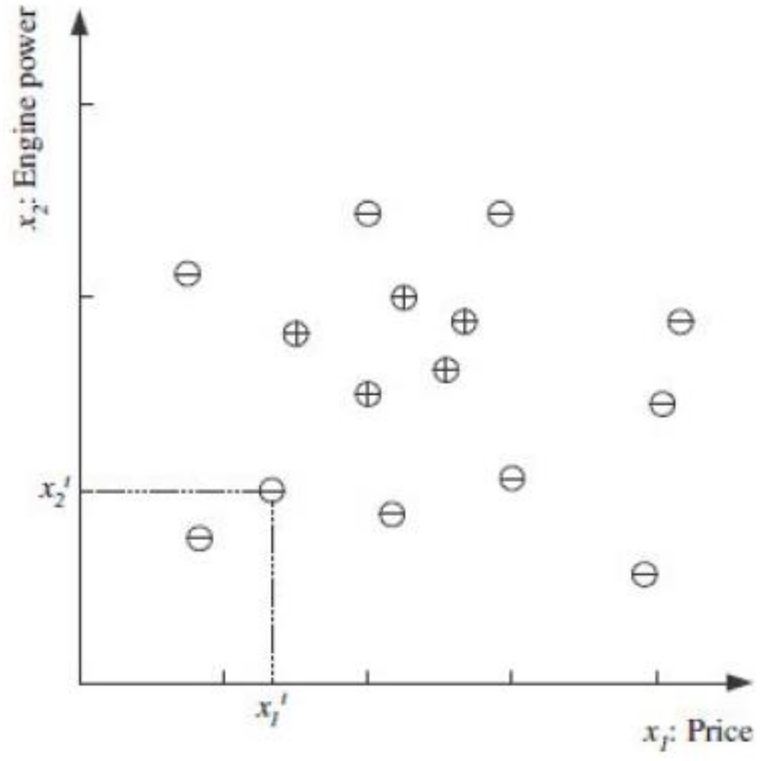
UE19EC353-Machine Learning
UNIT - 2

Raghavendra.M.J
Department of ECE, PESU.

Learning Class From Examples



C= Class C of a family car

- + denotes a positive example of a family car. It is with in the range of Price and Engine Power
- - denotes a negative example of a family car . It is outside the range of Price and Engine Power
- Each point corresponds to one example car
- Example: $x_1' = 4$ lakhs and $x_1 = 14$ lakhs
  $x_2' = 2$ hp and $x_2 = 8$ hp

- Training data (x_1^t, x_2^t),[Example (3,2.5)] is a point and it may be positive or negative and it is given by r^t

Learning Class From Examples

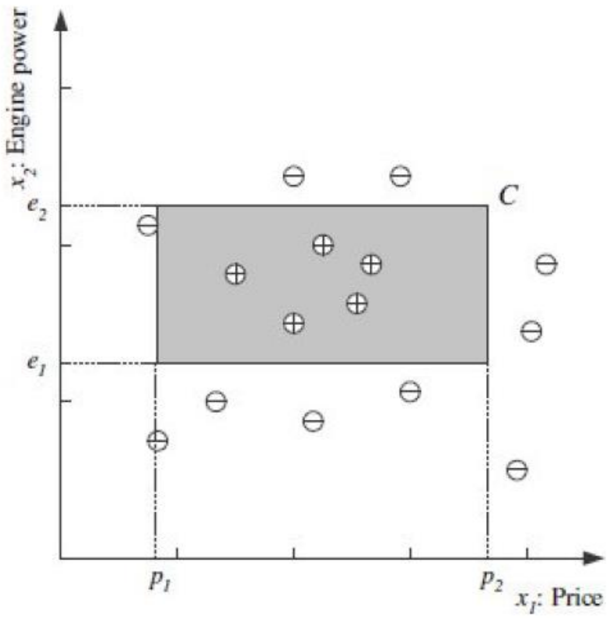
- Each car is represented by $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $x_1 = Price\ and\ x_2 = Engine\ Power$
- Each Car is labelled as follows

$$r = \begin{cases} 1\ if\ x\ is + ve\ example \\ 0\ if\ x\ is\ - ve\ example \end{cases}$$

- Each car is represented by pair (x , r)
- Training Set Contains N – Such examples
- $X = \{x^t, r^t\}_{t=1}^{t=N}$

MACHINE LEARNING Unit –1

Learning Class From Examples



- Training data (x_1^t, x_2^t) is a point and it may be positive or negative and it is given by r^t

- <u>Hypothesis Class</u>
- $p_1 = Lower\ limit\ of\ the\ price$
- $p_2 = Higher\ limit\ of\ the\ price$
- $e_1 = Lower\ limit\ of\ the\ engine\ power$
- $e_2 = Upper\ limit\ of\ the\ engine\ power$
- Let H be the Hypothesis class such that
- $(p_1 \leq price \leq p_2) AND (e_1 \leq Engine\ Power \leq e_2$
- EX.$(4\text{ lakh} \leq price \leq 14\ lakh) AND\ \ 2 \leq Engine\ Power \leq 8)$

Learning Class From Examples

- The learning algorithm finds the particular hypothesis $h \in H$
- It is specified by $(p_1^h, p_2^h, e_1^h, e_2^h)$, EX.$(4 \text{ lakh} \leq price \leq 14\ lakh) AND\ \ 2 \leq Engine\ Power \leq 8)$
- $h \in H$ is equal to C or closet to C
- Aim is to find $h \in H$ is as similar as possible to C. Learning implies finding the four parameters of h.
- $$h(x) = \begin{cases} 1\ if\ h\ classifies\ x\ as + ve\ example \\ 0\ if\ h\ classifies\ x\ as\ - ve\ example \end{cases}$$

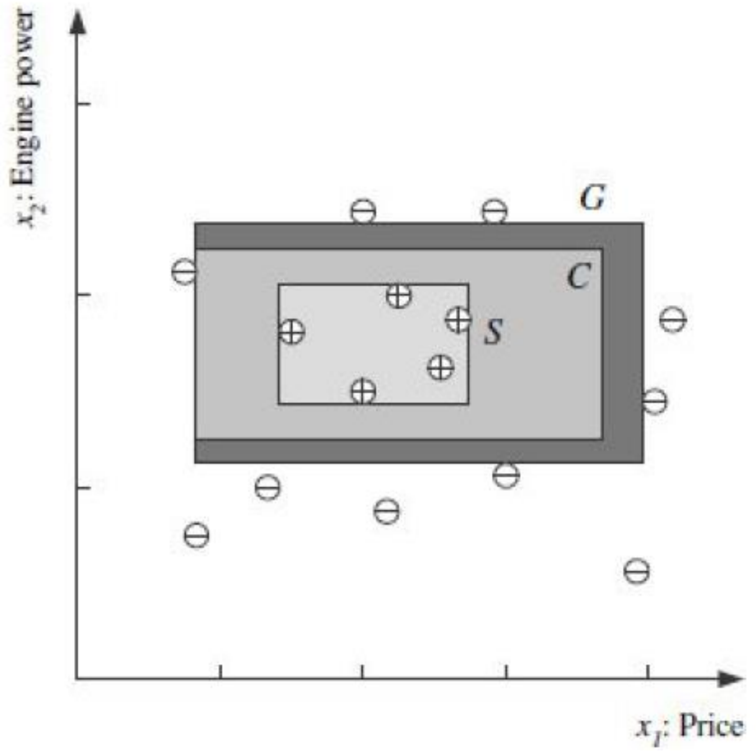
Learning Class From Examples

- In real life C(x) is not known .We can not evaluate how well h(x) matches C(x).Therefore we have to evaluate empirical error which is the proportion of training instances.
- The Error of hypothesis h is given the training set X is
- $E(h|X) = \sum_{t=1}^{t=N} 1(h(x^t) \neq r^t)$
- $1(h(x^t) \neq r^t) is\ 1\ if h(x^t) \neq r^t$
- $1(h(x^t) \neq r^t) is\ 0\ if h(x^t) = r^t$
- We have to select this h such that E=0

MACHINE LEARNING Unit –1

Learning Class From Examples

❖ **Most Specific Hypothesis S, Most General Hypothesis G and Version Space**



❖ S is the Most Specific Hypothesis

❖ G is the Most General Hypothesis

❖ Any $h \in H$ between S and G is a valid hypothesis with no error, said to be consistent with the training set and such h make up the Version space

❖ S :$(8 \text{ lakh} \leq price \leq 10 \ lakh) AND \ \ (4 \leq Engine\ Power \leq 6)$

❖ G:$(4 \text{ lakh} \leq price \leq 14 \ lakh) AND \ \ (2 \leq Engine\ Power \leq 8)$

❖ C:$(4 \text{ lakh} \leq price \leq 13 \ lakh) AND \ \ (2.5 \leq Engine\ Power \leq 7.5)$

MACHINE LEARNING Unit –1

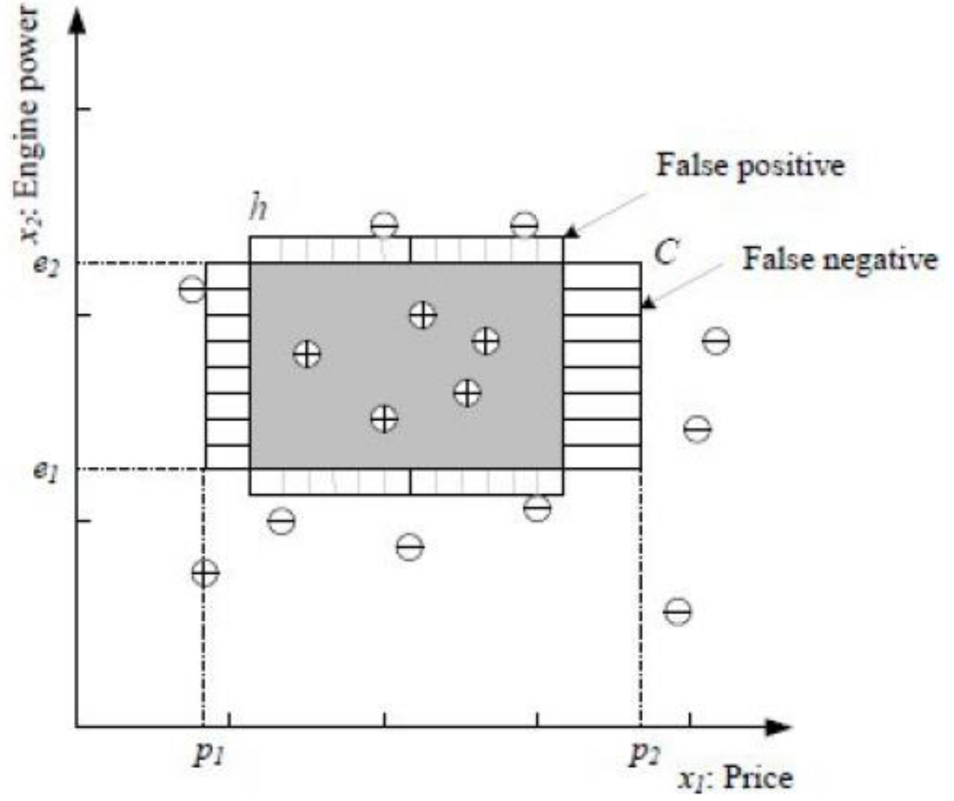
Learning Class From Examples

False Positive and False Negative

- C is the Actual Class
- h is the induced hypothesis
- The point where C is 1 and h is 0 is False Negative . Price=12.75 lakhs Engine Power=4

- The point where C is 0 and h is 1 is False Positive. Price=10 lakhs Engine Power=7.9

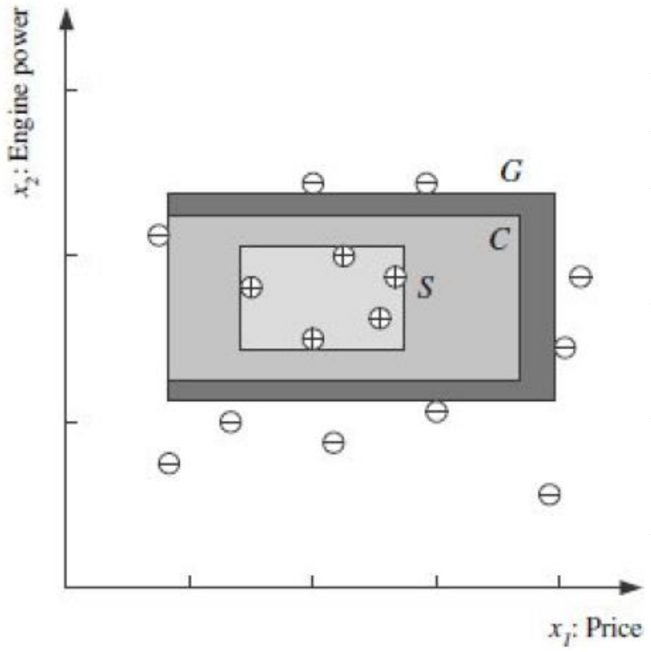
- C:$(4 \text{ lakh} \leq price \leq 13\ lakh) AND\ (2.5 \leq Engine\ Power \leq 7.5)$
- h:$(5 \text{ lakh} \leq price \leq 12.5\ lakh) AND(\ 3 \leq Engine\ Power \leq 7.75)$



MACHINE LEARNING Unit –1

Learning Class From Examples

- Actually depending Training Set X and Hypothesis Class X, there may be several S_i and G_j

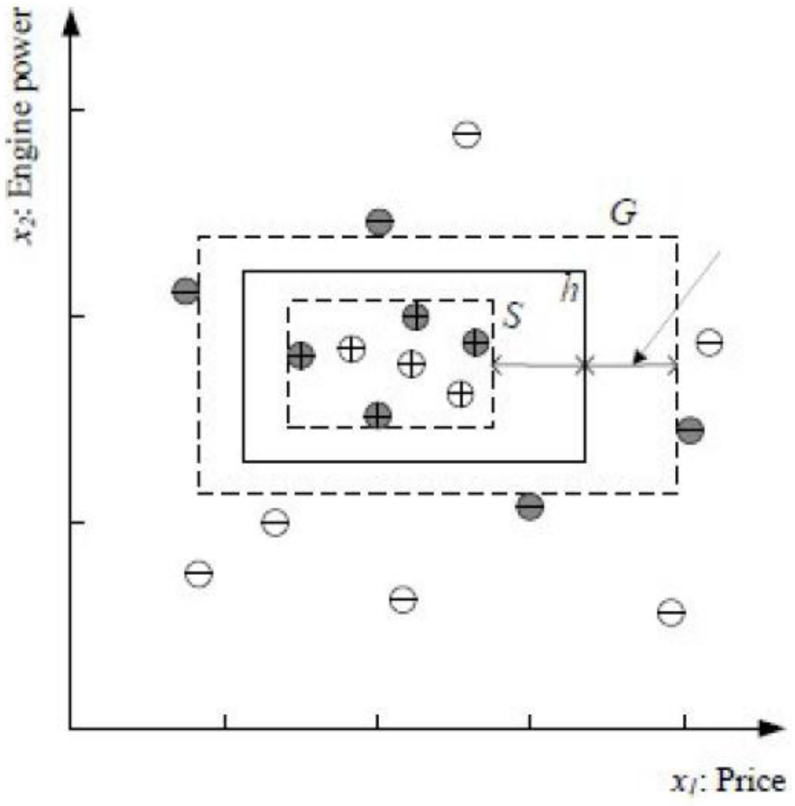


- Several S_i make up S-Set
- Several G_i make up G-Set
- Every member of the S-Set is consistent with all the instances and that there are no consistent hypothesis that are more Specific.
- Every member of the G-Set is consistent with all the instances and that there are no consistent hypothesis that are more General.
- These two make up the boundary sets and any hypothesis between them is consistent and is a part of version space.
- There is an algorithm called Candidate Elimination that incrementally updates the S-Set and G-Set as it sees training instances one by one.
- We assume that training set is large enough that there is a unique S and G.

MACHINE LEARNING Unit –1

Learning Class From Examples

- **Doubt**

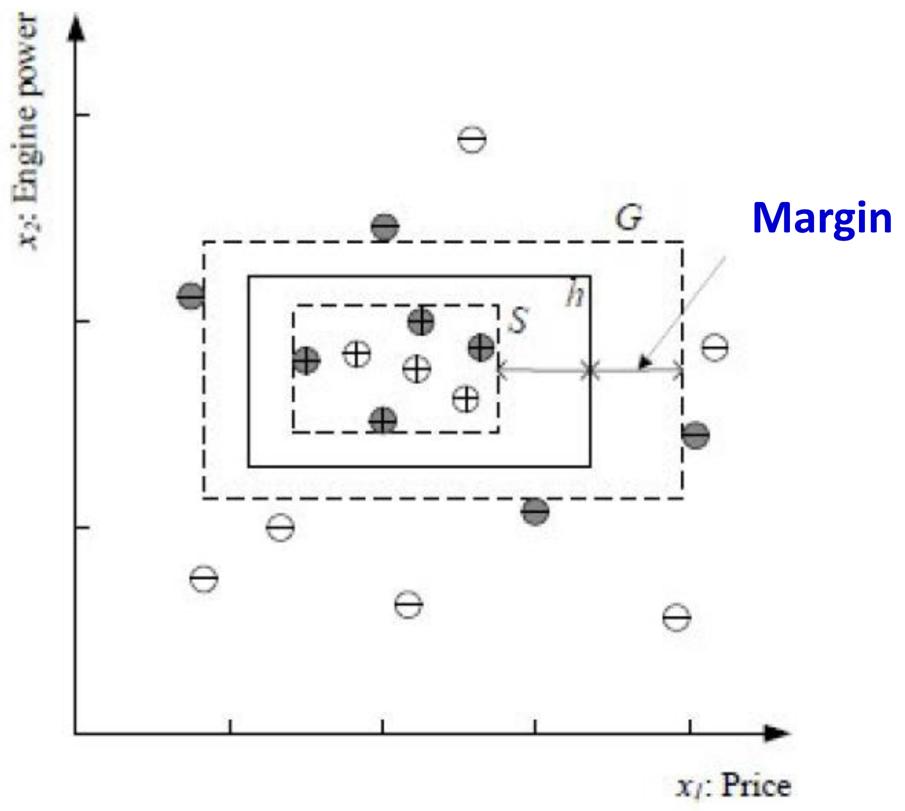


- **In some applications, there may be hypothesis that falls between S and G is a case of doubt which we cannot label due to lack of data. In such cases , the system rejects the instance and defers the decision to human expert**

- S :$(8 \text{ lakh} \leq price \leq 10\ lakh) AND\ (4 \leq Engine\ Power \leq 6)$
- G:$(4 \text{ lakh} \leq price \leq 14\ lakh) AND\ (2 \leq Engine\ Power \leq 8)$
- h:$(4 \text{ lakh} \leq price \leq 13.9\ lakh) AND\ (2.5 \leq Engine\ Power \leq 7.95)$

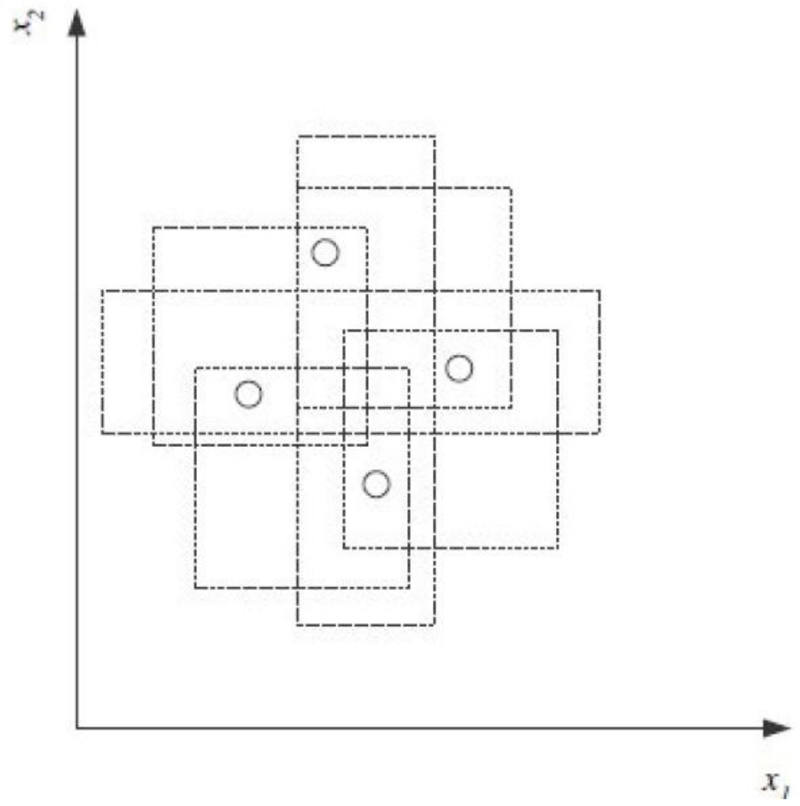
Learning Class From Examples

❖ **Margin**



- We chose the hypothesis with the largest margin, for best separation.
- The shaded instances are those that define margin, other instances can be removed without affecting h

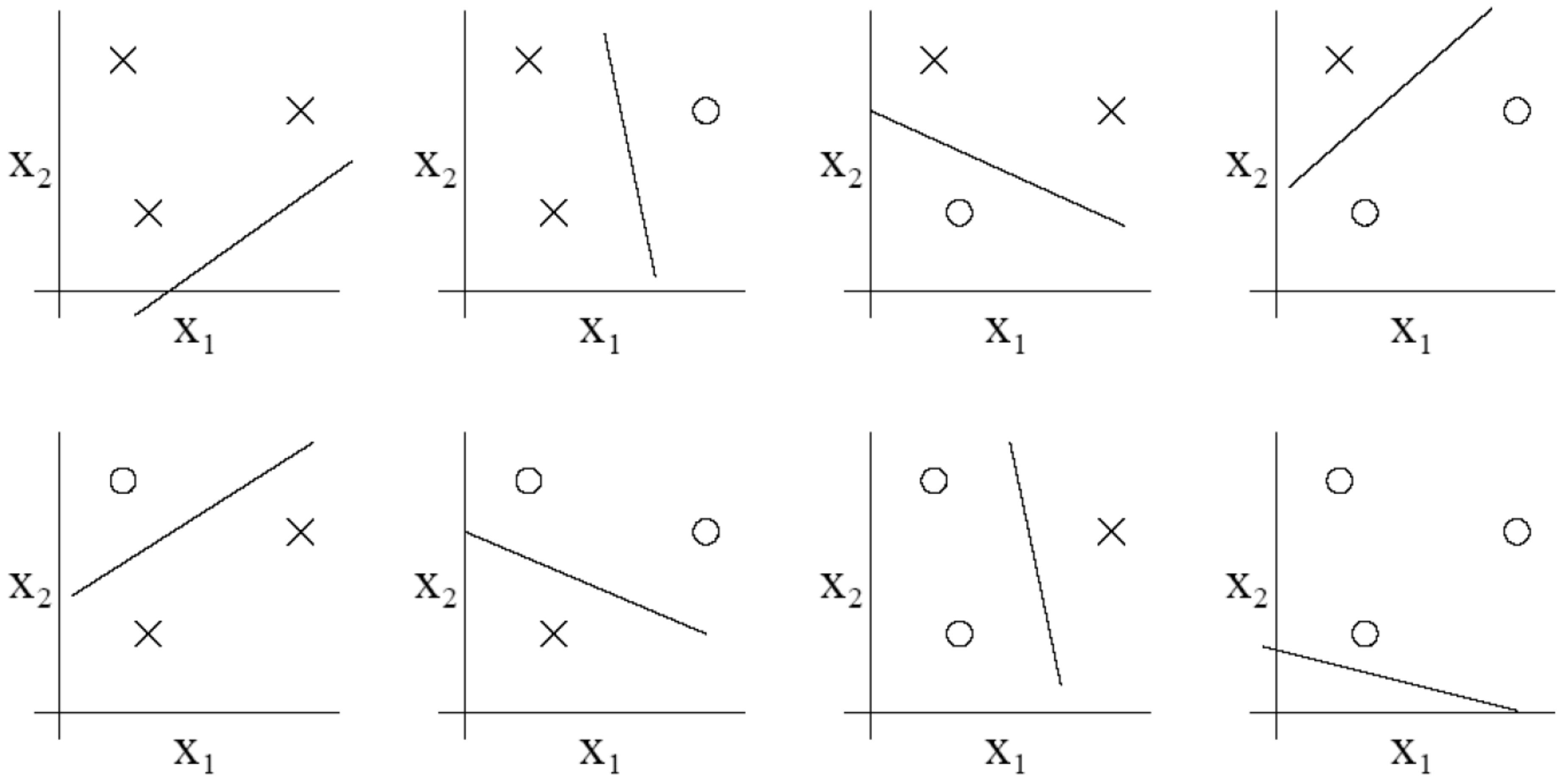
Vapnik Chervonenkis(VC) Dimension



- The maximum number of points that can be shattered by H is called Vapnik Chervonenkis Dimension

- In the figure VC(H) where H is the hypothesis class of axis angled rectangle in two dimension is four

Vapnik Chervonenkis(VC) Dimension



- VC(H)=3
- × = 0
- o = 1

Source:https://www.cs.cmu.edu/~epxing/Class/10701/slides/lecture16-VC.pdf

Dimensions of a supervised Machine Learning

- Consider $\mathbf{X} = \{ x^t, r^t \}_{t=1}^{t=N}$
- $x^t = \textit{It is the arbitrary dimensional input}$
- $r^t = \textit{Associated desired output}$ it may be 0 or 1 for two class learning
- In order to build a good and useful approximation to r^t using the model $(g(x^t|\theta)$ we should make following three decision

1. Model, we in learning is denoted as g(x|θ) where g(.) is the model x= Arbitrary-dimensional input
   θ= θ are the parameters
   g(.) defines the hypothesis class H
   A particular value of θ instantiates one hypothesis h $h \in H$. For example in Class learning , we have taken rectangle as our model whose four coordinates make up θ . In linear regression , model is the linear function of the input, whose slope and intercept are the parameters learned from the data.
   $g(x|m,c) = y = 2x + 5$ in this m=slope=2 and intercept=5 learned from the data.

   The model (inductive bias) or H is fixed by the machine learning system designer based on the knowledge of application and the hypothesis is tuned by a learning algorithm using the training set sampled from p(x,r) . All instances are drawn from the joint distribution p(x,r)

MACHINE LEARNING Unit –2

Dimensions of a supervised Machine Learning

2. Loss function, L(.) to compute the difference between the desired set r^t and our approximation to it $g(x^t|\theta)$, given the current value of the parameter θ. The approximation error or is the sum of the losses over the individual instances

$E(\theta|X) = \sum_{t=1}^{N} L(r^t, g(x^t|\theta))$ In class learning L(.) checks for equality or not. Then it outputs 0/1. In regression we have ordering information for the distance and possibility is to use the square of the difference

3. Optimization procedure to find θ^* that minimizes the total error

$\theta^* = arg\ min_{\theta} E(\theta|X)$ where arg min returns the argument that minimizes. In polynomial regression, we can solve analytically for the optimum, but this is not always the case. With other models and error functions, the complexity of the optimization problem becomes important. We are especially interested in whether it has a single minimum corresponding to globally optimal solution, or whether there are multiple minima corresponding to locally optimal solution.

For this to work well

(1)Firstly, the hypothesis class of g(.) should large enough, that is , it should have enough capacity to include the unknown function generated the data X in noisy form.

(2) There should be enough training data to allow us to pinpoint the correct hypothesis from the hypothesis class

(3) Third, we should have a good optimization method that finds the correct hypothesis given the training data

MACHINE LEARNING Unit –2

Parametric Classification

- Let the posterior probability of class C_i is

$$P(C_i|x) = \frac{p(x|C_i)P(C_i)}{p(x)} = \frac{p(x|C_i)P(C_i)}{\sum_{k=1}^{K} p(x|C_k)P(C_k)}$$

- The discriminant function is given by $g_i(x) = p(x|C_i)P(C_i)$
- Taking log $g_i(x) = \log p(x|C_i) + \log P(C_i)$
- We can assume $p(x|C_i)$ as Gaussian $p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right]$
- Substituting $p(x|C_i)$ in $g_i(x)$ $g_i(x) = -\frac{1}{2}\log 2\pi - \log \sigma_i - \frac{(x-\mu_i)^2}{2\sigma_i^2} + \log P(C_i)$

- For example , Assume car company selling K-different cars

- x=customer’s choice that affects the income
- $P(C_i)$ = Proportion of customers who buy car of type i.

Parametric Classification

❖ If the yearly income distributions of such customers can be approximated with a Gaussian
Then $p(x|C_i)$ is the probability that customer who bought car type i has annual income x has normal distribution with mean $\text{mean } \mu_i$ and variance σ_i^2 where μ_i= mean of the annual income of the customers
σ_i^2= Variance of the annual income of the customers.

❖ When we do not know $P(C_i)$ *and* $p(x|C_i)$ we estimate as follows

$$\mathcal{X} = \{x^t, \boldsymbol{r}^t\}_{t=1}^N$$

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N}$$

$$r_i^t = \begin{cases} 1 & \text{if } \boldsymbol{x}^t \in C_i \\ 0 & \text{if } \boldsymbol{x}^t \in C_k, k \neq i \end{cases}$$

$$m_i = \frac{\sum_t x^t r_i^t}{\sum_t r_i^t}$$

$$s_i^2 = \frac{\sum_t (x^t - m_i)^2 r_i^t}{\sum_t r_i^t}$$

MACHINE LEARNING Unit –2

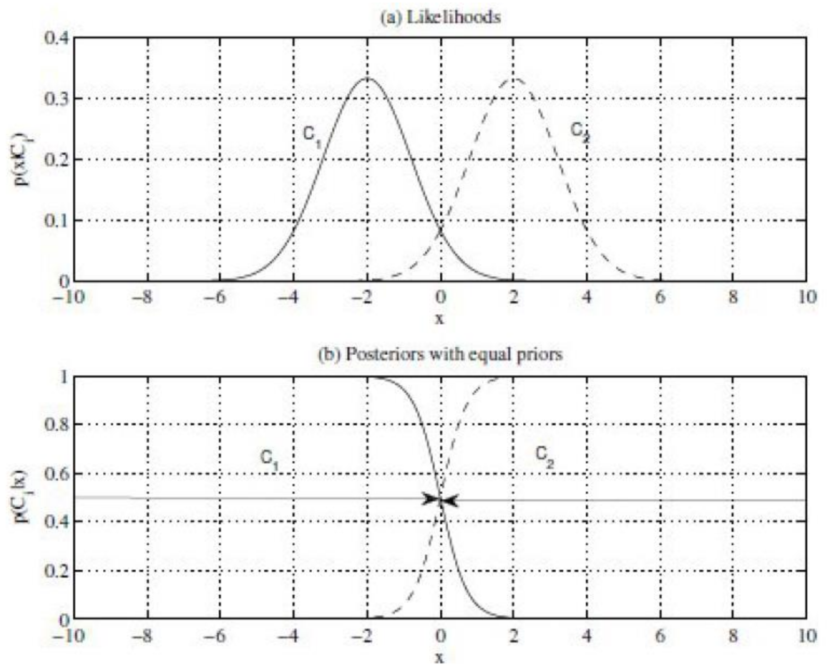
Parametric Classification

- $$g_i(x) = -\frac{1}{2}\log 2\pi - \log s_i - \frac{(x-m_i)^2}{2s_i^2} + \log \hat{P}(C_i)$$
- The first term is a constant and it can be neglected . If the priors are equal, the last term can also be neglected. If Variances are equal then we can write
- $$g_i(x) = -(x-m_i)^2$$
- The sample x can be assigned to the class with the nearest mean
- $$\text{Choose } C_i \text{ if } |x-m_i| = \min_k |x-m_k|$$
- With two adjacent classes $g_1(x) = g_2(x)$

$$\begin{aligned} (x-m_1)^2 &= (x-m_2)^2 \\ x &= \frac{m_1+m_2}{2} \end{aligned}$$

MACHINE LEARNING Unit –2

Parametric Classification

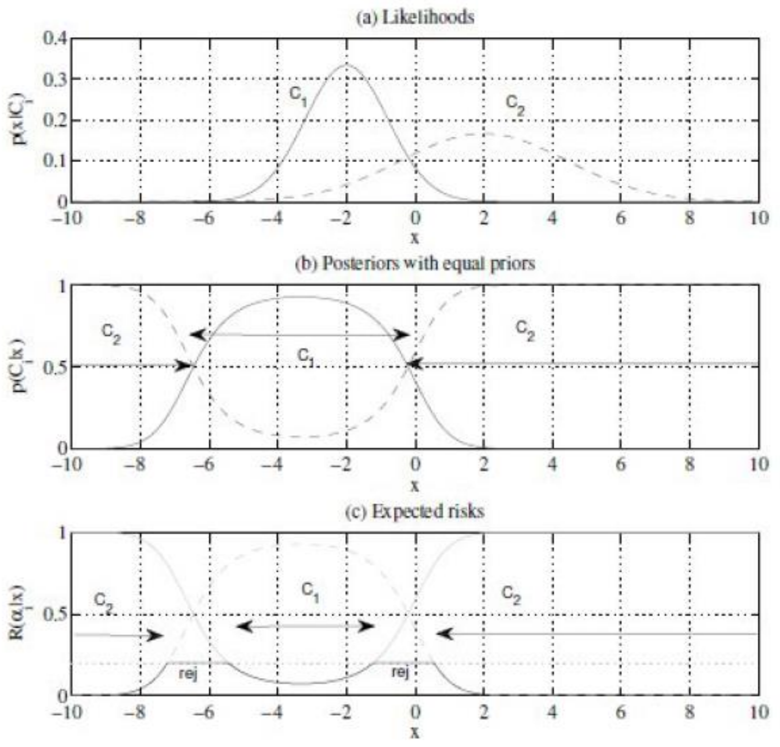


- Likelihood functions and posteriors with equal priors for the two classes when the input is one dimensional. Variances are equal and the posteriors intersect at one point, which is the threshold decision

MACHINE LEARNING Unit –2

Parametric Classification

❖

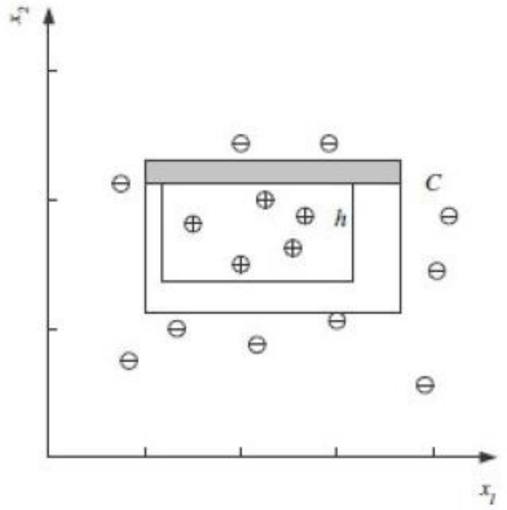


❖ **Likelihood functions and posteriors with equal priors for the two classes when the input is one dimensional. Variances are unequal and the posteriors intersect at two point, which is the threshold decision. The expected risks are shown for two classes and reject with λ =0.2**

❖ **When variances are different there are two thresholds.**

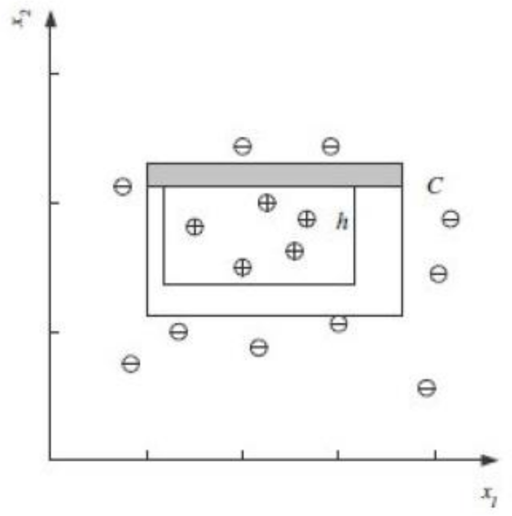
Probably Approximately Correct learning

- Class C , We want to find the examples drawn from some unknown but fixed probability distributions p(x).
- We want to find the number of examples , N, such that with probability (1-δ) , the hypothesis h has error at most ϵ
- $P\{C\Delta h \leq \epsilon\} = 1 - \delta$ where $c\Delta h$ = Region of difference between C and h,
- h=S is the tightest possible rectangle , the error region between C and h=S is the sum of four rectangular strips.



Probably Approximately Correct learning

- Probability that each positive example falling in the strip $= \epsilon/4$
- Probability that each example misses the strip= $1 - \epsilon/4$
- The probability that N-independent draws misses one of the strip $=(1-\frac{\epsilon}{4})^N$
- The probability that N-independent draws misses four strips $=4(1-\frac{\epsilon}{4})^N$



MACHINE LEARNING Unit –1

Probably Approximately Correct learning

- $4(1-\frac{\epsilon}{4})^N = 4(e^{\frac{-\epsilon}{4}})^N$
- $4(e^{\frac{-\epsilon}{4}})^N \leq \delta$
- $(e^{\frac{-\epsilon}{4}})^N \leq \delta/4$
- Taking natural logarithm on both sides
- $ln\left(\frac{4}{\delta}\right) \leq ln\left(e^{\frac{\epsilon}{4}}\right)^N$
- $N \geq (\frac{4}{\epsilon})\ ln\left(\frac{4}{\delta}\right)$
- Provided we take $N \geq (\frac{4}{\epsilon})\ ln\left(\frac{4}{\delta}\right)$ examples and use the tightest rectangle as our hypothesis h, with confidence probability (1-δ), a given point or sample will be misclassified with error probability ϵ

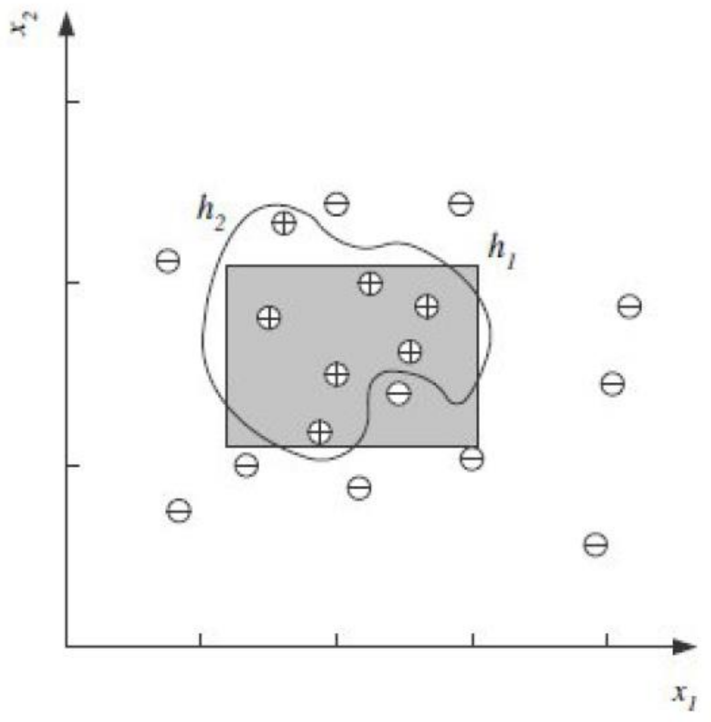
Probably Approximately Correct learning

- As δ decreases the (1-δ) confidence probability increases
- As ϵ decreases confidence probability increases
- We know $N \geq (\frac{4}{\epsilon})\, ln\left(\frac{4}{\delta}\right)$ As δ decreases N increases

MACHINE LEARNING Unit –1

Noise

- Noise is an unwanted anomaly in the data and due to noise , the class may be difficult to learn and zero error may be impossible with a simple hypothesis class

- 

Noise

- There may be imprecision in recording the input attributes, which may shift the data points in the input space.

- There may be errors in labelling the data points , which may relabel positive example as negative example and vice versa. This is sometimes known as teacher's noise.

- There may be additional attributes, which we have not taken into account, that affect the label of an instance. Such attributes may be hidden or latent in that they may be unobservable. The effect of these neglected attributes is modelled as a random component and it is included in the noise.

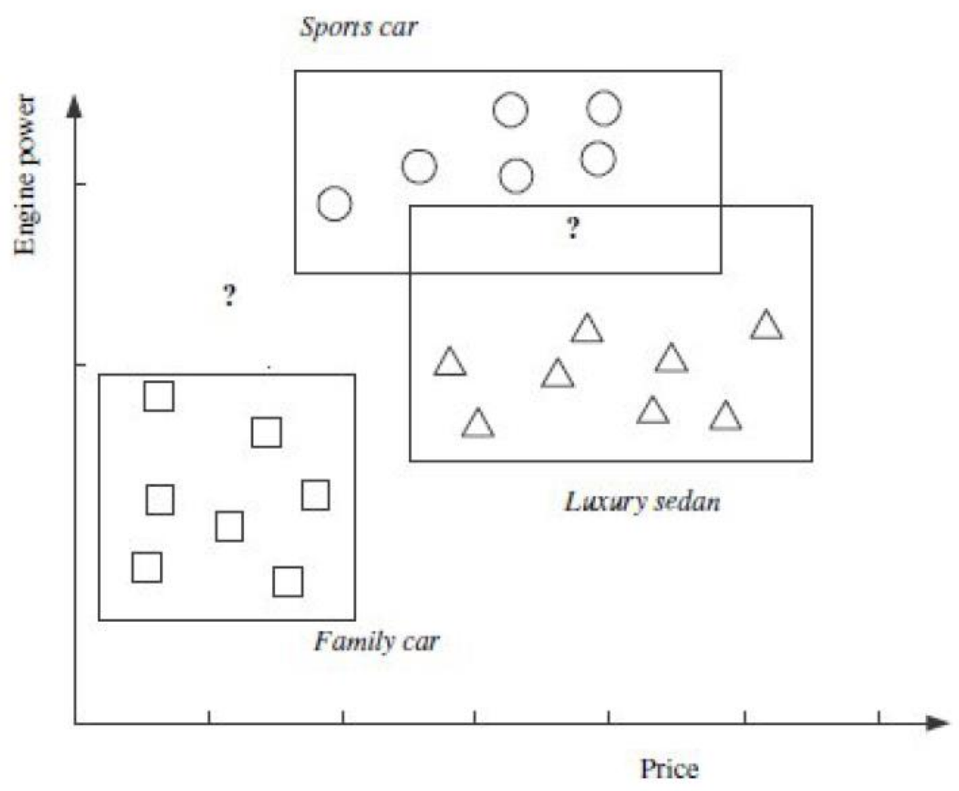
Noise

- When there is noise we need complicated hypothesis to separate positive and negative instances.

- Using the simple rectangle makes sense because of the following

1. It is simple model to use. It is easy to check whether a point is inside or outside a rectangle and we can easily check , for a future data instance , whether it is positive or a negative instance
2. It is a simple model to train and has fewer parameters. It is easier to find the corner values of a rectangle than the control points of an arbitrary shape. A simple model may have less variance. A simple model has more bias. Finding the optimal model corresponds to minimizing both the bias and variance
3. It is a simple model to explain. A rectangle simply corresponds to defining intervals on two attributes. By learning a simple model , we can extract information from the raw data given in the training set.
4. If indeed there is mislabelling or noise in input and the actual class is really a simple model like rectangle , Given comparable empirical error we say that a simple model would generalize better than a complex model. This principle is known as Occam's razor.

Learning Multiple Classes



- There are K-Classes denoted by $C_i \ for \ i = 1, 2, \dots, K$.
- For example Class $C_1 = Family\ Car, \quad Class\ C_2 = Luxury\ Car, Class\ C_3 = Sports\ car$

Learning Multiple Classes

- The training set consists of $\mathbf{X} = \{ x^t, r^t\}_{t=1}^{t=N}$
- x and r has K- dimension, and r has
- $r_i^t = \begin{cases} 1 \quad if\ x^t \in C_i \\ 0 \quad if\ x^t \in C_j for\ j \neq i \end{cases}$
- In Machine learning for classification , we would like to learn the boundary separating the instances of one class from the instances of all other classes
- The training examples belonging to C_i are the positive instances of hypothesis h_i and all other classes are negative instances of hypothesis h_i
- We have K- hypothesis to learn such that
- $h_i(x^t) = \begin{cases} 1 \quad if\ x^t \in C_i \\ 0 \quad if\ x^t \in C_j for\ j \neq i \end{cases}$
- The total empirical error takes a sum over the predictions for all the classes over all instances
- $E\left(h_{i\,i=1}^{\;K} \middle| X\right) = \sum_{t=1}^{N} \sum_{i=1}^{K} 1(h_i(x^t) \neq r_i^t)$ where K= No of Classes N = No. of instances

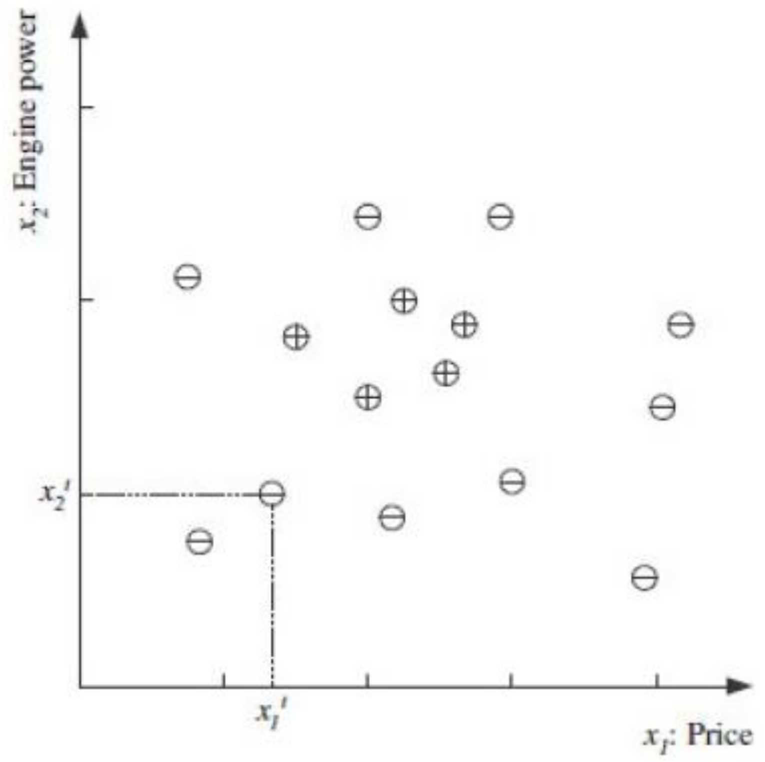
MACHINE LEARNING Unit –1

Learning Multiple Classes

- The total empirical error takes a sum over the predictions for all the classes over all instances
- $E\left({h_i}_{i=1}^{i=K} \middle| X\right) = \sum_{t=1}^{N} \sum_{i=1}^{K} \mathbf{1}(h_i(x^t) \neq r_i^t)$
- K= K-Classes, N= N No. of Instances $h_i = Hypothesis$ X= Training Set $x^t = Instance$
- For a given x ideally only one of $h_i(x), i = 1, 2, .. K \; is \; 1$ and we can choose a class
- We can not choose a class when two or more $h_i(x) = 1$
- We can not choose a class when no $h_i(x) = 1$. This is the case of doubt and classifier rejects such classes

MACHINE LEARNING Unit –1

Learning Class From Examples



C= Class C of a family car

- + denotes a positive example of a family car. It is with in the range of Price and Engine Power
- - denotes a negative example of a family car . It is outside the range of Price and Engine Power
- Each point corresponds to one example car

- Training data (x_1^t, x_2^t) is a point and it may be positive or negative and it is given by r^t

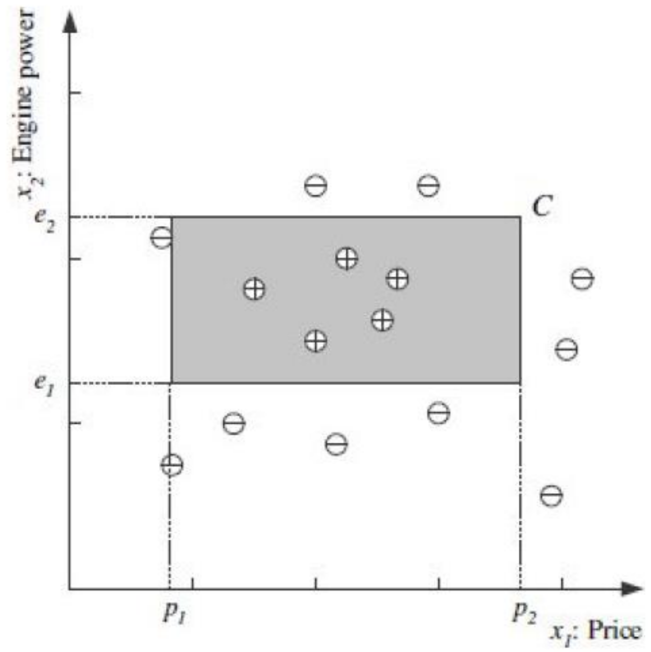
Learning Class From Examples

- Each car is represented by $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $x_1 = Price\ and\ x_2 = Engine\ Power$
- Each Car is labelled as follows

$$r = \begin{cases} 1\ if\ x\ is + ve\ example \\ 0\ if\ x\ is\ - ve\ example \end{cases}$$

- Each car is represented by pair (x , r)
- Training Set Contains N – Such examples
- $X = \{x^t, r^t\}_{t=1}^{t=N}$

Learning Class From Examples



- Training data (x_1^t, x_2^t) is a point and it may be positive or negative and it is given by r^t

- **Hypothesis Class**
- p_1 = *Lower limit of the price*
- p_2 = *Higher limit of the price*
- e_1 = *Lower limit of the engine power*
- e_2 = *Upper limit of the engine power*
- Let H be the Hypothesis class such that
- $(p_1 \leq price \leq p_2) AND (e_1 \leq Engine\ Power \leq e_2)$

Learning Class From Examples

- The learning algorithm finds the particular hypothesis

  $h \in H$

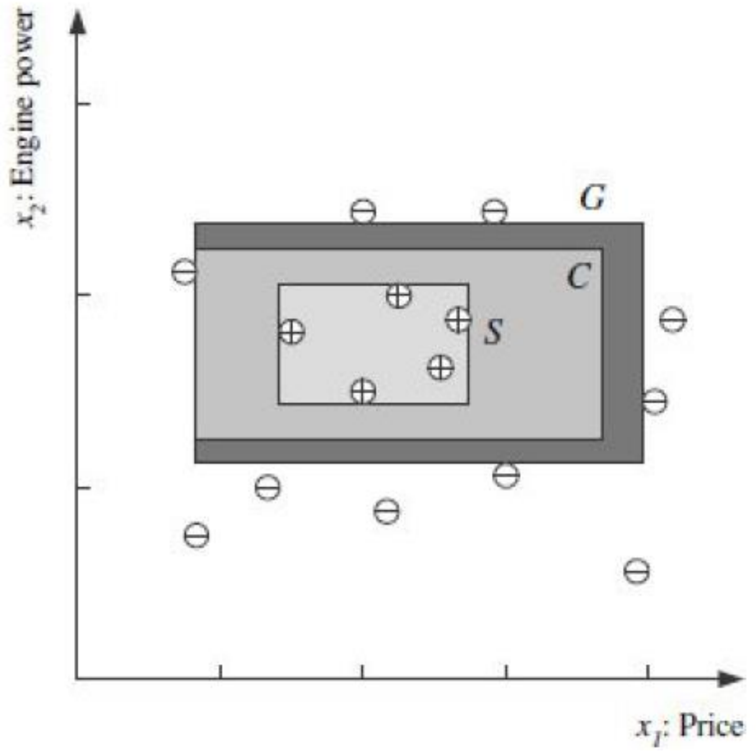
- It is specified by $(p_1^h, p_2^h, e_1^h, e_2^h)$
- $h \in H$ is equal to C or closet to C
- Aim is to find $h \in H$ is as similar as possible to C
- $$h(x) = \begin{cases} 1 \text{ } if \text{ } h \text{ } classifies \text{ } x \text{ } as + ve \text{ } example \\ 0 \text{ } if \text{ } h \text{ } classifies \text{ } x \text{ } as \text{ } - ve \text{ } example \end{cases}$$

Learning Class From Examples

- In real life C(x) is not known .We can not evaluate how well h(x) matches C(x).There fore we have to evaluate empirical error which is the proportion of training instances.
- The Error of hypothesis h is given the training set X is
- $E(h|X) = \sum_{t=1}^{t=N} 1(h(x^t) \neq r^t)$
- $1(h(x^t) \neq r^t) is\ 1\ if h(x^t) \neq r^t$
- $1(h(x^t) \neq r^t) is\ 0\ if h(x^t) = r^t$
- We have to select this h such that E=0

Learning Class From Examples

❖ Most Specific Hypothesis S, Most General Hypothesis G and Version Space

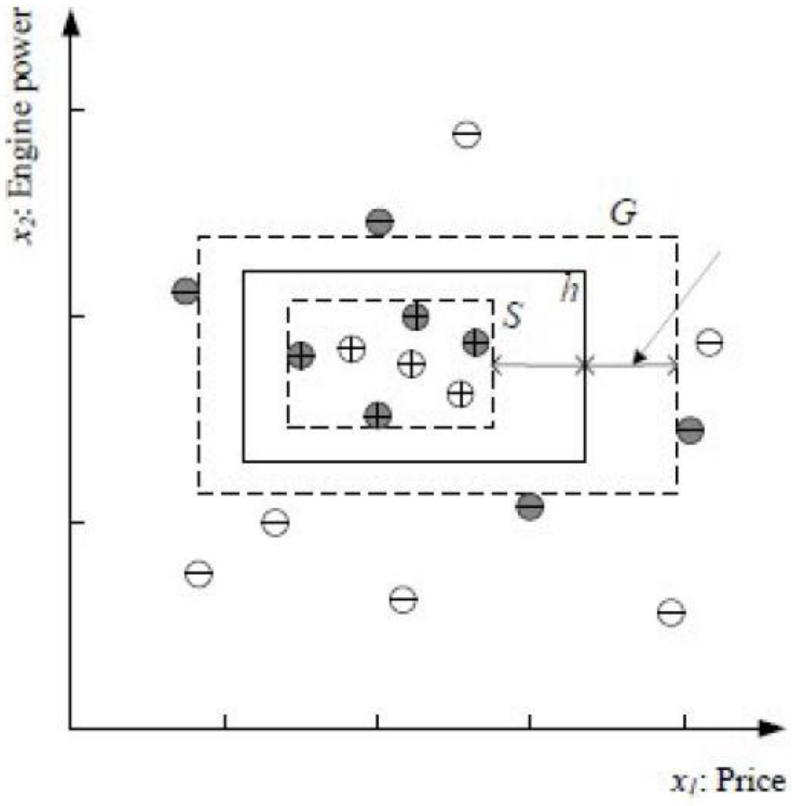


- ❖ S is the Most Specific Hypothesis
- ❖ G is the Most General Hypothesis

- ❖ Any $h \in H$ between S and G is a valid hypothesis with no error, said to be consistent with the training set and such h make up the Version space

Learning Class From Examples

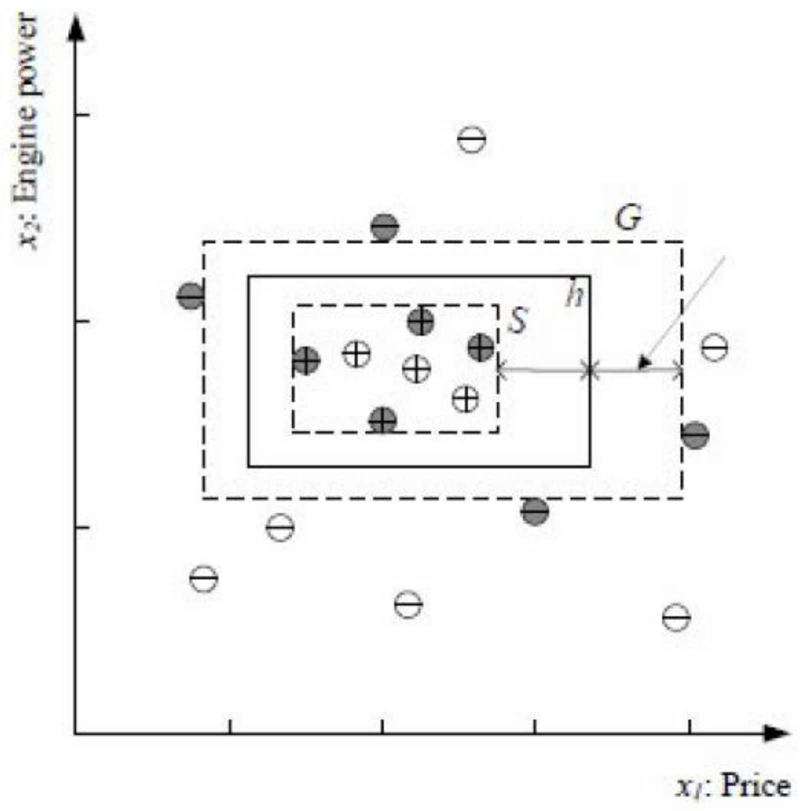
- **Doubt**



- **In some applications, there may be hypothesis that falls between S and G is a case of doubt which we cannot label due to lack of data.**

Learning Class From Examples

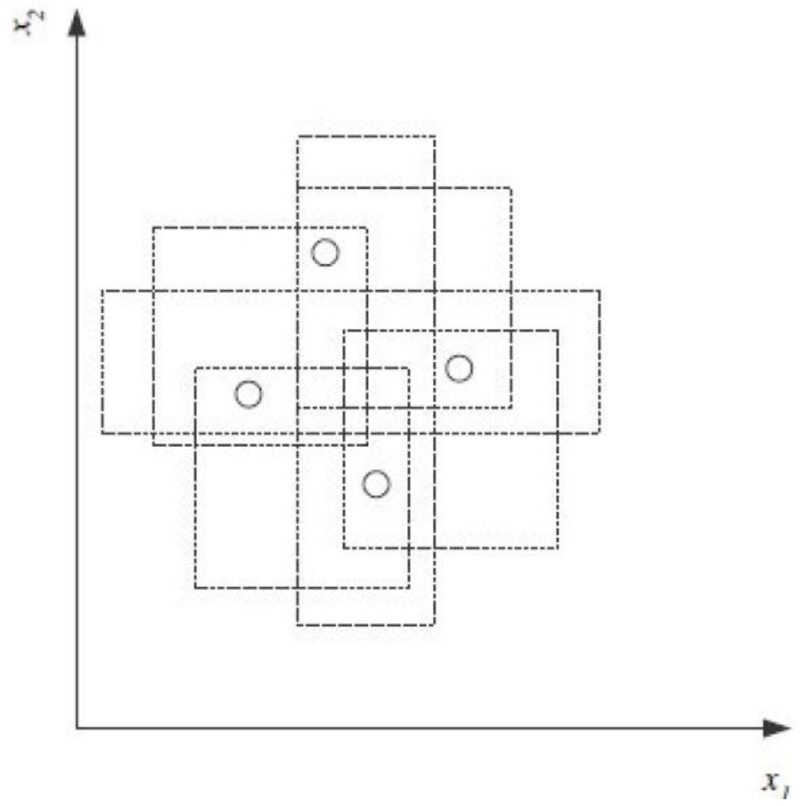
❖ **Margin**



❖ We chose the hypothesis with the largest margin, for best separation.

❖ The shaded define margin

Vapnik Chervonenkis(VC) Dimension



- The maximum number of points that can be shattered by H is called Vapnik Chervonenkis Dimension
- In the figure VC(H) where H is the hypothesis class of axis angled rectangle in two dimension is four

Regression

- In Classification output is Boolean , but in Regression output is a numeric function

- We have the training set $\mathbf{X} = \{ x^t, r^t\}_{t=1}^{t=N}$, we would like to find the function f(x) that passes x^t , there fore we have $r^t = f(x^t)$. If there is no noise , the task is interpolation. We would like to find the function f(x) that passes through these points.
- In polynomial interpolation , given N-Points , we find (N-1) degree polynomial that we used to predict the output for any input x. This is called extrapolation if x is outside the range of x^t in the training set.

- In time-series prediction , we have data up to the present we want to predict the value for future.
- In regression , there is noise added to the output of the unknown function

$r^t = f(x^t) + \epsilon$ where f(x) is an unknown function and ϵ is random noise.

- These noise occur due to extra hidden variables that we can not observe.

- $r^t = f^*(x^t, z^t)$ where z^t= Represents hidden variables

MACHINE LEARNING Unit –2

Regression

- The empirical error on the training set X is

- $E(g|X) = \frac{1}{N}\sum_{t=1}^{N}[r^t - g(x^t)]^2$ where $g(x^t) = \mathit{Output\ by\ our\ model}$ r^t = output from the proposed function

- Where r and g() Numeric quantities. Our aim is to find g(.) That minimizes the empirical error. There fore we have g(.) as

$$g(\boldsymbol{x}) = w_1x_1 + \cdots + w_dx_d + w_0 = \sum_{j-1}^{d} w_jx_j + w_0$$

- For single input linear model

- $g(x) = w_1x + w_0$ where w_1 *and* w_0 are parameters to learn. w_1 *and* w_0 values should minimize

$$E(w_1, w_0|X) = \frac{1}{N}\sum_{t-1}^{N}[r^t - (w_1x^t + w_0)]^2$$

- Its minimum point can be calculated by taking the partial derivatives of E with respect w_1 *and* w_0 and setting them equal to 0 and solving for the two unknowns

Regression

❖
$$w_1 = \frac{\sum_t x^t r^t - \overline{x}\,\overline{r}N}{\sum_t (x^t)^2 - N\overline{x}^2}$$
$$w_0 = \overline{r} - w_1\overline{x}$$

where $\overline{x} = \sum_t x^t/N$ and $\overline{r} = \sum_t r^t/N$.

❖ **If the linear model is too simple , it is too constrained and causes a large approximation error in such case , the output may be taken as the higher order function of the input**

$$g(x) = w_2x^2 + w_1x + w_0$$

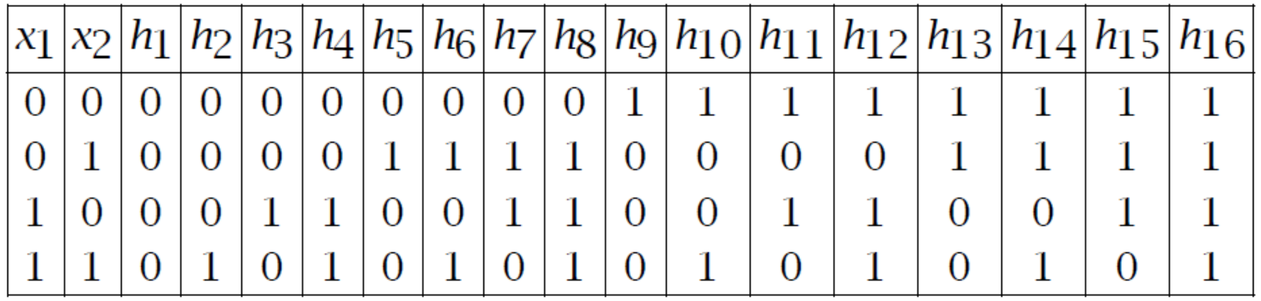
where w_{2}, w_1 *and* w_0 are parameters to learn

MACHINE LEARNING Unit –2

Model Selection and Generalization

- For a Boolean function.

- Number of inputs=d (x1,x2), Training instances = 2^d (00,01,10,11),the number of outputs =2^{2^d}
- h1,h2,h3,h4,h5……h16 are sixteen possible functions

- 

x_1	x_2	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}	h_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- Each distinct training example removes the hypothesis, namely whose guesses are wrong

- For example , $x_1 = 0, x_2 = 1$ ouput is zero . This removes h_5, h_6 , h_7, h_8 , h_{13}, h_{14} ,h_{15}, h_{16} .

- This is because $Output = f(x_1, x_2)$

Model Selection and Generalization

- We start with all possible hypotheses and as we see more training examples, we remove those hypothesis that are not consistent with the training data

- In case of a Boolean function to end up with a single hypothesis we need to examine 2^d *training examples*

- If the training set consists of small subset of all possible instances and if we know the output for a small percentage of the cases , then solution is not unique.

- After seeing N- example cases(N=3) ,if there remain 2^{2^d-N} (if d=2 and N=3 , 2^{2^d-N}=2) possible functions. This is an example of ill-posed problem. Where the data itself is not sufficient to find the unique solution.

- Since learning is ill-posed and data by itself is not sufficient to find the solution , we should make extra assumptions to have unique solution with the data we have.

- The set of assumptions we make to have learning possible is called inductive bias of the learning algorithm .

Model Selection and Generalization

- ❖ Since data itself is not sufficient to find the unique solution, we should make some extra assumptions to have a unique solution with the data we have. The set of assumptions we make to have learning possible is called inductive bias .

- ❖ For Example : (1) In learning the class of family cars , assuming the shape of the rectangle is one inductive bias and then the rectangle with largest margin is another inductive bias.

- ❖ (2) In linear regression assuming a linear function is an inductive bias.

Model Selection and Generalization

- Thus learning is not possible without inductive bias, and now the question is how to choose the right bias. This is called model selection , which is choosing between possible H.

- We would like to be able to generate the right output for an input instance outside the training set, one for which the correct output is not given in the training set.

- How well a model trained on trained on the training set predicts the right output for new instances is called generalization.

Model Selection and Generalization

- For best generalization , we should match the complexity of hypothesis class H with the complexity of underlying function.
- Underfitting
- If the hypothesis class H is less complex than the function. We have underfitting.
- Example: We are trying to fit a line to a data sampled from the third order polynomial
- In such a case, as we increase complexity , the training error decreases.

Model Selection and Generalization

- **Overfitting**
- When fitting a sixth order polynomial to noisy sampled data from a third order polynomial
- In such cases having more training data helps but only up to a certain point.
- **Triple Trade-off**
- The complexity of the hypothesis to fit data , namely the capacity of the hypothesis class.
- The amount of training data.
- The generalization error on new examples

Model Selection and Generalization

- **Generalization ability of a hypothesis**
- **Hypothesis is can tested on Validation Set. For this we can use Cross Validation**
- **Using Cross validation it is possible to evaluate the quality of inductive bias.**
- **In addition to training set and validation set, a third set which is known as test set or sometimes known as publication set may be used to test the algorithm.**

Regression

- In Regression , output variables are functions of the input called the independent variables.
- $r = f(x) + \epsilon$
- Where f(x)= It is the unknown function which is approximated by $g(x|\theta)$
- ϵ is Gaussian with zero mean and constant variance σ^2

$$p(r|x) \sim \mathcal{N}(g(x|\theta), \sigma^2)$$

$$p(x, r) = p(r|x)\,p(x)$$

Regression

- $p(r|x) = probability\ of\ the\ output\ given\ the\ input$
- $p(x) = input\ probability$ density

$$\mathcal{X} = \{x^t, r^t\}_{t=1}^N,$$

- Log likelihood

$$\begin{aligned}\mathcal{L}(\theta|\mathcal{X}) &= \log \prod_{t=1}^{N} p(x^t, r^t) \\ &= \log \prod_{t=1}^{N} p(r^t|x^t) + \log \prod_{t=1}^{N} p(x^t)\end{aligned}$$

- We can ignore the second term since it does not depends on estimator

MACHINE LEARNING Unit –1

Regression

$$\begin{aligned}
\mathcal{L}(\theta|\mathcal{X}) &= \log \prod_{t=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{[r^t - g(x^t|\theta)]^2}{2\sigma^2}\right] \\
&= \log \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left[-\frac{1}{2\sigma^2}\sum_{t=1}^{N}[r^t - g(x^t|\theta)]^2\right] \\
&= -N\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\sum_{t=1}^{N}[r^t - g(x^t|\theta)]^2
\end{aligned}$$

- **The first term is independent of θ and it can be neglected and also the factor**
- $1/\sigma^2$. **can be neglected. There fore we get**

$$E(\theta|\mathcal{X}) = \frac{1}{2}\sum_{t=1}^{N}[r^t - g(x^t|\theta)]^2$$

$\boldsymbol{E(\theta|x)}$ **is the most frequently used error function and θ that minimizes it are called Least Square Estimates**

Regression

- In linear regression, we have linear model
- $g(x^t|w_1, w_0) = w_1 x^t + w_0$
- Consider $E(\theta|X) = \frac{1}{2}\sum_{t=1}^{N}[r^t - g(x^t|\theta)]^2$
- Placing $g(x^t|w_1, w_0) = w_1 x^t + w_0$
- Taking the derivative of the sum of the squared errors with respect w_1 and w_0

$$\begin{aligned}\sum_t r^t &= N w_0 + w_1 \sum_t x^t \\ \sum_t r^t x^t &= w_0 \sum_t x_t + w_1 \sum_t (x^t)^2\end{aligned}$$

Regression

❖ It can be written in vector form $\mathbf{A}w = y$

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t \\ \sum_t x^t & \sum_t (x^t)^2 \end{bmatrix}, \; \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \; \boldsymbol{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \end{bmatrix}$$

$$w = \mathbf{A}^{-1} y.$$

Regression

- In case of polynomial regression, the model is a polynomial in x of order k
- $g(x^t|w_k, \ldots, w_2, w_1, w_0) = w_k(x^t)^k + \cdots + w_2(x^t)^2 + w_1 x^t + w_0$

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t & \sum_t (x^t)^2 & \cdots & \sum_t (x^t)^k \\ \sum_t x^t & \sum_t (x^t)^2 & \sum_t (x^t)^3 & \cdots & \sum_t (x^t)^{k+1} \\ \vdots & & & & \\ \sum_t (x^t)^k & \sum_t (x^t)^{k+1} & \sum_t (x^t)^{k+2} & \cdots & \sum_t (x^t)^{2k} \end{bmatrix}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \\ \sum_t r^t (x^t)^2 \\ \vdots \\ \sum_t r^t (x^t)^k \end{bmatrix}$$

Regression

- We can write $\mathbf{A} = \mathbf{D}^T\mathbf{D}$ and $\boldsymbol{y} = \mathbf{D}^T\boldsymbol{r}$ where
- $$\mathrm{D} = \begin{bmatrix} 1 & x^1 & (x^1)^2 & \cdots & (x^1)^k \\ 1 & x^2 & (x^2)^2 & \cdots & (x^2)^k \\ \vdots & & & & \\ 1 & x^N & (x^N)^2 & \cdots & (x^N)^k \end{bmatrix}, \boldsymbol{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

$$\boldsymbol{w} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\boldsymbol{r}$$

MACHINE LEARNING Unit –1

Regression

- <u>Relative Square Error</u>

  Another measure is called the Relative Square Error

$$E_{RSE} = \frac{\sum_t [r^t - g(x^t|\theta)]^2}{\sum_t (r^t - \overline{r})^2}$$

- If $E_{RSE} = 1$, Then our prediction is as good as predicting the average
- If $E_{RSE} = 0$, Then we have a better fit.
- A measure $R^2 = 1 - E_{RSE}$ lness by fit regression is the coefficient of regression
- For regression to be useful we require R^2 to be close to 1

Model Selection Procedure

- **Minimum Description Length:**
- **Kolmogrov complexity of a dataset is defined as the shortest description of the data.**
- **For example , if it is a sequence of ‘0’ s , we can just write 0 and length of sequence**
- **If the data is simple , it has a short complexity.**
- **If the data is completely random, we can not have any description of the data shorter that data itself. If a model is appropriate for the data , then it is a good fit to the data.**
- **Out of all the models that describe the data, we want to have simplest model, that lends itself the shortest description.**
- **There is a trade of between simplicity of the model and description .**

Model Selection Procedure

- **Bayesian Model Selection:**
- $$p(\text{model}|\text{data}) = \frac{p(\text{data}|\text{model})\,p(\text{model})}{p(\text{data})}$$
- $p(model|data) = Posterori\ probability\ of\ the\ given\ model$
- $p(model) = Prior\ \ subjective\ knowledge$
- Taking logarithm on both sides
- $$\log p(\text{model}|\text{data}) = \log p(\text{data}|\text{model}) + \log p(\text{model}) - c$$
- In this log likelihood of the data is the training error log(p(model)) is the penalty form
- In Case of regression we have
- $$E = \sum_t [r^t - g(x^t|\mathbf{w})]^2 + \lambda \sum_i w_i^2$$

Model Selection Procedure

- $$E = \sum_t [r^t - g(x^t|\mathbf{w})]^2 + \lambda \sum_i w_i^2$$
- We look for w_i that decreases error and also it will be close to zero.
- Whenever it is zero the fitted polynomial is smoother. As the polynomial order increases, function will go up and down.

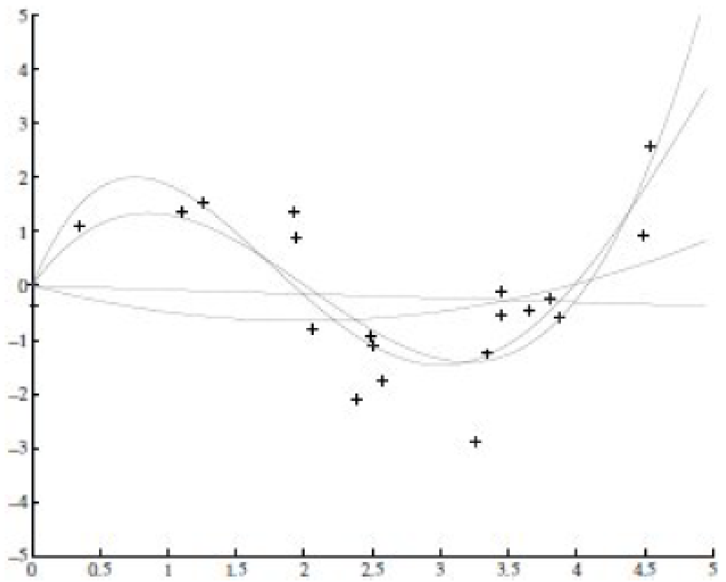


Figure 4.8 In the same setting as that of figure 4.5, polynomials of order 1 to 4 are fitted. The magnitude of coefficients increase as the order of the polynomial increases. They are as follows: 1 : $[-0.0769, 0.0016]^T$, 2 : $[0.1682, -0.6657, 0.0080]^T$, 3 : $[0.4238, -2.5778, 3.4675, -0.0002]^T$, 4 : $[-0.1093, 1.4356, -5.5007, 6.0454, -0.0019]^T$.

Multivariate Data

- Now we have to see the classification or regression in which we have multiple inputs.
- Output is a class code or continuous output is a function of multiple inputs. The inputs may be discrete or numeric
- The sample of multivariate data may be as follows

$$\mathbf{X} = \begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_d^1 \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & & & \\ X_1^N & X_2^N & \cdots & X_d^N \end{bmatrix}$$

- Where d-Columns represent d-variables, inputs , features or attributes
- N-rows corresponds to independent and identically distributed observations, examples or instances on N- individuals or events.
- Ex: For loan application assume there are N-Customers, Financial corporation may see their age, annual income, assets as features .

MACHINE LEARNING Unit –1

Parameter Estimation

- Consider

$$\mathbf{X} = \begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_d^1 \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & & & \\ X_1^N & X_2^N & \cdots & X_d^N \end{bmatrix}$$

- The mean vector μ may be defined such that each of its elements is the mean of one column of X and it is given by

$$E[\mathbf{x}] = \boldsymbol{\mu} = [\mu_1, \ldots, \mu_d]^T$$

- The variance of X_i is denoted as σ_i^2 and the covariance of X_i and X_j are defined as

$$\sigma_{ij} \equiv \mathrm{Cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] = E[X_i X_j] - \mu_i \mu_j$$

- With i=j , $\sigma_{ij} = \sigma_{ji} = \sigma_{ii} = \sigma_i^2$

Parameter Estimation

- With d Variables , there are d-variances and d(d-1) /2 covariances which are generally represented by dxd matrix, it is known as covariance matrix denoted as Σ, whose (i, j)th element is σ_{ij}

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

- The diagonal elements are variances and the off-diagonal elements are covariances variances and the matrix is symmetric. In matrix notation

$$\Sigma \equiv \mathrm{Cov}(\boldsymbol{X}) = E[(\boldsymbol{X}-\boldsymbol{\mu})(\boldsymbol{X}-\boldsymbol{\mu})^T] = E[\boldsymbol{X}\boldsymbol{X}^T] - \boldsymbol{\mu}\boldsymbol{\mu}^T$$

- The correlation between the variable $X_i \; and X_j$ is normalized between +1 and -1
- With i=j , $\sigma_{ij} = \sigma_{ji} = \sigma_{ii} = \sigma_i^2$

$$\mathrm{Corr}(X_i, X_j) \equiv \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

MACHINE LEARNING Unit –1

Parameter Estimation

- The sample mean is given by m

$$\boldsymbol{m} = \frac{\sum_{t=1}^{N} \boldsymbol{x}^t}{N} \text{ with } m_i = \frac{\sum_{t=1}^{N} x_i^t}{N}, i = 1, \dots, d$$

The estimator of $\boldsymbol{\Sigma}$ is $\mathbf{S}$, the ***sample covariance*** matrix, with entries

- The sample correlation coefficients are

$$r_{ij} = \frac{s_{ij}}{s_i s_j}$$

The sample correlation matrix R contains r_{ij}.

Estimation of missing values

- Frequently, values of certain variables may be missing observations. We try to fill these variables by estimating them. This is called imputation.

- Mean imputation : For a numeric variable , we substitute the mean of the available data for that variable in the sample.
- For a discrete variable , we fill with most likely value

- Imputation by Regression : We try to predict the value of a missing variable from other variables whose values are known for that case. If many different variables are missing, we take the means as the initial estimates and the procedure is iterated until predicted value stabilizes. If the variables are not highly correlated, the regression approach is equivalent to mean imputation.

Multivariate Normal distribution

- In Multivariate case x is d-dimensional and normal distributed
- x follows normal distribution with mean vector $\boldsymbol{\mu}$ and Σ is the covariance matrix $p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$
- In the multivariate case Mahalanobis distance is used

$$(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$$

- $(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = c^2$ is the d-dimensional hyper ellipsoid

  centered at $\boldsymbol{\mu}$, and its shape and orientation are defined by Σ.

- Because of the use of inverse of Σ. If a variable has a larger variance , it receives less weight in the Mahalanobis distance.

Multivariate Normal distribution

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(z_1^2 - 2\rho z_1 z_2 + z_2^2\right)\right]$$

$$z_i = (x_i - \mu_i)/\sigma_i,\ i = 1, 2,$$

Gradient Descent

- In likelihood – based classification, the parameters were sufficient statistics of $P(x|C_i)$ and $P(C_i)$ and the method we used to estimate the parameter is maximum likelihood estimation

- In discriminant based approach , the parameters are those of discriminants and they are optimized to minimize the classification error on the training set.

- In many cases , there is no analytical methods and we need to resort to iterative optimization methods, the most commonly used is the Gradient descent.

- When E(w) is differential function of a vector of variables, we have gradient vector composed of the partial derivatives

MACHINE LEARNING Unit –2

- $\nabla_w E = [\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_3}, \frac{\partial E}{\partial w_4}, \ldots\ldots \frac{\partial E}{\partial w_d}]^T$
- The gradient descent procedure is to minimize
- E starts from a random w and each step updates w, in the opposite direction of the gradient.
- $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$ where η = step size or learning factor and determines how much to move in that direction.
- Gradient ascent is used to maximize a function and goes in the direction of gradient.
- Gradient descent is used to minimize a function and goes in the opposite direction of gradient.
- When the Gradient is minimum , the derivative is zero.
- This indicates that the procedure finds the nearest minimum, that can be a local minimum and there is no guarantee of finding the global minimum unless the function has only one minium

MACHINE LEARNING Unit –2

Logistic Discrimination

- **Two Classes**
- **In logistic discrimination , we will make use of ratio of conditional densities** $p(\mathbf{x}|C_i)$
- **Consider** $\log \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} = \mathbf{w}^T\mathbf{x} + w_0^o$
- $$\begin{aligned}\text{logit}(P(C_1|\mathbf{x})) &= \log \frac{P(C_1|\mathbf{x})}{1 - P(C_1|\mathbf{x})} \\ &= \log \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} + \log \frac{P(C_1)}{P(C_2)} \\ &= \mathbf{w}^T\mathbf{x} + w_0\end{aligned}$$
- **Where** $w_0 = w_0^o + \log \frac{P(C_1)}{P(C_2)}$
- **Rearranging the terms we get** $y = \hat{P}(C_1|\mathbf{x}) = \frac{1}{1 + \exp[-(\mathbf{w}^T\mathbf{x} + w_0)]}$ **as our estimator of** $P(C_1|x)$

MACHINE LEARNING Unit –2

- We are given a sample of two classes $\mathcal{X} = \{\mathbf{x}^t, r^t\}$
- Where $r^t = 1$ if $\mathbf{x} \in C_1$ and $r^t = 0$ if $\mathbf{x} \in C_2$
- Where we assume r^t, given $\mathbf{x}^t$, is Bernoulli with probability $y^t \equiv P(C_1|\mathbf{x}^t)$
- The sample likelihood is $l(\mathbf{w}, w_0|\mathcal{X}) = \prod_t (y^t)^{(r^t)}(1-y^t)^{(1-r^t)}$
- $E = \log l$ $\quad E(\mathbf{w}, w_0|\mathcal{X}) = -\sum_t r^t \log y^t + (1-r^t)\log(1-y^t)$
- $y = \text{sigmoid}(a) = 1/(1+\exp(-a)), \qquad \frac{dy}{da} = y(1-y)$
-

$$\begin{aligned}
\Delta w_j &= -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1-r^t}{1-y^t}\right) y^t(1-y^t)x_j^t \\
&= \eta \sum_t (r^t - y^t)x_j^t, j = 1, \dots, d \\
\Delta w_0 &= -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t)
\end{aligned}$$

- Generally before training it is a good idea to initialize all inputs to have mean zero and unit variance by z-normalization.

- For this , for input j=1,2…d we calculate average m_j and standard deviation s_j and calculate

$$x_j^t = \frac{\tilde{x}_j^t - m_j}{s_j}$$

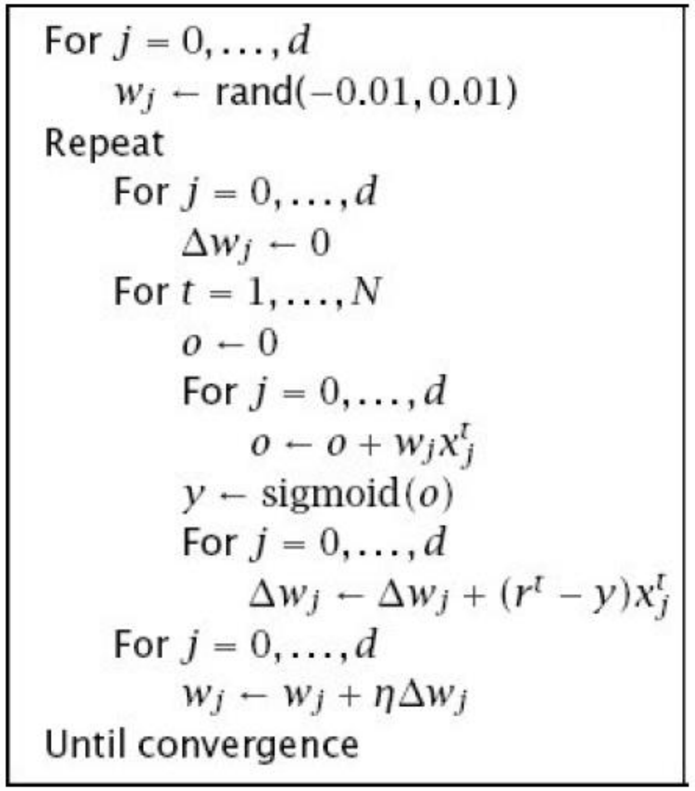
- $\tilde{x}^t$ are original values and x^t are values after normalization that we use in training. Note that same m_j and s_j values calculated on the training set are used to normalize the test instances. After normalization, all inputs are centered around zero and all have the same scale.

- Once the training is complete and we have the final w and w_0
- During testing we choose $C_1 \ if \ y^t > 0.5 \ and \ Choose \ C_2 \ otherwise$
- Stopping early before we have 0 training error is a form of regularization.
- We start with weights almost 0 and they move away as training continues ,

stopping early corresponds to a model with more weights close to 0.

MACHINE LEARNING Unit –2

❖ **Logistic discrimination algorithm implementing Gradient descent for the single output case with two classes. For w_0, we assume that there is an extra input x_0 which is always +1 .**
$x_0^t = +1 \; for \; all \; t$



```
For $j = 0, \ldots, d$
    $w_j \leftarrow$ rand$(-0.01, 0.01)$
Repeat
    For $j = 0, \ldots, d$
        $\Delta w_j \leftarrow 0$
    For $t = 1, \ldots, N$
        $o \leftarrow 0$
        For $j = 0, \ldots, d$
            $o \leftarrow o + w_j x_j^t$
        $y \leftarrow \mathrm{sigmoid}(o)$
        For $j = 0, \ldots, d$
            $\Delta w_j \leftarrow \Delta w_j + (r^t - y) x_j^t$
    For $j = 0, \ldots, d$
        $w_j \leftarrow w_j + \eta \Delta w_j$
Until convergence
```

Multiple Classes

- To generalize to K > 2 Classes. We take one of the classes C_K as the reference class.
- $$\log \frac{p(\mathbf{x}|C_i)}{p(\mathbf{x}|C_K)} = \mathbf{w}_i^T\mathbf{x} + w_{i0}^o$$
- $$\frac{P(C_i|\mathbf{x})}{P(C_K|\mathbf{x})} = \exp[\mathbf{w}_i^T\mathbf{x} + w_{i0}]$$
- Where $w_{i0} = w_{i0}^o + \log P(C_i)/P(C_K)$.
- $$\sum_{i=1}^{K-1} \frac{P(C_i|\mathbf{x})}{P(C_K|\mathbf{x})} = \frac{1 - P(C_K|\mathbf{x})}{P(C_K|\mathbf{x})} = \sum_{i=1}^{K-1} \exp[\mathbf{w}_i^T\mathbf{x} + w_{i0}]$$
- $$P(C_K|\mathbf{x}) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp[\mathbf{w}_i^T\mathbf{x} + w_{i0}]}$$

$$\frac{P(C_i|\mathbf{x})}{P(C_K|\mathbf{x})} = \exp[\mathbf{w}_i^T\mathbf{x} + w_{i0}]$$

MACHINE LEARNING Unit –2

- $$P(C_i|\mathbf{x}) = \frac{\exp[\mathbf{w}_i^T\mathbf{x} + w_{i0}]}{1 + \sum_{j=1}^{K-1}\exp[\mathbf{w}_j^T\mathbf{x} + w_{j0}]},\ i = 1,\ldots,K-1$$
- To treat all classes uniformly , we can write
- $$y_i = \hat{P}(C_i|\mathbf{x}) = \frac{\exp[\mathbf{w}_i^T\mathbf{x} + w_{i0}]}{\sum_{j=1}^{K}\exp[\mathbf{w}_j^T\mathbf{x} + w_{j0}]},\ i = 1,\ldots,K$$

Which is called the Softmax function.

Tuning Model Complexity : Bias / Variance Dilemma

- Consider a sample $X = \{x^t, r^t\}$
- $x^t = \textit{It is the arbitrary dimensional input}$
- $r^t = \textit{Associated desired output}$ it may be 0 or 1 for two class learning
- Let g(.) be our estimate. The expected square error at x can be written as

$$E[(r-g(x))^2|x] = \underbrace{E[(r-E[r|x])^2|x]}_{noise} + \underbrace{(E[r|x]-g(x))^2}_{squared\ error}$$

- The First term is the variance of the noise added, σ^2 .
- The second term quantifies how much g(x) deviates from regression function $E[r|x]$. It depends on the estimator and the training set .
- For one example x , g(x) may be very good fit and for some other example it may be bad fit. To quantify how well an estimator g(.) , we average over possible datasets.

Tuning Model Complexity : Bias / Variance Dilemma

- For one example x , g(x) may be very good fit and for some other example it may be bad fit. To quantify how well an estimator g(.) , we average over possible datasets. The expected value is given by

$$E_X[(E[r|x]-g(x))^2|x] = \underbrace{(E[r|x] - E_X[g(x)])^2}_{bias} + \underbrace{E_X[(g(x) - E_X[g(x)])^2]}_{variance}$$

- The bias measures how much g(x) is wrong disregarding the effect of varying samples and variance measures how much g(x) fluctuates around the expected value E[g(x)]. As sample varies we want both bias and variance to be small.

- Then $E_X[g(x)]$ is estimated by the average of $g_i(x)$

$$\overline{g}(x) = \frac{1}{M}\sum_{i-1}^{M} g_i(x)$$

Tuning Model Complexity : Bias / Variance Dilemma

- As the order of the polynomial increases small changes in the dataset cause a great change in fitted polynomials, thus variance increases. This is called bias/variance dilemma and is true for any machine learning system.
- To decrease the bias , model should be flexible , at the risk of having high variance.
- If the variance is kept low, we may not be able to make a good fit to the data and have high bias.
- The optimal model is one that has best trade-off between the bias and variance.

Tuning Model Complexity : Bias / Variance Dilemma

- Under fitting :- If there is bias, it indicates that our model class does not contain the solution . This is underfitting.

- Overfitting :- If there is variance , the model class is too general and also learns the noise . This is called overfitting. If the estimate g(.) of the same hypothesis class with function f(.) . Where the order of f(.) and g(.) is same , then we have an unbiased estimator , and estimated bias decreases as the number of models increases. This shows the error – reducing effect of choosing the right model.

- When the variance is large, bias is low $\overline{g}(x)$ is a good estimator. So to get a small value of error, we can take a large number of high variance models and use their average as our estimator.

Model Selection procedure

1. Cross Validation
2. Regularization
3. Akaike's information criterion (AIC)and Bayesian Information Criterion(BIC)
4. Structural Risk Minimization
5. Minimum Description Length
6. Bayesian model selection

❖ Cross Validation: We use Cross validation to find the optimal model. Given a dataset we divide it into two parts namely Training set and validation set. Models of different complexities are trained on training dataset and their error is tested on the validation set. As the model complexity increases , training error decreases. However , the error on the validation set decreases up to a certain point of complexity, and then stops decreasing or does not decrease further significantly , or even increases if there is a noise in the data . This elbow corresponds to the optimal complexity level.

Model Selection procedure

❖

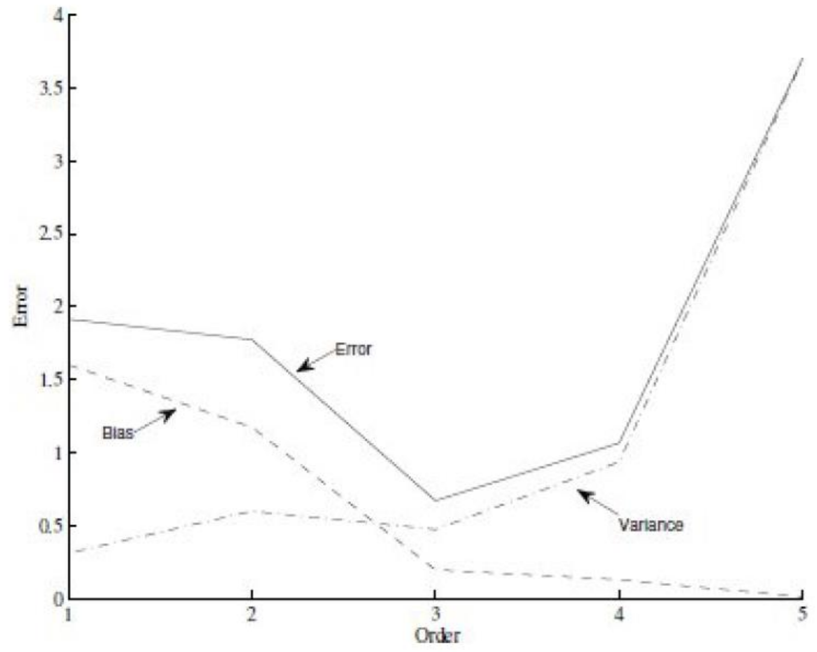


Figure 4.6 In the same setting as that of figure 4.5, using one hundred models instead of five, bias, variance, and error for polynomials of order 1 to 5. Order 1 has the smallest variance. Order 5 has the smallest bias. As the order is increased, bias decreases but variance increases. Order 3 has the minimum error.

Model Selection procedure

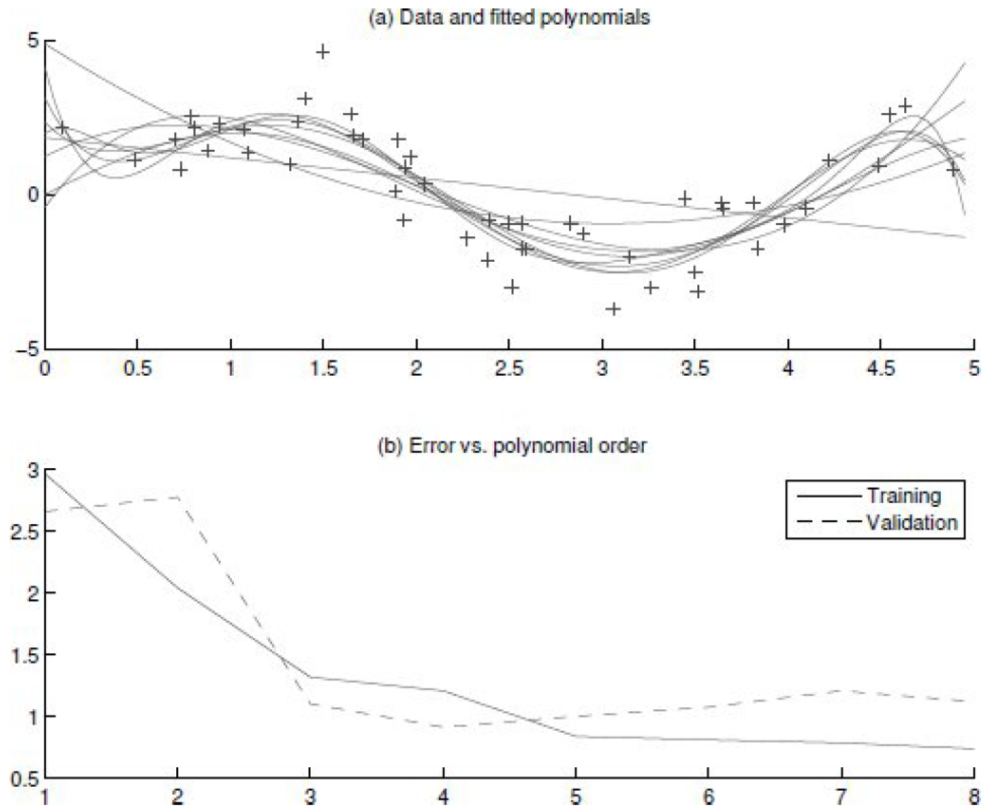


Figure 4.7 In the same setting as that of figure 4.5, training and validation sets (each containing 50 instances) are generated. (a) Training data and fitted polynomials of order from 1 to 8. (b) Training and validation errors as a function of the polynomial order. The "elbow" is at 3.

Model Selection procedure

- Regularization
- In this we write an augmented error function
- $E' = \text{error on data} + \lambda \cdot \text{model complexity}$
- The second term penalizes the complex models with large variance. Where λ gives the weight for this penalty. We minimizes the augmented error function instead of error on the data only. We penalize complex models and this decrease variance.
- If λ is too large , only a very simple model is allowed and there is a risk of introducing bias. λ is optimized using cross validation.
- Considering $E' = \text{error on data} + \lambda \cdot \text{model complexity}$ The second term is known as optimism.
- Optimism estimates the error between the training and test error

Model Selection procedure

- Akaike's information criterion (AIC)and Baysian Information Criterion(BIC)

$$E' = \text{error on data} + \lambda \cdot \text{model complexity}$$

- In this optimism is estimated and it is added to the training error. In this validation is not conducted.
- The magnitude of optimism increases linearly with d, the number of inputs and decreases with N, training set size increases.
- It increases with σ^2 , the variance of the noise. For models that are not linear , d should be replaced with effective number of parameters

MACHINE LEARNING Unit –1

Model Selection procedure

❖ <u>Structural Risk Minimization</u>

❖ It uses the set of models ordered in terms of their complexities .

$$E' = \text{error on data} + \lambda \cdot \text{model complexity}$$

❖ In the above equations we can have a set of decreasing λ_i to get a set of models ordered in increasing complexity

Model Selection Procedure

- **Minimum Description Length:**
- **Kolmogrov complexity of a dataset is defined as the shortest description of the data.**
- **For example , if it is a sequence of '0' s , we can just write 0 and length of sequence**
- **If the data is simple , it has a short complexity.**
- **If the data is completely random, we can not have any description of the data shorter that data itself. If a model is appropriate for the data , then it is a good fit to the data.**
- **Out of all the models that describe the data, we want to have simplest model, that lends itself the shortest description.**
- **There is a trade of between simplicity of the model and description .**

MACHINE LEARNING Unit –1

Model Selection Procedure

- **Bayesian Model Selection:**
- $$p(\text{model}|\text{data}) = \frac{p(\text{data}|\text{model})\,p(\text{model})}{p(\text{data})}$$
- $p(model|data) = Posterori\ probability\ of\ the\ given\ model$
- $p(model) = Prior\ \ subjective\ knowledge$
- **Taking logarithm on both sides**
- $$\log p(\text{model}|\text{data}) = \log p(\text{data}|\text{model}) + \log p(\text{model}) - c$$
- **In this log likelihood of the data is the training error log(p(model)) is the penalty form**
- **In Case of regression we have**
- $$E = \sum_t [r^t - g(x^t|\mathbf{w})]^2 + \lambda \sum_i w_i^2$$

Model Selection Procedure

- $$E = \sum_t [r^t - g(x^t|\mathbf{w})]^2 + \lambda \sum_i w_i^2$$
- We look for w_i that decreases error and also it will be close to zero.
- Whenever it is zero the fitted polynomial is smoother. As the polynomial order increases, function will go up and down.

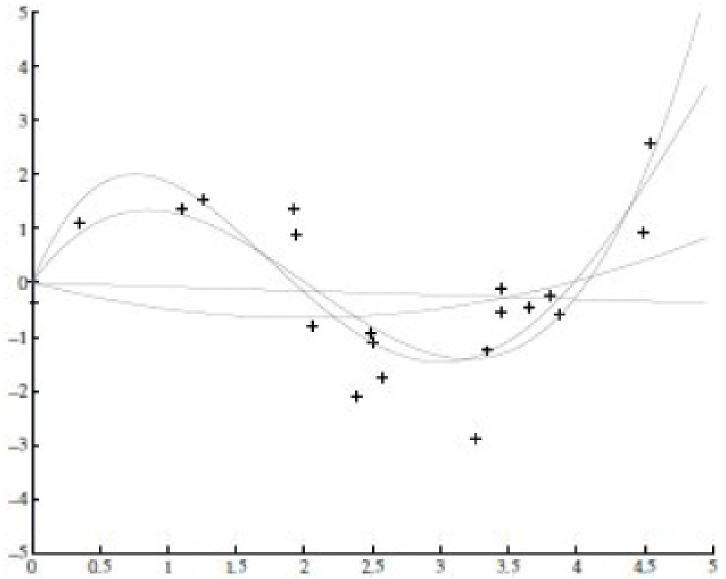


Figure 4.8 In the same setting as that of figure 4.5, polynomials of order 1 to 4 are fitted. The magnitude of coefficients increase as the order of the polynomial increases. They are as follows: 1 : $[-0.0769, 0.0016]^T$, 2 : $[0.1682, -0.6657, 0.0080]^T$, 3 : $[0.4238, -2.5778, 3.4675, -0.0002]^T$, 4 : $[-0.1093, 1.4356, -5.5007, 6.0454, -0.0019]^T$.

Multivariate Data

- Now we have to see the classification or regression in which we have multiple inputs.
- Output is a class code or continuous output is a function of multiple inputs. The inputs may be discrete or numeric
- The sample of multivariate data may be as follows

$$\mathbf{X} = \begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_d^1 \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & & & \\ X_1^N & X_2^N & \cdots & X_d^N \end{bmatrix}$$

- Where d-Columns represent d-variables, inputs , features or attributes
- N-rows corresponds to independent and identically distributed observations, examples or instances on N- individuals or events.
- Ex: For loan application assume there are N-Customers, Financial corporation may see their age, annual income, assets as features .

Multivariate Classification

- When $x \in \Re^d$, The conditional class densities $p(\mathbf{x}|C_i)$, are taken Normal density $\mathcal{N}_d(\boldsymbol{\mu}_i, \Sigma_i)$, , then we have

$$p(\mathbf{x}|C_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]$$

- For example , we want to predict the type of car that a customer would be interested in. Different cars are the classes and x are observable data of customers, for example age of the customer and income of the customer. $\boldsymbol{\mu}_i$ Is the mean vector of the mean age and income of the customers who buy car type i and Σ_i is their covariance matrix.

- σ_{i1}^2 and σ_{i2}^2 are age and income variance σ_{i12} is the covariance of age and income in Class i

❖ Let the discriminant function

$$g_i(\mathbf{x}) = \log p(\mathbf{x}|C_i) + \log P(C_i)$$

But

$$p(\mathbf{x}|C_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]$$

$$\begin{aligned}\hat{P}(C_i) &= \frac{\sum_t r_i^t}{N}\\ \boldsymbol{m}_i &= \frac{\sum_t r_i^t \mathbf{x}^t}{\sum_t r_i^t}\\ \mathbf{S}_i &= \frac{\sum_t r_i^t(\mathbf{x}^t-\boldsymbol{m}_i)(\mathbf{x}^t-\boldsymbol{m}_i)^T}{\sum_t r_i^t}\end{aligned}$$

MACHINE LEARNING Unit –2

Multivariate Classification

- $$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)^T\mathbf{S}_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i) + \log\hat{P}(C_i)$$

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}\left(\boldsymbol{x}^T\mathbf{S}_i^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T\mathbf{S}_i^{-1}\boldsymbol{m}_i + \boldsymbol{m}_i^T\mathbf{S}_i^{-1}\boldsymbol{m}_i\right) + \log\hat{P}(C_i)$$

- $$g_i(\boldsymbol{x}) = \boldsymbol{x}^T\mathbf{W}_i\boldsymbol{x} + \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}$$

- **Where**

$$\begin{aligned}
\mathbf{W}_i &= -\frac{1}{2}\mathbf{S}_i^{-1}\\
\boldsymbol{w}_i &= \mathbf{S}_i^{-1}\boldsymbol{m}_i\\
w_{i0} &= -\frac{1}{2}\boldsymbol{m}_i^T\mathbf{S}_i^{-1}\boldsymbol{m}_i - \frac{1}{2}\log|\mathbf{S}_i| + \log\hat{P}(C_i)
\end{aligned}$$

- ❖ The number of parameters to be estimated are K.d for the means and K.d(d+1)/2 for the covariance matrices.
- ❖ When d is large and Samples are Small, S_i may be Singular and inverses may not exist or $|S_i|$ may be non - zero but too small. For the estimates to be reliable on small samples , one may want to decrease the dimensionality
- ❖ OR Another possibility is to have Common Variance Matrix
- ❖ In case of common variance for all classes $\mathrm{S} = \sum_i \hat{P}(C_i)\mathrm{S}_i$

$$g_i(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^T \mathrm{S}^{-1} (\boldsymbol{x} - \boldsymbol{m}_i) + \log \hat{P}(C_i)$$

- ❖ The number of parameters of K.d and d(d+1)/2 for the shared covariance matrices.

Multivariate Classification

- ❖ If the prioris are equal ,optimal decision rule is to assign input to the class whose Mahalanobis distance to the input is smallest.
- ❖ In this case quadratic term $\boldsymbol{x}^T\mathbf{S}^{-1}\boldsymbol{x}$ is common in all discriminants and the decision boundaries are linear leading to linear discriminant.

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}$$

Where

$$\begin{aligned} \boldsymbol{w}_i &= \mathbf{S}^{-1}\boldsymbol{m}_i \\ w_{i0} &= -\frac{1}{2}\boldsymbol{m}_i^T\mathbf{S}^{-1}\boldsymbol{m}_i + \log \hat{P}(C_i) \end{aligned}$$

Further simplification may be possible by assuming all the off-diagonals of the covariance matrix to be 0, thus assuming independent variables. This is called naïve Bayes Classifier. Where $p(x_j|C_i)$ are Univariate diagonal. S and its inverse are diagonal

- There fore $g_i(\mathbf{x}) = -\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j^t - m_{ij}}{s_j}\right)^2 + \log \hat{P}(C_i)$
- Simplifying even further , if we assume all variances are equal , the Mahalanobis distance reduces to Euclidean distance. Then
- $|\mathbf{S}| = s^{2d}$ and $\mathbf{S}^{-1} = (1/s^2)\mathbf{I}$.
- $g_i(\mathbf{x}) = -\frac{\|\mathbf{x} - \boldsymbol{m}_i\|^2}{2s^2} + \log \hat{P}(C_i) = -\frac{1}{2s^2}\sum_{j=1}^{d}(x_j^t - m_{ij})^2 + \log \hat{P}(C_i)$
- If the priori are equal , then $g_i(\mathbf{x}) = -\|\mathbf{x} - \boldsymbol{m}_i\|^2$
- This is named as Nearest Mean Classifier

Multivariate Classification

- $$g_i(\mathbf{x}) = -\|\mathbf{x} - \mathbf{m}_i\|^2 = -(\mathbf{x} - \mathbf{m}_i)^T(\mathbf{x} - \mathbf{m}_i)$$
  $$= -(\mathbf{x}^T\mathbf{x} - 2\mathbf{m}_i^T\mathbf{x} + \mathbf{m}_i^T\mathbf{m}_i)$$

- $$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0}$$

- where $\mathbf{w}_i = \mathbf{m}_i$ and $w_{i0} = -(1/2)\|\mathbf{m}_i\|^2$.

- If $\mathbf{m}_i$ have similar norms, then this term can be neglected and we can have $g_i(\mathbf{x}) = \mathbf{m}_i^T\mathbf{x}$

❖ Reducing Variance through simplifying assumptions

Assumption	Covariance matrix	No. of parameters
Shared, Hyperspheric	$\mathbf{S}_i = \mathbf{S} = s^2\mathbf{I}$	1
Shared, Axis-aligned	$\mathbf{S}_i = \mathbf{S}$, with $s_{ij} = 0$	d
Shared, Hyperellipsoidal	$\mathbf{S}_i = \mathbf{S}$	$d(d+1)/2$
Different, Hyperellipsoidal	$\mathbf{S}_i$	$K \cdot (d(d+1)/2)$

Tuning Complexity

- There is a trade off between comfort of a simple model with generality.
- When assumptions about the covariance matrix are made and decrease the number of parameters to be estimated. Then we need to introduce bias
- Regularized Discriminant Analysis : In the case of parametric classification with Gaussian densities , the covariance matrices can be written as a weighted of the three special cases
- $S_i' = \alpha\sigma^2 I + \beta S + (1 - \alpha - \beta) S_i$
- $when\ \alpha = 0$, β=0 It leads to quadratic classifier
- $when\ \alpha = 0$, β=1 It leads to linear classifier
- $when\ \alpha = 1$, β=0 It leads to nearest mean classifier

MACHINE LEARNING Unit –2

Discrete Features

- In some applications we have discrete attributes taking one of n different values.
- For an example an attribute may be color ∈ {red , green, blue} or pixel ∈ {0,1}

$$p_{ij} \equiv p(x_j = 1|C_i)$$

- Where $C_i = Class$

$$p(\mathbf{x}|C_i) = \prod_{j=1}^{d} p_{ij}^{x_j}(1-p_{ij})^{(1-x_j)}$$

- The discriminant function
- $$\begin{aligned} g_i(\mathbf{x}) &= \log p(\mathbf{x}|C_i) + \log P(C_i) \\ &= \sum_j \left[x_j \log p_{ij} + (1-x_j)\log(1-p_{ij})\right] + \log P(C_i) \end{aligned}$$ Linear

MACHINE LEARNING Unit –2

- The estimator for p_{ij} is $\hat{p}_{ij} = \frac{\sum_t x_j^t r_i^t}{\sum_t r_i^t}$
- This approach is used in document categorization. Ex: Classifying news reports. In the bag of words representation we choose a priori d- words
- After training $\hat{p}_{ij}$ estimates the probability that word j occurs document type i
- In the general case instead of binary features we have multinomial x_j chosen from the set $\{v_1, v_2, \ldots, v_{n_j}\}$.

$$z_{jk}^t = \begin{cases} 1 & \text{if } x_j^t = v_k \\ 0 & \text{otherwise} \end{cases}$$

MACHINE LEARNING Unit –2

- p_{ijk} = *probablity the* x_j *belonging to Class* C_i takes value v_k

- $$p_{ijk} \equiv p(z_{jk} = 1|C_i) = p(x_j = v_k|C_i)$$

- If all the attributes are independent , then $p(\mathbf{x}|C_i) = \prod_{j-1}^{d} \prod_{k-1}^{n_j} p_{ijk}^{z_{jk}}$ then

- Then discriminant function $g_i(\mathbf{x}) = \sum_j \sum_k z_{jk} \log p_{ijk} + \log P(C_i)$

MACHINE LEARNING Unit –2

- **The maximum likelihood estimator for p_{ijk} is**
- $$\hat{p}_{ijk} = \frac{\sum_t z_{jk}^t r_i^t}{\sum_t r_i^t}$$

MACHINE LEARNING Unit –2

Multivariate Regression

- In Multivariate Linear Regression , the numeric output r is assumed to be written as linear function, that is , a weighted sum , of several input variables $x_1, \ldots, x_d$, and noise.

- Actually in Statistical literature , this is called the multiple regression. Statisticians use the term multivariate when there are multiple outputs. The multivariate linear model is

$$r^t = g(\mathbf{x}^t | w_0, w_1, \ldots, w_d) + \epsilon = w_0 + w_1 x_1^t + w_2 x_2^t + \cdots + w_d x_d^t + \epsilon$$

- We assume ε to be normal with mean zero and constant variance , and maximizing the sum of squared errors

- $$E(w_0, w_1, \ldots, w_d | \mathcal{X}) = \frac{1}{2} \sum_t (r^t - w_0 - w_1 x_1^t - w_2 x_2^t - \cdots - w_d x_d^t)^2$$

Multivariate Regression

$$r^t = g(\mathbf{x}^t | w_0, w_1, \ldots, w_d) + \epsilon = w_0 + w_1 x_1^t + w_2 x_2^t + \cdots + w_d x_d^t + \epsilon$$

$$E(w_0, w_1, \ldots, w_d | \mathcal{X}) = \frac{1}{2} \sum_t (r^t - w_0 - w_1 x_1^t - w_2 x_2^t - \cdots - w_d x_d^t)^2$$

$$\begin{aligned}
\sum_t r^t &= N w_0 + w_1 \sum_t x_1^t + w_2 \sum_t x_2^t + \cdots + w_d \sum_t x_d^t \\
\sum_t x_1^t r^t &= w_0 \sum_t x_1^t + w_1 \sum_t (x_1^t)^2 + w_2 \sum_t x_1^t x_2^t + \cdots + w_d \sum_t x_1^t x_d^t \\
\sum_t x_2^t r^t &= w_0 \sum_t x_2^t + w_1 \sum_t x_1^t x_2^t + w_2 \sum_t (x_2^t)^2 + \cdots + w_d \sum_t x_2^t x_d^t \\
&\vdots \\
\sum_t x_d^t r^t &= w_0 \sum_t x_d^t + w_1 \sum_t x_d^t x_1^t + w_2 \sum_t x_d^t x_2^t + \cdots + w_d \sum_t (x_d^t)^2
\end{aligned}$$

Estimation of missing values

- Frequently, values of certain variables may be missing observations. We try to fill these variables by estimating them. This is called imputation.

- Mean imputation : For a numeric variable , we substitute the mean of the available data for that variable in the sample.
- For a discrete variable , we fill with most likely value

- Imputation by Regression : We try to predict the value of a missing variable from other variables whose values are known for that case. If many different variables are missing, we take the means as the initial estimates and the procedure is iterated until predicted value stabilizes.
  If the variables are not highly correlated, the regression approach is equivalent to mean imputation.

Multivariate Normal distribution

- In Multivariate case x is d-dimensional and normal distributed
- x follows normal distribution with mean vector $\boldsymbol{\mu}$ and Σ is the covariance matrix

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

- In the multivariate case Mahalanobis distance is used

$$(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

- $(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = c^2$ is the d-dimensional hyper ellipsoid

  centered at $\boldsymbol{\mu}$, and its shape and orientation are defined by Σ.

- Because of the use of inverse of Σ. If a variable has a larger variance , it receives less weight in the Mahalanobis distance.

Multivariate Normal distribution - Continued

- For d= 2 , The mean vector $\boldsymbol{\mu}^T = [\mu_1, \mu_2]$,
- The covariance matrix $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$
- The joint bivariate density is

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(z_1^2 - 2\rho z_1 z_2 + z_2^2\right)\right]$$

- Where $z_i = (x_i - \mu_i)/\sigma_i,\ i = 1, 2,$ are standardized variables . This is called z-normalization
- In the multivariate case , a small value of $|\boldsymbol{\Sigma}|$ indicates sample values are close to $\boldsymbol{\mu}$,

MACHINE LEARNING Unit –2

Multivariate Normal distribution - Continued

- If $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, a special naïve case is that where the components of x are independent and $\mathrm{Cov}(X_i, X_j) = 0, \text{ for } i \neq j, \text{ and } \mathrm{Var}(X_i) = \sigma_i^2, \forall i.$. Then the converse matrix is diagonal and the joint density is the product of individual univariate densities .

$$p(\boldsymbol{x}) = \prod_{i=1}^{d} p_i(x_i) = \frac{1}{(2\pi)^{d/2} \prod_{i=1}^{d} \sigma_i} \exp\left[-\frac{1}{2} \sum_{i=1}^{d} \left(\frac{x_i - \mu_i}{\sigma_i}\right)^2\right]$$

- Let us say $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\boldsymbol{w} \in \Re^d$, Then

$$\mathbf{w}^T \boldsymbol{x} = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d \sim \mathcal{N}(\mathbf{w}^T \boldsymbol{\mu}, \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w})$$

MACHINE LEARNING Unit –2

Multivariate Normal distribution - Continued

$$\begin{aligned}
E[\mathbf{w}^T\mathbf{x}] &= \mathbf{w}^T E[\mathbf{x}] = \mathbf{w}^T\boldsymbol{\mu} \\
\mathrm{Var}(\mathbf{w}^T\mathbf{x}) &= E[(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})^2] = E[(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})] \\
&= E[\mathbf{w}^T(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{w}] = \mathbf{w}^T E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T]\mathbf{w} \\
&= \mathbf{w}^T\Sigma\mathbf{w}
\end{aligned}$$

- ❖ In general case W is a d x k matrix with a rank k<d , then the K-dimensional $\mathrm{W}^T\mathbf{x}$ k-variate normal $\mathrm{W}^T\mathbf{x} \sim \mathcal{N}_k(\mathrm{W}^T\boldsymbol{\mu}, \mathrm{W}^T\Sigma\mathrm{W})$ That is, if we project a d-dimensional normal distribution to a space that is k-dimensional , Then it projects a k-dimensional normal

- ❖ When $x \in \Re^d$, The conditional class densities $p(\mathbf{x}|C_i)$, are taken Normal density $\mathcal{N}_d(\boldsymbol{\mu}_i, \Sigma_i)$, , then we have

$$p(x|C_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right]$$

- ❖ For example , we want to predict the type of car that a customer would be interested in. Different cars are the classes and x are observable data of customers, for example age of the customer and income of the customer. $\boldsymbol{\mu}_i$ Is the mean vector of the mean age and income of the customers who buy car type i and Σ_i is their covariance matrix.

- ❖ σ_{i1}^2 and σ_{i2}^2 are age and income variance σ_{i12} is the covariance of age and income in Class i

- **Let the discriminant function**

$$g_i(\mathbf{x}) = \log p(\mathbf{x}|C_i) + \log P(C_i)$$

But

$$p(\mathbf{x}|C_i) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]$$

$$\begin{aligned} \hat{P}(C_i) &= \frac{\sum_t r_i^t}{N} \\ \boldsymbol{m}_i &= \frac{\sum_t r_i^t \mathbf{x}^t}{\sum_t r_i^t} \\ \mathbf{S}_i &= \frac{\sum_t r_i^t (\mathbf{x}^t - \boldsymbol{m}_i)(\mathbf{x}^t - \boldsymbol{m}_i)^T}{\sum_t r_i^t} \end{aligned}$$

MACHINE LEARNING Unit –2

Multivariate Classification

❖ $$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathrm{S}_i| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)^T\mathrm{S}_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i) + \log\hat{P}(C_i)$$

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathrm{S}_i| - \frac{1}{2}\left(\boldsymbol{x}^T\mathrm{S}_i^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T\mathrm{S}_i^{-1}\boldsymbol{m}_i + \boldsymbol{m}_i^T\mathrm{S}_i^{-1}\boldsymbol{m}_i\right) + \log\hat{P}(C_i)$$

❖ $$g_i(\boldsymbol{x}) = \boldsymbol{x}^T\mathrm{W}_i\boldsymbol{x} + \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}$$

❖ **Where**

$$\begin{aligned}
\mathrm{W}_i &= -\frac{1}{2}\mathrm{S}_i^{-1} \\
\boldsymbol{w}_i &= \mathrm{S}_i^{-1}\boldsymbol{m}_i \\
w_{i0} &= -\frac{1}{2}\boldsymbol{m}_i^T\mathrm{S}_i^{-1}\boldsymbol{m}_i - \frac{1}{2}\log|\mathrm{S}_i| + \log\hat{P}(C_i)
\end{aligned}$$

Multivariate Regression

- In Multivariate Linear Regression , the numeric output r is assumed to be written as linear function, that is , a weighted sum , of several input variables $x_1, \ldots, x_d$, and noise.
- Actually in Statistical literature , this is called the multiple regression. Statisticians use the term multivariate when there are multiple outputs. The multivariate linear model is

$$r^t = g(\boldsymbol{x}^t | w_0, w_1, \ldots, w_d) + \epsilon = w_0 + w_1 x_1^t + w_2 x_2^t + \cdots + w_d x_d^t + \epsilon$$

- We assume ***ε*** to be normal with mean zero and constant variance , and maximizing the sum of squared errors
- $$E(w_0, w_1, \ldots, w_d | \mathcal{X}) = \frac{1}{2} \sum_t (r^t - w_0 - w_1 x_1^t - w_2 x_2^t - \cdots - w_d x_d^t)^2$$

MACHINE LEARNING Unit –2

$$E(w_0, w_1, \ldots, w_d | \mathcal{X}) = \frac{1}{2} \sum_t (r^t - w_0 - w_1 x_1^t - w_2 x_2^t - \cdots - w_d x_d^t)^2$$

- Taking derivative with respect to w_j , j=0,1,…..d, we get the following normal equation.

$$\begin{aligned}
\sum_t r^t &= N w_0 + w_1 \sum_t x_1^t + w_2 \sum_t x_2^t + \cdots + w_d \sum_t x_d^t \\
\sum_t x_1^t r^t &= w_0 \sum_t x_1^t + w_1 \sum_t (x_1^t)^2 + w_2 \sum_t x_1^t x_2^t + \cdots + w_d \sum_t x_1^t x_d^t \\
\sum_t x_2^t r^t &= w_0 \sum_t x_2^t + w_1 \sum_t x_1^t x_2^t + w_2 \sum_t (x_2^t)^2 + \cdots + w_d \sum_t x_2^t x_d^t \\
&\vdots \\
\sum_t x_d^t r^t &= w_0 \sum_t x_d^t + w_1 \sum_t x_d^t x_1^t + w_2 \sum_t x_d^t x_2^t + \cdots + w_d \sum_t (x_d^t)^2
\end{aligned}$$

MACHINE LEARNING Unit –2

- Let us define the following vectors and matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_d^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & & & & \\ 1 & x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d, \end{bmatrix}, \boldsymbol{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

- Then the normal equations can be written as $\mathbf{X}^T\mathbf{X}\boldsymbol{w} = \mathbf{X}^T\boldsymbol{r}$

- We can solve for the parameters $\boldsymbol{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{r}$

- One advantage of linear models is that after the regression looking at the w_j , j=1,2,…d , we can extract the knowledge .
- First by looking at the signs of w_j we can see whether x_j have a positive or negative effect on the output.
- Second , if all x_j are in the same range , by looking at the absolute values w_j , we can get an idea how important a feature is , rank the feature in terms of their importance. Remove those features whose w_j are close to zero.

THANK YOU

Raghavendra.M.J
Department of ECE