Page No.:
Date:
YOUVA

Resources : Book - Good fellow
: videos - Fifi Lee, Nando de freitas
Ali Ghodsi

→ Machine learning classified into :
1) supervised learning
2) Unsupervised learning
3) semi - supervised learning.

→ $x^{(i)}$ — data / feature vector
$y^{(i)}$ - output (mostly scalar)

→ In supervised learning, data is labelled (answer is known)
→ Based on training examples, we try developing relation between input and output
→ Methods - (supervised learning techniques)
1) Linear regression → Except this, all
2) Logistic regression others are
3) Naive Baye's classification classification
4) Linear discriminant analysis
5) Quadratic discriminant analysis.
6) support vector machine
7) Neural Network classification

→ Classification - Discrete outputs only (Identify digit from image)

→ In unsupervised learning, we try finding some coherent structure in the data. Here there is only input, no output. (unsupervised learning also called segmentation).
→ Techniques — 1) K means clustering
2) principle component analysis

3) Independent component analysis
4) Gaussian mixture model (GMM)

Disadvantage of S.L. - manually label data, which is tedious.

Linear regression

Eg. Predicting price of flat. Following training set

Size (sq. feet)		Price (in Rs)
2000		80K
1400	Suppose such so examples	50K
1500		60K
872		15K

→ Hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$. (linear model)

→ To fit a model for given problem, we

$$\underset{\text{wrt } \theta}{\text{minimize}} \quad J(\theta) = \frac{1}{2m} \sum \left(\underbrace{h_\theta(x^{(i)})}_{\substack{\text{model} \\ \text{output} \\ (\hat{y}^{(i)})}} - \underbrace{y^{(i)}}_{\substack{\text{given} \\ \text{output}}} \right)^2$$

where $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}^T$

→ Regression - a measure to represent relation between input and output

→ Linear regression - regression with linear relationship between input and output. (Here $y^{(i)}$ can be real, unlike classification)

→ Here our aim is to find θ_0 and θ_1. We do it as follows

$y^{(i)} = h_\theta(x^{(i)}) + \epsilon^{(i)}$, $i = 1, \ldots, m$ where $\epsilon^{(i)}$, $i = 1, \ldots, m$ are errors

Page No.:
Date:

$$\epsilon = \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(m)} \end{bmatrix} \quad \text{gives} \quad \|\epsilon\|^2 = (\epsilon^{(1)})^2 + (\epsilon^{(2)})^2 + \cdots + (\epsilon^{(m)})^2$$

So we need to minimize $\frac{1}{2m} \|\epsilon\|^2$

$$\underline{Y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad X = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \\ 1 & x^{(m)} \end{bmatrix} \quad \underline{\Theta} = \begin{bmatrix} \Theta_0 \\ \Theta_1 \end{bmatrix}$$

$J(\Theta) = \frac{1}{2m} \| \underline{Y} - X\underline{\Theta} \|^2$ → loss function (objective function) or cost function

$= \frac{1}{2m} \sum (y^{(i)} - \Theta_0 - \Theta_1 x^{(i)})^2$

→ Eg. $J(\Theta) = \Theta_1^2 + \Theta_2^2$

$\frac{\partial J(\Theta)}{\Theta_1} = 2\Theta_1 = 0 \Rightarrow \Theta_1 = 0$

$\frac{\partial J(\Theta)}{\Theta_2} = 2\Theta_2 = 0 \Rightarrow \Theta_2 = 0$

For min

→ $\nabla J(\Theta) = \begin{bmatrix} \frac{\partial J(\Theta)}{\Theta_0} \\ \frac{\partial J(\Theta)}{\Theta_1} \end{bmatrix} = \underline{0}$ (zero vector)

→ $J(\Theta) = \frac{1}{2m} \sum (y^{(i)} - \Theta_0 - \Theta_1 x^{(i)})^2$

$\frac{\partial J(\Theta)}{\Theta_0} = \frac{1}{2m} \sum 2 (y^{(i)} - \Theta_0 - \Theta_1 x^{(i)}) \cdot (-1) = -\frac{1}{m} \sum (y^{(i)} - \Theta_0 - \Theta_1 x^{(i)})$

$$\frac{\partial J(\theta)}{\theta_1} = \frac{1}{2m} \Sigma 2 \left(y^{(i)} - \theta_0 - \theta_1 x^{(i)} \right) \left(-x^{(i)} \right)$$

→ Taking $m = 2$

$$J(\theta) = \frac{1}{2(2)} \left[\begin{array}{l} \left(y^{(1)} - (\theta_0 + \theta_1 x^{(1)}) \right)^2 + \\ \left(y^{(2)} - (\theta_0 + \theta_1 x^{(2)}) \right)^2 \end{array} \right]$$

$$\frac{\partial J(\theta)}{\theta_0} = \frac{-1}{2} \left[\begin{array}{l} \left[y^{(1)} - (\theta_0 + \theta_1 x^{(1)}) \right] + \\ \left[y^{(2)} - (\theta_0 + \theta_1 x^{(2)}) \right] \end{array} \right]$$

$$= \frac{-1}{2} \left[\sum_{i=1}^{2} y^{(i)} - \theta_0 - \theta_1 x^{(i)} \right]$$

$$\frac{\partial J(\theta)}{\theta_1} = -\frac{1}{2} \left[\begin{array}{l} \left(y^{(i)} - (\theta_0 + \theta_1 x^{(1)}) \right) x^{(1)} + \\ \left(y^{(2)} - (\theta_0 + \theta_1 x^{(2)}) \right) x^{(2)} \end{array} \right]$$

$$= \frac{-1}{2} \left[\sum_{i=1}^{2} \left(y^{(i)} - \theta_0 - \theta_1 x^{(i)} \right) x^{(i)} \right]$$

In general,

$$\nabla J(\theta) = \frac{-1}{m} \left[\begin{array}{l} \sum_{i=1}^{m} y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \\ \sum_{i=1}^{m} \left(y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \right) x^{(i)} \end{array} \right]$$

$$= \frac{-1}{m} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \end{bmatrix} \begin{bmatrix} y^{(1)} - (\theta_0 + \theta_1 x^{(1)}) \\ y^{(2)} - (\theta_0 + \theta_1 x^{(2)}) \\ \vdots \\ y^{(m)} - (\theta_0 + \theta_1 x^{(m)}) \end{bmatrix}$$

$$\theta = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \end{bmatrix} = -\frac{1}{m} X^T (Y - X\theta) = 0$$

X^T: $2 \times m$, $(Y - X\theta)$: $m \times 1$ $\rightarrow$ 2×1

Hence $X^T Y = X^T X \theta$ gives $\theta = (X^T X)^{-1} X^T Y$

$(X^T X)^{-1} X^T$ $\rightarrow$ pseudo inverse of X

Here we have m equations and 2 unknowns. This is overdetermined set of equations.

$Y = X\theta \Rightarrow \theta = X^{-1} Y$ but this works only when X is a square matrix. But that's not the case here. Hence we use pseudo inverse.

→ Suppose we had more number of features;

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \quad x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ x_3^{(i)} \end{bmatrix} \quad h_\theta(x^{(i)}) = \theta^T x^{(i)}$$

Probabilistic interpretation of LR

→ Earlier we considered $\epsilon^{(i)}$ as error between $y^{(i)}$ and $h_\theta(x^{(i)})$. Consider $\epsilon^{(i)}$ as a R.V. Assume that $\epsilon^{(i)}$ is gaussian distributed. $N(0, \sigma^2)$. Here $f(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\epsilon_i^2 / 2\sigma^2}$

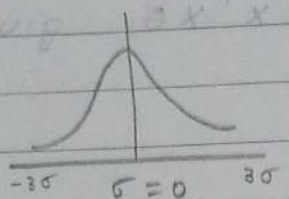
↓

Probability of $\epsilon^{(i)}$

Date:

Consider $y^{(i)} = \theta_0 + \theta_1 x^{(i)} + \epsilon^{(i)}$

Error is gaussian distributed. So value drawn everytime is random but its closely in surrounding of the mean



EQ Given $\epsilon^{(i)}$ is gaussian. What is probability of $y^{(i)}$ given $x^{(i)}$

given so constant

$y^{(i)} = \theta_0 + \theta_1 x^{(i)} + \epsilon^{(i)}$

$P(y^{(i)} | x^{(i)}; \theta)$ has the form of gaussian $N(\theta^T x^{(i)}, \sigma^2)$.

shifted mean

(By adding any mean changes, variance same)

$$P(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}$$

x parameterized by θ.

→ We want to solve LR using probabilistic interpretation, one of the approach to formulate the problem is to seek those θ's (θ_0, θ_1) such that $L(\theta) = P(y^{(1)}, y^{(2)}, \ldots, y^{(m)} / x^{(1)}, x^{(2)} \ldots x^{(m)}; \theta)$ is maximum.

(Choose each θ_i that maximizes the chances of getting those y_i for given x_i).

$L(\theta)$ is called the likelihood function.

(independent, identically distributed)

→ Lets consider $\epsilon^{(i)}$'s as iid with $N(0, \sigma^2)$

$y^{(1)} = \theta_0 + \theta_1 x^{(1)} + \epsilon^{(1)}$ independent

$y^{(2)} = \theta_0 + \theta_1 x^{(2)} + \epsilon^{(2)}$ of each other

Hence $L(\theta) = P(y^{(1)} | x^{(1)}; \theta)\, P(y^{(2)} | x^{(2)}; \theta) \ldots$

$P(y^{(m)} | x^{(m)}; \theta)$

$$= \prod_{i=1}^{m} P(y^{(i)} | x^{(i)}; \theta)$$

$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}$$

→ So the problem can be formulated as follows

$\max_{\theta} L(\theta)$ (Max possible value is 1).

→ This formulation is called maximum likelihood estimation of θ (MLE).

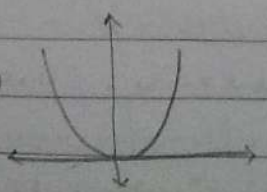
→ Maximization can be simplified by taking $l(\theta)$ instead of $L(\theta)$ where $l(\theta) = \log[L(\theta)]$.

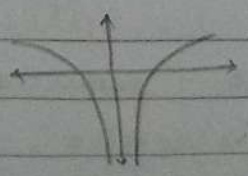
Reason for taking log ---

→ $L(\theta) = \theta^2$. $L(\theta)_{min} = 0$ at $\boxed{\theta = 0}$

→ $l(\theta) = \log(L(\theta))$ $l(\theta)_{min} = -\infty$ at $\boxed{\theta = 0}$

Conclusion — we are only interested in θ at which function maximizes / minimizes. This does not change by taking log. Hence fine.





$\theta^T x$ — line in linear regression (In general a hyperplane)

Page No.: Date: YOUVA

So here we will maximize $l(\theta)$ in place of $L(\theta)$ to simplify calculation.

$$l(\theta) = \sum_{i=1}^{m} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}} \right]$$

$$= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi\sigma^2}} - \sum_{i=1}^{m} \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

↓ constant is independent of θ

→ So what we need is —

$$\max_{\forall\theta} l(\theta) = \max_{\forall\theta} - \sum_{i=1}^{m} \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$ Equivalently

$$\min_{\forall\theta} l(\theta) = \min_{\forall\theta} \sum_{i=1}^{m} \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

→ Estimated $\theta^* = \min_{\forall\theta} l(\theta) = \min_{\forall\theta} \sum_{i=1}^{m} \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$

This minimization leads to least square estimation of θ which was obtained earlier by pseudo inverse approach. (similar expression with just σ^2 in place of m)

both solved by pseudo inverse.

Hence MLE method is equivalent to pseudo inverse method. (least square approach).

Note that the noise error here is gaussian iid $N(0, \sigma^2)$. If this changes then the expression changes.

→ White noise — mean of each element = 0, variance of each element is finite, all elements are uncorrelated

→ iid noise — white + all elements independent also

For non gaussian noise, MLE is not equivalent to least square. MLE depends of $P(\epsilon_i)$. Since least square doesn't work here, use MLE with pdf of non gaussian ϵ_i

Page No.:
Date:
YOUVA

★ What happens if the noise error is not gaussian?

Typical non gaussian noises are levy and studdred gaussian noise. We must note that gaussian noise is a precondition for using least square method. Using this method on non gaussian noise gives wrong estimations (specially when noise levels > 5%). Modifying least mean square method for non gaussian noise is under study.

What happens if errors are not iid?

If errors are not iid, then we will not be able to write joint probality $P(y^{(1)} \dots y^{(m)} | x^{(1)} \dots x^{(m)}, \theta)$ as product of individual probabilities. The distribution here now is jointly gaussian.

$p(x_1, x_2) = p(x_1|x_2)\, p(x_2)$ ⟶ Taking just two jointly gaussian RVs

Here $p(x_1|x_2) = N(x_1 | \mu_{1|2}, \Sigma_{1|2})$

$p(x_2) = N(x_2 | \mu_2, \Sigma_{22})$

As number of non iid RVs increases, it becomes extremely difficult to calculate the estimation manually

$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$

$\Sigma_{12} = \Sigma / \Sigma_{22} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$

Page No.:
Date:

→ Sometimes, some features are more important than other in $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$. Here, we can get better estimate of θ if we use prior knowledge about θ. This also gives better fit for given data. (i.e. a better model / hypothesis.). This method is called regularisation in ML. In probabilistic terms its called maximum a posteriorie (MAP) estimation of θ. For regression problem is refferred to as ridge regression.

Here, we need some prior knowledge of θ. Generally we consider θ also as a distribution mostly gaussian.

Eg. Face recognition — Image as a matrix of pixels. Now we know that there is huge change in intensity when there is an edge. Around cheeks, same intensity. So we have some idea from beginning.

→ When we know something about what could our answer θ be, we will get better results. So clearly MAP is better than MLE

→ For MLE we maximize $L(\theta) = P(y^{(1)} \cdots y^{(m)} \mid x^{(1)} \cdots x^{(m)}; \theta)$

→ For MAP, problem is formulated as follows:

$$\max_{\theta} L(\theta) = \max_{\theta} \frac{P(y^{(1)} \cdots y^{(m)}, x^{(1)} \cdots x^{(m)} \mid \theta)\, P(\theta)}{P(y^{(1)} \cdots y^{(m)}, x^{(1)} \cdots x^{(m)})}$$

Using Bayes Rule $P(A|B)\,P(B) = P(B|A)\,P(A)$ we get...

$$\max_{\theta} L(\theta) = \max_{\theta} P(\theta \mid y^{(1)} \dots y^{(m)}, x^{(1)} \dots x^{(m)})$$

→ So this means we are trying to maximize the probability of getting a particular θ given the data we have. So the θ corresponding to the max value of probability is the answer.

→ From ①, we can ignore denominator as it is independent of θ so maximisation is ...

$$\max_{\theta} L(\theta) = \max_{\theta} P(y^{(1)} \dots y^{(m)}, x^{(1)} \dots x^{(m)} \mid \theta) \times P(\theta)$$

y — output (real number)
x — feature (maybe a feature vector also)

→ Now y_i, x_i are dependent on each other but each pair of y_i, x_i are independent of one another.

$$L(\theta) = \left[\prod_{i=1}^{m} P(y^{(i)}, x^{(i)} \mid \theta)\right] P(\theta)$$

$$= \left[\prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)}, \theta)\, P(x^{(i)} \mid \theta)\right] P(\theta)$$

$P(\theta)$ ↓ Prior probability that we are trying to use

→ Note that θ and $x^{(i)}$'s are independent so $P(x^{(i)} \mid \theta) = P(x^{(i)})$ and this is now not dependent on θ. So ...

$$L(\theta) = \left[\prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)}, \theta)\right] P(\theta)$$

additional to MLE

→ So here we consider θ also as a RV and take it as gaussian distributed $\mathcal{N}(0, \tau^2)$
Also we consider multiple θ as iid.

$$P(\theta) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\theta^2/2\tau^2}$$ → For each of $\theta_0, \theta_1, \ldots, \theta_n$

We actually need $P(\theta_0, \theta_1, \ldots, \theta_n)$ But since iid, it is $P(\theta_0) P(\theta_1) \ldots P(\theta_n)$
which is $P(\theta) = \left(\frac{1}{\sqrt{2\pi\tau^2}}\right)^{n+1} e^{-\|\theta\|^2/2\tau^2}$

(Here θ is the entire vector and $\|\theta\|^2 = \theta_0^2 + \theta_1^2 + \ldots + \theta_n^2$)

$$L(\theta) = \prod_{i=1}^{m} \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}\right] \left(\frac{1}{\sqrt{2\pi\tau^2}}\right)^{n+1} e^{-\|\theta\|^2/2\tau^2}$$

We ignore the constants which are independent of θ and then take log before maximizing

$$L(\theta) = \prod_{i=1}^{m} \left[e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}\right] e^{-\|\theta\|^2/2\tau^2}$$

$$l(\theta) = \sum_{i=1}^{m} \left[-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right] - \frac{\|\theta\|^2}{2\tau^2}$$

$$\max_{\forall(\theta)} l(\theta) = \max -\left[\sum \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} + \frac{\|\theta\|^2}{2\tau^2}\right]$$

$$\theta^* = \min_{\theta} \left[\frac{1}{2\sigma^2} \sum (y^{(i)} - \theta^T x^{(i)})^2 + \frac{1}{2\tau^2} \|\theta\|^2\right]$$

↓ Objective function

This means θ^* is that value of θ that minimizes the objective function. This is regularization.

→ Note that $\frac{\|\theta\|^2}{2\tau^2}$ is the additional term for MAP from MLE

→ σ^2 - variance of errors
τ^2 - variance of θ (prior knowledge)

→ $\theta^* = \min_{\theta} \frac{1}{2\sigma^2}\left[\sum (y^{(i)} - \theta^T x^{(i)})^2 + \frac{\sigma^2}{\tau^2}\|\theta\|^2\right]$

regularisation parameter λ

(Note that estimating λ is also a problem within itself. Here, instead we simply assume λ)

→ So we minimize the following for MAP

$$L(\theta) = \frac{1}{2\sigma^2}\left[\sum (y^{(i)} - \theta^T x^{(i)})^2 + \lambda\|\theta\|^2\right]$$

Hoping to get a close formed solution, we differentiate this wrt θ and equate to 0.

partial differential wrt $\theta_0, \theta_1, \ldots, \theta_n$

So we derivate $\sum (y^{(i)} - \theta^T x^{(i)})^2 + \lambda\|\theta\|^2$

↓

This is equivalently $\|Y - X\theta\|^2$

$$Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad X = \begin{bmatrix} 1 & x^{(1)} & \cdots & (x^{(1)})^n \\ 1 & x^{(2)} & & (x^{(2)})^n \\ & & \vdots & \\ 1 & x^{(m)} & & (x^{(m)})^n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$m \times 1$ $\quad m \times (n+1)$ $\quad (n+1) \times 1$

↓

Each row is $x^{(i)}$

* $\|Y - X\theta\|^2$ $\qquad$ $\lambda \|\theta\|^2$

$\downarrow d$ $\qquad$ $\downarrow d$

$-2X^T[Y - X\theta]$ $(n+1 \times 1)$ $\qquad$ $2\lambda\theta$ $(n+1 \times 1)$

→ Hence $-X^TY + X^TX\underline{\theta} + \lambda\underline{\theta} = \underline{0}$ (zero vector)

$X^TY = (X^TX + \lambda \underline{I})\,\theta$

$m \times m$ unit vector

→ Hence $\underset{(n+1)\times 1}{\underline{\theta}^*} = (X^TX + \lambda I)^{-1} X^TY$ → MAP estimate of θ

additional term : better $\lambda \Rightarrow$ better solution

In real life n is of the order of $10^4 - 10^6$
In this case finding inverse is almost impossible due to computational inefficiency.

In such cases we use iterative approach — gradient descent.

* $J(\theta) = \Sigma (y^{(i)} - \theta_0 - \theta_1 x^{(i)} - \theta_2 (x^{(i)})^2 \ldots - \theta_n (x^{(i)})^n)^2$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = \begin{bmatrix} -2 \Sigma (\quad) \\ -2 \Sigma (\quad) x^{(i)} \\ \\ -2 \Sigma (\quad)(x^{(i)})^n \end{bmatrix}$$

Page No.
Date
YOUVA

EQ Under what conditions MLE and MAP are equal?

Ans When θ is a uniform distribution i.e when $P(\theta)$ is constant, MAP will becomes equivalent to MLE as $L(\theta)$ will no longer depend on $P(\theta)$.

→ Types of gradient descent : stochastic, moment based etc

→ $J(\theta)$ used while minimizing the objective func. was $J(\theta) = \frac{1}{2m} \sum [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})]^2$

→ By using G.D. θ_0 and θ_1 can be computed as follows:

① $$\theta_0^{k+1} = \theta_0^k - \alpha \frac{\partial J(\theta)}{\partial \theta_0} \Big|_{\theta_0 = \theta_0^k}$$

$$\theta_1^{k+1} = \theta_1^k - \alpha \frac{\partial J(\theta)}{\partial \theta_1} \Big|_{\theta_1 = \theta_1^k}$$

(α is the learning rate (+ve))

(Here k represents the iteration number)

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{2m} \sum_{i=1}^{m} 2 (y^{(i)} - \theta_0 - \theta_1 x^{(i)}) (-1)$$

$$= -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})]$$

$$\frac{\partial J}{\partial \theta_1} = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})] x^{(i)}$$

→ Assume some initial parameter values θ_0^0, θ_1^0

→ Keep using ① iteratively. Stop the iteration when algorithm converges at $\|\theta^{k+1} - \theta^k\|^2 <$ small value

→ If we use entire set of examples, then it is called batch gradient descent. (Σ over 1 to m)

→ In stochastic G.D we do not use summation we use randomly any one example in the iteration. Even then algorithm converges. This saves a lot of computation (SGD)

→ Most important assumption - $J(\theta)$ is differentiable (However there are techniques to work with non differential functions as well - genetic approach)

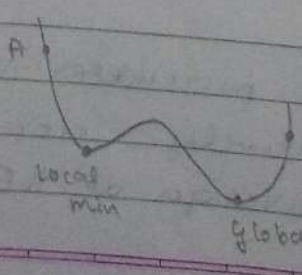
→ Problem with GD - we might get stuck at local min and not reach global min. However stochastic GD helps solve this problem in majority cases.

GD Algo ---

Given $J(\theta)$ which is differentiable write $k+1^{th}$ iteration θ values as

$$\theta^{k+1} = \theta^k - \underset{\alpha > 0}{\alpha} \nabla J(\theta) \mid \theta = \theta^k$$

where $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ $\nabla J(\theta) = \begin{bmatrix} \partial J / \partial \theta_0 \\ \partial J / \partial \theta_1 \\ \vdots \\ \partial J / \partial \theta_n \end{bmatrix}$



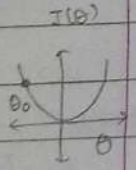
Initial A gives incorrect one

Initial B gives correct one.

Page No.:
Date:
YOUVA

Algorithm stops when $\nabla J(\theta)$ is ≈ 0
(This is what we usually do to find min - derivate and equate with 0).

EG $J(\theta) = \theta^2$ Try to find θ when $J(\theta)$ is minimum.



$$J(\theta_0 + s) = J(\theta_0) + \left.\frac{dJ}{d\theta}\right|_{\theta=\theta_0} s + \text{higher order terms.}$$

We try to approximate the function with a line at θ_0. Take $\theta = \theta_0 + s$.

$$J(\theta) = J(\theta_0) + 2\theta_0(\theta - \theta_0) \quad \text{Ignore higher order}$$
$$= 4 + 2(2)(\theta - 2) \quad (\theta_0 = 2)$$
$$= -4 + 4\theta$$

So this is approximated by a line with slope = 4 and intercept = -4.

Choose $s = -\alpha \left.\frac{dJ}{d\theta}\right|_{\theta=\theta_0}$, $\alpha > 0$

Take $\left.\frac{\partial J}{\partial \theta}\right|_{\theta=\theta_0} = \frac{\partial J}{\partial \theta_0}$

$$J(\theta_0 + s) = J(\theta_0) + \underbrace{\frac{dJ}{d\theta_0}\left(-\alpha \frac{dJ}{d\theta_0}\right)}_{\text{always -ve}}.$$

so $J(\theta_0 + s) < J(\theta_0)$

As this continues, we go to smaller and smaller values and hence we are guaranteed to get a min.

$$J(\theta_0^{k+1}) = J(\theta_0^{k}) - \alpha \left(\left.\frac{dJ}{d\theta}\right|_{\theta=\theta_0^{k}}\right)^2$$

$\alpha = 1$ Eg: Take $k = 0$ $\theta^1 = 2$ $J(\theta) = 4$

$k = 1$ $\theta^2 = -2$ $J(\theta) = 4$

$k = 2$ $\theta^3 = 2$ $J(\theta) = 4$

$\alpha = 1.2$ Eg: Take $k = 0$ $\theta^1 = 2$ $J(\theta) = 4$

$k = 1$ $\theta^2 = -2.8$ $J(\theta) = 7.64$

$k = 2$ $\theta^3 = 3.92$ $J(\theta) = 15.3664$

$\alpha = 0.1$ Eg: Take $k = 0$ $\theta^1 = 2$ $J(\theta) = 4$

$k = 1$ $\theta^2 = 1.6$ $J(\theta) = 2.56$

$k = 2$ $\theta^3 = 1.28$ $J(\theta) = 1.6384$

Here used : $\theta^{k+1} = \theta^k - \alpha \underbrace{\frac{dJ}{d\theta}\Big|_{\theta=\theta^k}}_{= 2\theta^k}$

→ This shows that we must choose α wisely

→ Large α - function diverges

→ small α - function converges but slowly (i.e. takes more number of iteration to converge. i.e. higher training time)

→ Steepest descent choose such α that we get faster convergence

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left[y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \right]^2$$

$$\frac{dJ}{d\theta_0} = \frac{1}{2m} 2 \sum_{i=1}^{m} \left[y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \right] (-1)$$

$$\frac{dJ}{d\theta_1} = \frac{1}{2m} 2 \sum_{i=1}^{m} \left[y^{(i)} - (\theta_0 + \theta_1 x^{(i)}) \right] (-x^{(i)})$$

and so many derivatives go on if we have a lot no. of parameter. Not practically feasible

→ In steepest descent optimisation, we take α^k (this is α at k^{th} iteration). Choose α such that $J(\theta^0 - \alpha \nabla J(\theta))$ is minimized over α. ($\nabla J(\theta)$ evaluated at $\theta = \theta^0$ next value of θ^0 i.e. θ^1 and so on for every successive θ^k..)

→ Choose some initial θ^0 (i.e. θ_0^0 and θ_1^0) and find $\nabla J(\theta)\Big|_{\theta=\theta^0} = \begin{bmatrix} \frac{\partial J}{\partial \theta_0}\Big|_{\theta_0=\theta_0^0} \\ \frac{\partial J}{\partial \theta_1}\Big|_{\theta_1=\theta_1^0} \end{bmatrix}$ and put in $\theta^0 - \alpha \nabla J(\theta)\Big|_{\theta=\theta^0}$ to then get J as a function of α and minimize it over α.

$$\alpha^{k+1} = \alpha^k - \frac{J'(\alpha)}{J''(\alpha)}\Big|_{\alpha=\alpha^k} \quad \text{——— (1)}$$

Note that though here we have iterations for finding α for each iteration to find θ, still this is faster than GD.

This is called Newton's method. (and we do not have any learning rate here)

The expression (1) comes as ...

→ Use Taylor series expression of $J(\alpha)$ ...

$$J(\alpha) = J(\alpha_0) + \frac{d}{d\alpha} J(\alpha)\Big|_{\alpha=\alpha_0} (\alpha - \alpha_0) + \frac{1}{2!} \frac{d^2}{d\alpha^2} J(\alpha)\Big|_{\alpha=\alpha_0} (\alpha - \alpha_0)^2$$

(use 2nd order approximation)

$\frac{dJ(\alpha)}{d\alpha} = 0$ gives minimum

$$J'(\alpha)\Big|_{\alpha=\alpha_0} + \frac{J''(\alpha)}{2}\Big|_{\alpha=\alpha_0} 2(\alpha - \alpha_0) = 0.$$

$$\alpha = \alpha_0 - \frac{J'(\alpha_0)}{J''(\alpha_0)}$$

EQ. When θ has multiple parameter, lets say θ_0 and θ_1, why do we do simultaneous update and not minimize each of θ_0 and θ_1 separately?

In case of $J(\theta) = \theta^2$, we approximated the function with a line because we had just one parameter θ.

Now, if we had something of the form $J(\theta_0, \theta_1, \ldots \theta_n)$, then to approximate this at a point, we would need a hyper plane.

In earlier case, the line was completely defined by its slope, i.e. only one parameter hence changing that one parameter we reach a new point.

But a hyperplane is defined by multiple parameter. so to go from a point θ^k to θ^{k+1}, we need to change all of $\theta_0 \ldots \theta_n$, only then it makes sense of shifting.

Minimizing each of $\theta_0 \ldots \theta_n$ does not make sense in terms of point shifting on hyperplane and hence gives no correct result. So we need to use SIMULTANEOUS UPDATE.

→ Newton's method for single dimensional eqn (i)
α is as follows

$$\alpha^{k+1} = \alpha^k - \left.\frac{f'(\alpha)}{f''(\alpha)}\right|_{\alpha=\alpha^k}$$

→ Newton's method for multi dimensional)
α is as follows :

Let $\alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}$

$$\alpha^{k+1} = \alpha^k - [Hf(\alpha)]^{-1} \nabla f(\alpha)\Big|_{\alpha=\alpha^k}$$

size

(Replace derivative by gradient and reciprocal of double derivative with inverse of Hessian matrix)

$$\nabla f(\alpha) = \begin{bmatrix} \frac{\partial f}{\partial \alpha_0} \\ \frac{\partial f}{\partial \alpha_1} \end{bmatrix}$$

at $\alpha_0 = \alpha_0^k$, $\alpha_1 = \alpha_1^k$

+1)

$$Hf(\alpha) = \begin{bmatrix} \frac{\partial^2 f}{\partial \alpha_0^2} & \frac{\partial^2 f}{\partial \alpha_0 \alpha_1} \\ \frac{\partial^2 f}{\partial \alpha_1 \alpha_0} & \frac{\partial^2 f}{\partial \alpha_1^2} \end{bmatrix}$$

m)

at $\alpha_0 = \alpha_0^k$
$\alpha_1 = \alpha_1^k$

EQ Minimize $x^4 + 2x^2y^2 + y^4$.

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$X^{k+1} = X^k - [Hf(x)]^{-1} \nabla f(x)$$

$$-\nabla f(x) = Hf(x)\,(X^{k+1} - X^k)$$

Assume start point (a, a).

$$\nabla f(x) = \begin{bmatrix} 4x^3 + 4xy^2 \\ 4x^2y + 4y^3 \end{bmatrix}$$

$$Hf(x) = \begin{bmatrix} 12x^2 + 4y^2 & 8xy \\ 8xy & 4x^2 + 12y^2 \end{bmatrix}$$

$$\begin{bmatrix} -8a^3 \\ -8a^3 \end{bmatrix} = \begin{bmatrix} 16a^2 & 8a^2 \\ 8a^2 & 16a^2 \end{bmatrix} \begin{bmatrix} x-a \\ y-a \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{2}{3}\begin{bmatrix} a \\ a \end{bmatrix}$$ is the solution for first iteration

In general for k^{th} iteration it is $\left(\frac{2}{3}\right)^k \begin{bmatrix} a \\ a \end{bmatrix}$

Page No.:
Date:

Polynomial Regression

→ $h_\theta(x)^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)}$ (line ($n=1$ so just 2 parameters))

hyperplane $= [\theta_0 \ \theta_1 \ \theta_2 \ \cdots \ \theta_n] \begin{bmatrix} 1 \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$

$1 \times n+1$ from 1 to m

$= \theta^T x^{(i)}$ (For linear regression)

$n+1 \times 1$

→ Now, for polynomial regression ....

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 \sqrt{x} + \cdots$

↓

If x is size, $h_\theta(x)$ is price depending on size

→ Here also, we aim to find the θs.

→ Dimension of X matrix in general is $m \times (n+1)$

$$X = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^n \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^n \\ & & & & \\ & & & & \\ 1 & x_m & (x_m)^2 & \cdots & (x_m)^n \end{bmatrix} \quad m \times (n+1)$$

(Fit a model through data - fitting problem)

Logistic regression

→ This is a supervised learning algorithm where $\{x^{(i)}, y^{(i)}\}_{i=1 \text{ to } m}$ are given.

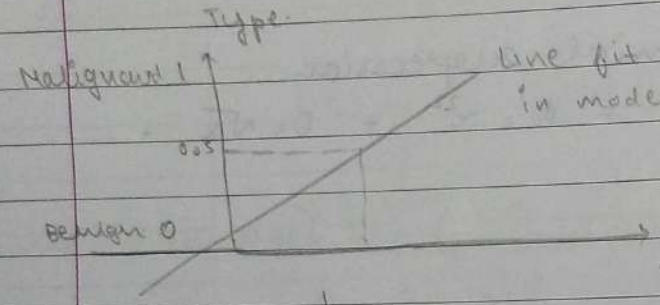
→ This is a binary classifier so $y^{(i)}$ takes only 2 values 0 or 1

Date:

→ $x^{(i)}$ could be scalar or vector.

→ Thus unlike linear regression, where we could have output from $-\infty$ to ∞, here y is not a real number. It takes value from a fix set. Thus this is discrete whereas linear regression is continuous.

Eg. classify tumour as maliguant or benign.



$y^{(i)} = 1$: x Maliguant

$y^{(i)} = 0$: o Benign

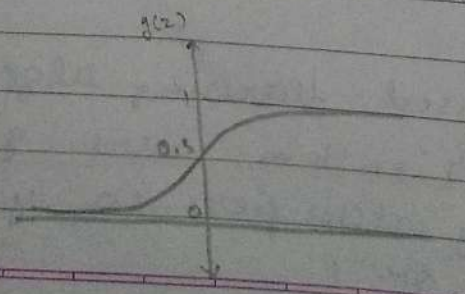
Any point on right of model line denotes maliguant tumour while on left of it denotes benign. (Fixed y as the midway point between 0 and 1).

→ We can use linear regression to plot this model fit line, but not a good idea as we do not need multiple possible outputs here. So the model is modified.

$$\underset{\text{hypothesis}}{h_\theta(x)} = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

$(n+1) \times 1 \quad 1 \times (n+1)$

(must be a scalar value) Here $n = 1$



→ Let us consider $h_\theta(x)$ as the estimated probability that $y = 1$ on input x.

→ for 1 example ... $x = \begin{bmatrix} 1 \\ x_1 \end{bmatrix}$ → Tumor size (test example)

→ Eg $h_\theta(x) = 0.7$ — This means patient has maligmant cancer with probability 0.7 and has benign cancer with probability 0.3

→ We know that $y = 0$ or 1. So

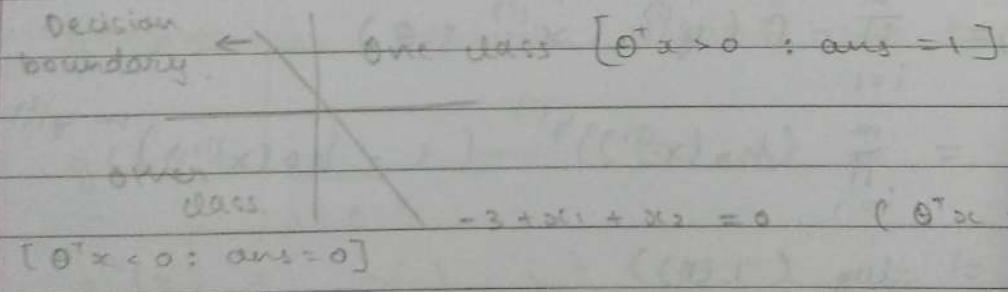
$P(y=1 \mid x; \theta) = 1 - P(y=0 \mid x; \theta)$

$h_\theta(x)$

→ $h_\theta(x) = g(\underbrace{\theta_0 + \theta_1 x_1 + \theta_2 x_2}_{\theta^T x})$

Suppose we know $\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$.

So $-3 + x_1 + x_2 \geq 0$ → belongs to one class (like eg maligmant)

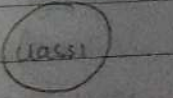


→ Note that here y takes only one of the 2 values : 0 or 1. Hence y is a Bernoulli Random Variable.

→ If $\theta_0 = 0$, the decision boundary passes through (0, 0). (y intercept = 0)

→ If $\theta^T x$ turns up to be $= 0$, toss a coin and decide what to classify it as

→ Decision boundary may not always be linear.



→ Here we had supposed $\theta_0, \theta_1, \theta_2$. Now we will see how to estimate these values actually. We will use maximum likelihood estimation

→ Assume that $P(Y=1 \mid x; \theta) = h_\theta(x)$

$P(Y=0 \mid x; \theta) = 1 - h_\theta(x)$

→ This means $P(Y=y \mid x; \theta) = (h_\theta(x))^y (1-h_\theta(x))^{1-y}$

→ $L(\theta) = P(y^{(1)}, \ldots, y^{(m)} \mid x^{(1)}, \ldots, x^{(m)}; \theta)$

→ $\theta^* = \max_{\forall\theta} L(\theta) = \max_{\forall\theta} P(y^{(1)}, \ldots, y^{(m)} \mid x^{(1)} \ldots x^{(m)}; \theta)$

Here $y_1, \ldots y_m$ are independent of each other once $x_1, \ldots, x_m$ are given

→ $\theta^* = \max_{\forall\theta} P(y^{(1)} \mid x^{(1)}; \theta) \ldots P(y^{(m)} \mid x^{(m)}; \theta)$

→ $L(\theta) = \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)}; \theta)$ m training examples

$= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$

→ $l(\theta) = \log(L(\theta))$

$= \sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))$

Note: When there are probability terms in likelihood function, its better to use log because since probability is between 0 and 1, multiplying m such values gives a very small number, and hence difficult to handle by computer. So take log and convert multiply into addition.

→ Maximization can be done using gradient ascent

→ We instead take a minus sign and minimize only.

→ $\theta^* = \min_{\forall\theta} -\left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + \sum_{i=1}^{m} (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right]$

↓

This is called binary cross entropy

↓

$H = \sum -p_i \log p_i$

(max when $p_i = 0.5$)

→ It's called cross entropy because here we are using 2 probabilities : one actual $y^{(i)}$ and other estimated of $h_\theta(x^{(i)})$.

→ Entropy basically means randomness. Hence, we are essentially trying to minimize this randomness between actual and expected output.

→ $g(z) = \frac{1}{1+e^{-z}}$ — (A)

→ $g'(z) = -\left(\frac{1}{1+e^{-z}}\right)^2 (-e^{-z}) = \left(\frac{1}{1+e^{-z}}\right)^2 (e^{-z})$

$g'(z) = \left(\frac{1}{1+e^{-z}}\right)\left(1 - \frac{1}{1+e^{-z}}\right)$

$= g(z)\,(1 - g(z))$

Now $g'(\theta) = g(z)\,(1-g(z))\,\frac{dz}{d\theta}$

→ We use gradient descent to find θs

$$\theta_0^{k+1} = \theta_0^k - \alpha \frac{\partial}{\partial \theta_0} l(\theta) \Big|_{\theta = \theta^k}$$

$$\theta_1^{k+1} = \theta_1^k - \alpha \frac{\partial}{\partial \theta_1} l(\theta) \Big|_{\theta = \theta^k}$$

∴ so on till θ_n^{k+1}.

(simultaneous update is important because here we will have a hyperplane. Each point on hyperplane is defined by all parameters. So to go from one point to other, we need to change all parameters at a time).

If we use just one training example instead of $\sum_{i=1}^{m}$, we come to stochastic gradient descent.

$$l(\theta) = -\left[y \log(h_\theta(x)) + (1-y) \log(1 - h_\theta(x)) \right]$$

$$\frac{\partial}{\partial \theta_j} l(\theta) = -\left[y \frac{1}{h_\theta(x)} - (1-y) \frac{1}{1-h_\theta(x)} \right] h_\theta(x)(1-h_\theta(x)) \frac{\partial}{\partial \theta_j}(\theta^T x)$$

(from (A))

$$= x_j$$

$$\frac{\partial}{\partial \theta_j} l(\theta) = -(y(1-h_\theta(x)) - (1-y) h_\theta(x)) x_j$$

$$= -(y - y h_\theta(x) - h_\theta(x) + y h_\theta(x)) x_j$$

$$= -(y - h_\theta(x)) x_j$$

$$\rightarrow \theta_0^{k+1} = \theta_0^k + \alpha \sum [(y^{(i)} - h_\theta(x^{(i)})) \mid (1)]$$

x_j for θ_0

at $\theta = \theta^k$

Here we evaluate at $\theta = \theta^k$ and not $\theta = \theta_0^k$ because $h_\theta(x^{(i)})$ depends on all $\theta_0, \theta_1, \ldots$ as it is $\frac{1}{1+e^{-\theta^T x}}$

$$\theta_1^{k+1} = \theta_1^k + \alpha\left[\left(y^{(i)} - h_\theta(x^{(i)})\right) x_1^{(i)}\right]$$

! and so on $\theta_j^{k+1} = \theta_j^k + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right) x_j$

→ This was all about binary classification. Now we use multiclass classification

Multiclass classification

Eg. Consider 3 classes : 1, 2, 3 ...

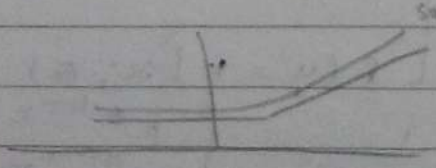
Combination
1 2,3 $h_{\theta_1}(x) = 0.9$ → highest so classified as class 1
2 1 3 $h_{\theta_2}(x) = 0.5$
3 1,2 $h_{\theta_3}(x) = 0.8$

This is essentially binary classification of one Vs all

→ Sometimes we also use softmax classifier

$\text{Max}(R_1, \ldots, R_n) : R^n \to R$ (Not differentiable)

Softmax → differentiable version of Max



$\max(x, 5)$

$\text{softmax}(x, 5) = \log(e^x + e^5)$

$\text{Softmax}(x_1, x_2) = \log(e^{x_1} + e^{x_2})$

(5)

$$\frac{ds}{dx_1} = \frac{e^{x_1}}{e^{x_1} + e^{x_2}} \qquad \frac{ds}{dx_2} = \frac{e^{x_2}}{e^{x_1} + e^{x_2}}$$

→ Addition of all these = 1 hence we can use this as probability.

→ softmax is also called multinomial logistic regression / generalised logistic regression

→ In logistic regression $y_i \in \{0, 1\}$. We learn $n+1$ parameters

→ For softmax classification, y can take k different values (k = no. of classes).

→ Eg: MNIST data set, $k = 10$ to recognise 10 digits. Each digit is an image of size $28 \times 28 = 784$. Thus we have θ_0 to θ_{784}, i.e. 785 parameters.

→ Given a test example x, we want our hypothesis to estimate $P(y = k \mid x)$ for each $k = 1, 2, \ldots k$ and classify k as the one with highest probabilities.

→ Hypothesis has to be such that sum of probabilities = 1.

→ Hence $h_\theta(x) = \begin{bmatrix} P(y=1 \mid x; \theta) \\ P(y=2 \mid x; \theta) \\ \vdots \\ P(y=k \mid x; \theta) \end{bmatrix}$

$$= \frac{1}{\sum_{j=1}^{k} e^{\theta^{(j)T} x}} \begin{bmatrix} e^{\theta^{(1)T} x} \\ e^{\theta^{(2)T} x} \\ \vdots \\ e^{\theta^{(k)T} x} \end{bmatrix}$$

→ $P(y=1 \mid x; \theta) = \frac{e^{\theta^{(1)T} x}}{e^{\theta^{(1)T} x} + e^{\theta^{(2)T} x}}$ (considering only 2 classes)

Here $\theta^{(1)T} = [\theta_0^{(1)} \quad \dots \quad \theta_n^{(1)}]$

$$x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

→ For digit classification, $K = 10$ and 785 parameters for each class. so we need to learn 7850 θs.

→ $\theta^{(1)}, \theta^{(2)}, \dots \theta^{(K)} \in R^{n+1}$

→ $\theta = (n+1) \; [\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}]$

→ For binary classification, we learnt $n+1$ parameters. Here $K = 2$. So we learnt $(n+1)(K-1)$ number of parameters (so actually for digit classification we'll need (785)(9) parameters only).

→ $l(\theta) = \sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1 - h_\theta(x^{(i)}))$

we need to maximize this.

* → $l(\theta) = \sum_{i=1}^{m} \sum_{k=1}^{K} I\{y^{(i)} = k\} \log \frac{e^{\theta^{(k)T} x^{(i)}}}{\sum_{j=1}^{K} e^{\theta^{(j)T} x^{(i)}}}$

differentiate!

→ We'll maximize this function using gradient descent to chose those 7850 Θs

After training, output 1000000000 when 0 is applied, 0100000000 when 1 is applied and so on i.e. only one value goes high.

→ Logistic regression - supervised learning (classification)

→ P(Y|X) - we model this as $h_\theta(x)$

Linear Discriminant Analysis (LDA)

→ Also called density estimation technique

→ Here we estimate $P(X = x \mid Y = 0)$ and $P(X = x \mid Y = 1)$ from x_i with $i = 1$ to m (given set of examples)

→ Considering a K class problem - LDA

$$P(Y = k \mid X = x) = \frac{P(X = x \mid Y = k)\, P(Y = k)}{P(X = x)}$$

$P(X = x \mid Y = k)$ - class conditional probability
$P(Y = k)$ - prior probability
$P(X = x)$ - marginal probability.

Considers examples of only one class $Y = k$

→ In logistic regression, we were directly modelling $P(Y = k \mid X = x)$, but here we are modelling the data $P(X = x \mid Y = k)$ itself as a gaussian.

Principle component Analysis –
Unsupervised learning – used for clustering

Page No.:
Date:
YOUVA

→ $P(X = x) = P(X = x, y = 0) + P(X = x, y = 1)$
(similar to $f(x) = \int f(x, y)\, dy$)

→ $P(X = x) = P(X = x \mid y = 0)\, P(y = 0) + P(X = x \mid y = 1)\, P(y = 1)$.
(for a binary class)

→ $f_k(x)$ – class conditional probability for k^{th} class

→ Π_k – prior probability of k^{th} class.

(K – Total number of classes
k – a particular class)

→ generalising for K class classification..

$$P(y = k \mid X = x) = \frac{f_k(x)\, \Pi_k}{\sum_{k=1}^{K} f_k(x)\, \Pi_k}$$

→ On decision boundary, we have
$P(Y = 1, X = x) = P(Y = 2 \mid X = x)$ if $K = 1, 2$.
(binary classification)

→ $f_0(x) = P(X = x \mid Y = 0)$
$f_1(x) = P(X = x \mid Y = 1)$.
We model using multivariate gaussian with a covariance matrix.

→ $$f_X(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}}\, e^{-(x - m_x)^2 / 2\sigma_x^2}$$

→ $$f_{x_1, x_2}(x_1, x_2) = \frac{1}{(2\pi)^{2/2} |\Sigma|^2}\, e^{-\frac{1}{2}\left[\underset{1\times 2}{(X - M)}\, \underset{2\times 2}{\Sigma^{-1}}\, \underset{2\times 1}{(X - M)^T}\right]}$$

$X = [x_1 \;\; x_2]$ $\quad X - M = [x_1 - m_{x_1} \;\; x_2 - m_{x_2}]$
$M = [m_{x_1} \;\; m_{x_2}]$

→ Σ is the covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_{x_1}^2 & cov(x_1, x_2) \\ cov(x_2, x_1) & \sigma_{x_2}^2 \end{bmatrix}$$ — symmetric

with $cov(X_1, X_2) = E[(X_1 - m_{X_1})(X_2 - m_{X_2})]$
(A single value)

→ Consider digit classification problem, so here 10 classes : $k = 0$ to 9. Take an image and make a 28×28 grid. Put all rows vertically to have a column with 784 entries. These are $x_1, \ldots, x_{784}$. So now we can calculate joint density given we know Σ matrix.

$$f_{x_1, \ldots, x_{784}}(x_1, \ldots x_{784}) = \frac{1}{(2\pi)^{\frac{784}{2}} |\Sigma|^{1/2}} e^{-\frac{1}{2}[(X-M)\Sigma^{-1}(X-M)^T]}$$

where $X = [x_1 \quad \ldots \quad x_{784}]$

$M = [m_{x_1} \quad \ldots \quad m_{x_{784}}]$

→ Suppose we take just digits 0 and 1, and suppose we have 50 zeroes written (the greater than 50, the better learning)

X_1	X_2		X_{784}
x_1	x_2'		x_{784}'
x_1^2	x_2^2		x_{784}^2
!	!		!
!	!		!
x_1^{50}	x_2^{50}		x_{784}^{50}

↓

The first pixel value from all the images. This gives us a random variable. This way, all pixel values (pixel 1 to 784.) are individually an RV

Date:

$$\Sigma = \begin{bmatrix} \sigma_{x_1}^2 & cov(x_1, x_2) & \cdots & cov(x_1, x_{784}) \\ \vdots & \ddots & & \\ cov(x_{784}, x_1) & \cdots & & \sigma_{x_{784}}^2 \end{bmatrix}_{784 \times 784}$$

Note $\sigma_{x_1}^2 = E[(x_1 - m_{x_1})^2]$ with m_{x_1} as the average of first column.

→ To find the decision boundary, we use the equiprobability $P(Y=1 \mid X=x) = P(Y=0 \mid X=x)$

→ $f_K(\underline{x}) = P(X=x \mid Y=K)$

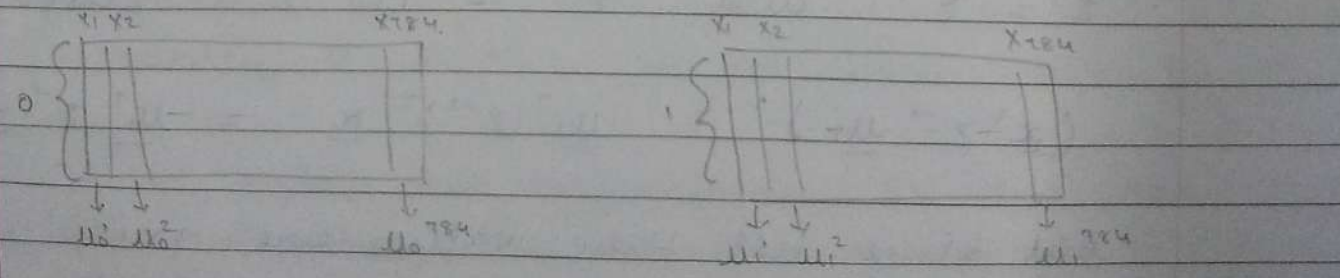
$$= \frac{1}{(2\pi)^{n/2} |\Sigma_K|^{1/2}} e^{-\frac{1}{2}[(\underline{x} - \underline{\mu_K})^T \Sigma^{-1} (\underline{x} - \underline{\mu_K})]}$$

(Σ^{-1}: 784×784; $(\underline{x} - \underline{\mu_K})$: 784×1)

→ Considering binary classification, for decision boundary...

$$\frac{f_0(x)\,\pi_0}{\sum_{K=0}^{1} f_K(x)\,\pi_K} = \frac{f_1(x)\,\pi_1}{\sum_{K=0}^{1} f_K(x)\,\pi_K}$$

→ Let μ_0 represent the mean vector of class 0 and Σ_0 represent the covariance matrix of class 0



(Mean of entire column)

$$f_1(x)\,\pi_1 = f_0(x)\,\pi_0$$

$$\frac{1}{(2\pi)^{n/2} |\Sigma_1|^{1/2}} e^{-\frac{1}{2}[(\underline{x} - \underline{\mu_1})^T \Sigma_1^{-1} (\underline{x} - \underline{\mu_1})]} \pi_1 = \frac{1}{(2\pi)^{n/2} |\Sigma_0|^{1/2}} e^{-\frac{1}{2}[(\underline{x} - \underline{\mu_0})^T \Sigma_0^{-1} (\underline{x} - \underline{\mu_0})]} \pi_0$$

Here 784 features, so decision boundary is a hyperplane

→ We assume $\Sigma_0 = \Sigma_1 = \Sigma$ (not the case always).

→ Using this and taking log both side.

$$-\frac{1}{2}\left[(\underline{x}-\underline{\mu_1})^T \Sigma^{-1} (\underline{x}-\underline{\mu_1})\right] + \log(\pi_1) = -\frac{1}{2}\left[(\underline{x}-\underline{\mu_0})^T \Sigma^{-1} (\underline{x}-\underline{\mu_0})\right] + \log(\pi_0)$$

Decision boundary is of the form $\beta^T \underline{x} = 0$

$$-\frac{1}{2}\left[(\underline{x}-\underline{\mu_1})^T \Sigma^{-1} (\underline{x}-\underline{\mu_1})\right] + \frac{1}{2}\left[(\underline{x}-\underline{\mu_0})^T \Sigma^{-1} (\underline{x}-\underline{\mu_0})\right] + \log\left(\frac{\pi_1}{\pi_0}\right) = 0$$

Equal so can be combined

$$\boxed{-\frac{1}{2}\underline{x}^T\Sigma^{-1}\underline{x}} + \frac{1}{2}\underline{x}^T\Sigma^{-1}\underline{\mu_1} + \frac{1}{2}\underline{\mu_1}^T\Sigma^{-1}\underline{x} - \frac{1}{2}\underline{\mu_1}^T\Sigma^{-1}\underline{\mu_1}$$

$$\boxed{+\frac{1}{2}\underline{x}^T\Sigma^{-1}\underline{x}} - \frac{1}{2}\underline{x}^T\Sigma^{-1}\underline{\mu_0} - \frac{1}{2}\underline{\mu_0}^T\Sigma^{-1}\underline{x} + \frac{1}{2}\underline{\mu_0}^T\Sigma^{-1}\underline{\mu_0}$$

$$+ \log\left(\frac{\pi_1}{\pi_0}\right) = 0$$

Each term is a scalar here.

Covariance matrix is symmetric, so $\Sigma = \Sigma^T$ and $\Sigma^{-1} = (\Sigma^{-1})^T$ (using $\Sigma\Sigma^{-1} = I$)

$$(\underline{x}^T\Sigma^{-1}\underline{\mu_1})^T = \underline{\mu_1}^T(\Sigma^{-1})^T\underline{x} = \underline{\mu_1}^T\Sigma^{-1}\underline{x}$$

But since all these values are 1×1, we get $(\underline{x}^T\Sigma^{-1}\underline{\mu_1})^T = \underline{x}^T\Sigma^{-1}\underline{\mu_1} = \underline{\mu_1}^T\Sigma^{-1}\underline{x}$

$$\underline{x}^T\Sigma^{-1}\underline{\mu_1} - \frac{1}{2}\underline{\mu_1}^T\Sigma^{-1}\underline{\mu_1} - \underline{x}^T\Sigma^{-1}\underline{\mu_0} + \frac{1}{2}\underline{\mu_0}^T\Sigma^{-1}\underline{\mu_0} + \log(\pi_1/\pi_0) = 0.$$

(Assumption $\Sigma_1 = \Sigma_0$ works in linear case as there will be no square terms here. Case is different in quadratic discriminant analysis).

$$x^T \Sigma^{-1} (\mu_1 - \mu_0) + \frac{1}{2}\left(\mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1\right) + \log\left(\frac{\pi_1}{\pi_0}\right) = 0$$

→ Now this is in the form of decision boundary $x^T \beta + b = 0$.

→ Considering both classes as equiprobable, $\pi_1 = \pi_0$, so $\log(\pi_1/\pi_0) = 0$.

→ All parameters μ_0, μ_1, Σ are given. Now we can use the decision boundary to test a new example.

$$DB: \quad x^T \Sigma^{-1} (\mu_1 - \mu_0) + \frac{1}{2}(\mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1) = 0$$

→ For testing, we some x and based on this expression. >0 or <0 we classify

→ When we consider $\Sigma_0 = \Sigma_1$, the decision boundary is linear. (Hyperplane when multiple parameters)

→ Usually $\Sigma_1 \neq \Sigma_0$. So we take $\Sigma = \frac{\Sigma_0 + \Sigma_1}{2}$

→ If we relax the assumption Σ_k are not identical, then the classifier results in Quadratic Discriminant Analysis (QDA)

→ $$x^T \Sigma x = (x_1 \; x_2) \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= a x_1^2 + c x_2^2 + 2b x_1 x_2$$

When Σ_k are assumed to be I, LDA becomes Naïve Based classification.

$$f(x_1, x_2) = \frac{1}{(2\pi)^{2/2} (1)} e^{-\frac{1}{2}[(X-M)^T I^{-1} (X-M)]}$$

$$(X-M)^T = [x_1 - m_{x_1}, \quad x_2 - m_{x_2}$$

$$(X-M) = \begin{bmatrix} x_1 - m_{x1} \\ x_2 - m_{x2} \end{bmatrix}$$

$$f(x_1, x_2) = \frac{1}{\sqrt{2\pi}\sqrt{2\pi}} e^{-\frac{1}{2}(x_1 - m_{x1})^2} e^{-\frac{1}{2}(x_2 - m_{x2})^2}$$

$$= f(x_1) f(x_2)$$

→ Hence independent. Thus for Gaussian RV only, we can say that uncorrelatedness implies independence. (Uncorrelated because $\Sigma_k = I$) The reverse independence → uncorrelated is true for any RV.

→ Thus for Σ_k as I;
$f(x_1 \dots x_n | Y = k) = f(x_1 | Y = k) \dots f(x_n | Y = k)$

→ So all $x_1, \dots x_n$ are independent

→ For Naïve Bayes classifier, the features of a class are assumed independent (which is not true in practice)

→ LDA - for multiclass classification. $k = 1$ to K where K is the number of classes.

→ Compute $f_k(x)\, \pi_k$ for all classes, given some x, classify as that digit which gives max value for $f_k(x)\, \pi_k$. (Note for finding $f_k(x)\, \pi_k$, we have k covariance matrix and mean vector)

→ $f_k(x)\,\pi_k = \frac{1}{(2\pi)^{n/2}\,|\Sigma_k|^{1/2}}\, e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}\,\pi_k$

→ We can equivalently check $\max(\log(f_k(x)\,\pi_k))$

$$\delta_k = \log(f_k(x)\,\pi_k) = \log(|\Sigma_k|^{-1/2}) - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) + \log \pi_k$$

μ_k - average image — Pixel 1 average of all pixel 1 in training example and so on... (each pixel is a RV)

→ So given a test example $\underline{x}$, compute δ_k for all $k = 1$ to K and choose the class with maximum δ_k.

→ Taking $\Sigma_k = I$

$$\delta_k = -\frac{1}{2}\underbrace{(\underline{x}-\underline{\mu_k})^T(\underline{x}-\underline{\mu_k})}_{\downarrow} + \underbrace{\log(\pi_k)}_{\downarrow}$$

Can ignore this term if we take equiprobable classes

So in a way, the distance between given example and average image is used for classifying. Minimum distance gives maximum δ_k.

Here we have erroneously modelled the RV as gaussian, however its not actually so when we plot and check. It works accurately when the values are drawn from actual gaussian distribution

→ Σ_k - covariance matrix of k^{th} class, so its symmetric

→ If Σ_k is not identity, we write it using matrix diagonalisation

$$\Sigma_k = \underset{n\times n}{U} \; \underset{n\times n}{S} \; U^{-1} \qquad (U \text{ is symmetric})$$

$$\underset{n\times n}{U} = [u_1 \dots u_n] \qquad \underset{n\times n}{S} = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_n \end{bmatrix}$$

Note : $S^T = S$

$S^{-1} = S^{-1/2} S^{-1/2}$

Here $u_1 \dots u_n$ each is a $n \times 1$ vector (our eigenvectors) corresponding to eigenvalues $\lambda_1 \dots \lambda_n$

Here $\underset{\downarrow}{U^{-1}} = U^T$ (Unitary matrix)

Matrix can be diagonalised if the eigenvectors are independent because only then inverse is possible.

$$\Sigma_k^{-1} = (U S U^{-1})^{-1} = U S^{-1} U^{-1} = U S^{-1} U$$

$$\delta_k = -\frac{1}{2} (\underline{x} - \underline{\mu_k})^T \Sigma_k^{-1} (\underline{x} - \underline{\mu_k}) + \log(\pi_k)$$

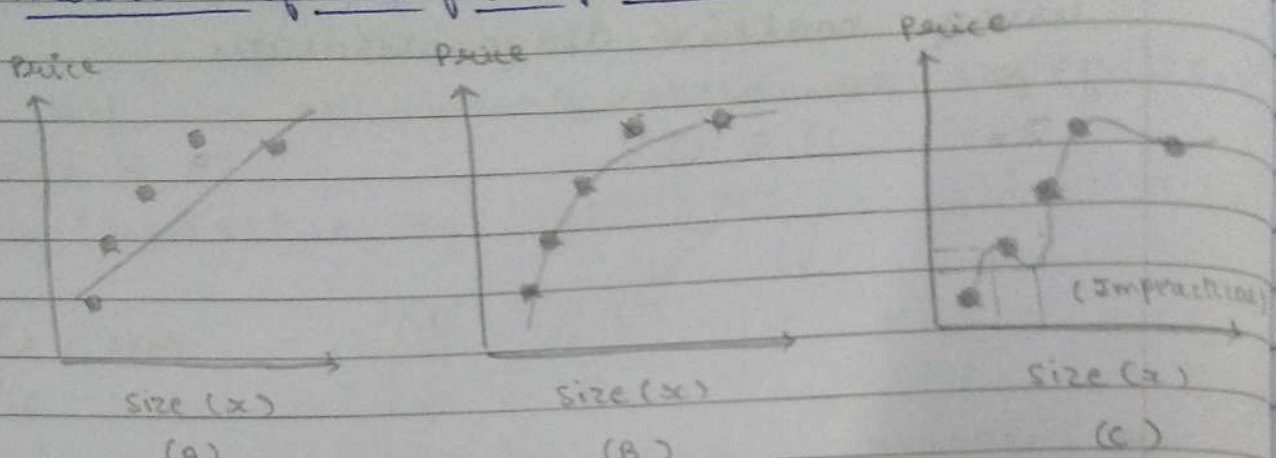
$$= -\frac{1}{2} (x - \mu_k)^T U S^{-1} U^T (\underline{x} - \underline{\mu_k}) + \log(\pi_k)$$

$$= -\frac{1}{2} (U^T \underline{x} - U^T \underline{\mu_k})^T S^{-1} (U^T \underline{x} - U^T \underline{\mu_k}) + \log(\pi_k)$$

$$= -\frac{1}{2} (S^{-1/2} U^T \underline{x} - S^{-1/2} U^T \underline{\mu_k})^T (S^{-1/2} U^T \underline{x} - S^{-1/2} U^T \underline{\mu_k}) + \log \pi_k$$

Even here we take difference between given and averages, but after taking a linear transformation with $S^{-1/2} U^T$

Problem of overfitting and regularization



A — $h_\theta(x) = \theta_0 + \theta_1 x$ (high bias)
[underfit]

B — $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
[Just right fit]

C — $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
(high variance) [Overfit]

→ Here, given data points are $y^{(i)}$
→ The fitted function is $\hat{f}(x^{(i)})$ with $i = 1$ to m
→ Regularisation takes care of choosing the right $h_\theta(x)$. This is done by chosing the correct λ in

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))^2 + \lambda \|\theta\|^2$$

Problem of overfitting — If we have too many features the learned hypothesis may fit very well to training data but fails to generalize to new example.

→ This concept applies to regression and classification both.

→ For every problem, we need to estimate the true function which gave the training data points

→ θ – true value — unknown
$\hat{\theta}$ – Random value

→ If $E[\hat{\theta}] = \theta$, then the estimator is unbiased.

→ If $-E[\hat{\theta}] + \theta \neq 0$, then estimator is biased.
bias value

→ $var(\hat{\theta}) = E[\hat{\theta} - E[\hat{\theta}]]^2$ (variance of estimator)

→ Ideal estimator has bias = 0 and variance = 0 (Not practically possible)

→ (bias)^2 + variance = mean squared error between true and estimated function

→ $(-E[\hat{\theta}] + \theta)^2 + E[\hat{\theta} - E[\hat{\theta}]]^2$
$= \theta^2 + (E[\hat{\theta}])^2 - 2\theta E[\hat{\theta}] + E[\hat{\theta}^2] - (E[\hat{\theta}])^2$
$= \theta^2 - 2\theta E[\hat{\theta}] + E[\hat{\theta}^2]$ — A

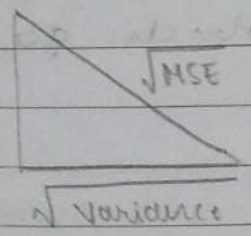
→ Also $E[(\theta - \hat{\theta})^2] = E[\theta^2] + E[\hat{\theta}^2] - 2E[\theta\hat{\theta}]$
$= \theta^2 + E[\hat{\theta}^2] - 2\theta E[\hat{\theta}]$
Hence A = B — B

Regularisation

→ In ML, the task is to estimate a function. This is done based on available data and

making assumptions on true function

→ We are choosing a function from a class (estimated function)



bias ∝ 1/variance (MSE constant)

→ Given set of examples: $T = \{x_i, y_i\}_{i=1}^{n}$

observed value (RV depending on RV ϵ_i)

→ $y_i = f(x_i) + \epsilon_i$ Here ϵ_i is assumed to be Gaussian $N(0, \sigma^2)$

$\hat{y}_i = \hat{f}(x_i) = h_\theta(x) = \theta_0 + \theta_1 x \ldots$

(a class of different functions)

→ We can reduce the empirical error i.e. $E[\hat{y}_i - \hat{y}_i]^2$ (for a finite set of values)

→ We cannot actually calculate the true error i.e. $E[\hat{f}_i - f_i]^2 = E[-\beta_i + \hat{y}_i]^2$ (for infinite set of values, i.e. the whole function)

→ Bias: $f - E[\hat{f}]$

variance: $E[\hat{f} - E[\hat{f}]]$

MSE: $E(f - \hat{f})^2$

→ High bias means estimated function is not able to cope up with actual function

→ Consider MSE between data points and fitted function for a point (x_0, y_0)

$E(\hat{y}_0 - y_0)^2 = E[\hat{f}_0 - f - \epsilon_0]^2$

$$E[\hat{y_0} - y_0]^2 = E[(\hat{f_0} - f_0)^2] + E[\epsilon_0^2] - 2E[\epsilon_0(\hat{f_0} - f_0)]$$

$$\text{Var}(\epsilon_0^2) = E[\epsilon_0^2] - (E[\epsilon_0])^2$$
$$\sigma^2 = E[\epsilon_0]^2 - 0$$
$$\sigma^2 = E[\epsilon_0]^2$$

$$E[(\hat{y_0} - y_0)^2] = E[(\hat{f_0} - f_0)^2] + \sigma^2 - 2E[(y_0 - f_0)(\hat{f_0} - f_0)]$$

(estimated error) (true error)

This is kind of covariance between y_0 and $\hat{f_0}$. Now, this $\hat{f_0}$ is obtained from the training data. So, if we choose a testing example (x_0, y_0) not from the training set, then the covariance will be zero.

$y_0 = f(x_0) + \epsilon_0$

true $\quad N(0, \sigma^2)$

Using this and assuming a $\hat{f}$ (ho(x)), we arrive at $\hat{f}(x_0) = \hat{y_0}$

→ Considering n training points, we can write ...

$$\sum_{i=1}^{n} (y_i - \hat{y_i})^2 = \sum_{i=1}^{n} (f_i - \hat{f_i})^2 + n\sigma^2$$

(Empirical) (True) (noise)

→ Empirical error in a way represents the true error (except the noise) provided we use a point outside the training set

Page No.:
Date:
YOUVA

for 'validation' (cross validation test)

→ If our fitted function works well for these new points outside training set, then the true error will be minimized

→ Thus empirical error is a good estimate of true error, when considering a point outside training set. This is the justification of using cross validation test.

→ This is not the case when we consider a point in the training set. i.e. (x_0, y_0) belongs to training set

→ Then $2E[\epsilon_0 (\hat{f_0} - f_0)] \neq 0$

★ → Consider Stein's Lemma:

If x is a R.V. $N(\theta, \sigma^2)$ and $g(x)$ is a differentiable function, then

$$E[g(x)(x-\theta)] = \sigma^2 E\left(\frac{\partial}{\partial x} g(x)\right)$$

$\epsilon_0 \rightarrow N(0, \sigma^2)$

$x = N(\theta, \sigma^2)$

$x - \theta = N(0, \sigma^2) = \epsilon_0$ (mean subtracted)

$E[g(x)(x-\theta)]$ compared with $E[\epsilon_0 (\hat{f_0} - f_0)]$ we get $g(x) = \hat{f_0} - f_0 = \hat{y_0} - f_0$ (so this is actually a function of ϵ_0 because $y_0 = x_0 + \epsilon_0$ and $\hat{y_0}$ is obtained from y_0)

$$E[\epsilon_0 (\hat{f_0} - f_0)] = \sigma^2 E\left(\frac{\partial}{\partial \epsilon_0} (\hat{f_0} - f_0)\right)$$

(constant so derivative zero)

$$= \sigma^2 E\left(\frac{d\hat{f_0}}{d\epsilon_0}\right)$$

$$\frac{d\hat{f}_0}{d\epsilon_0} = \frac{d\hat{f}_0}{dy_0}\frac{dy_0}{d\epsilon_0} = \frac{d\hat{f}_0}{dy_0} \qquad (y_0 = x_0 + \epsilon_0)$$

$$E[\epsilon_0(\hat{f}_0 - f_0)] = \sigma^2 E\left(\frac{d\hat{f}_0}{dy_0}\right)$$

↓

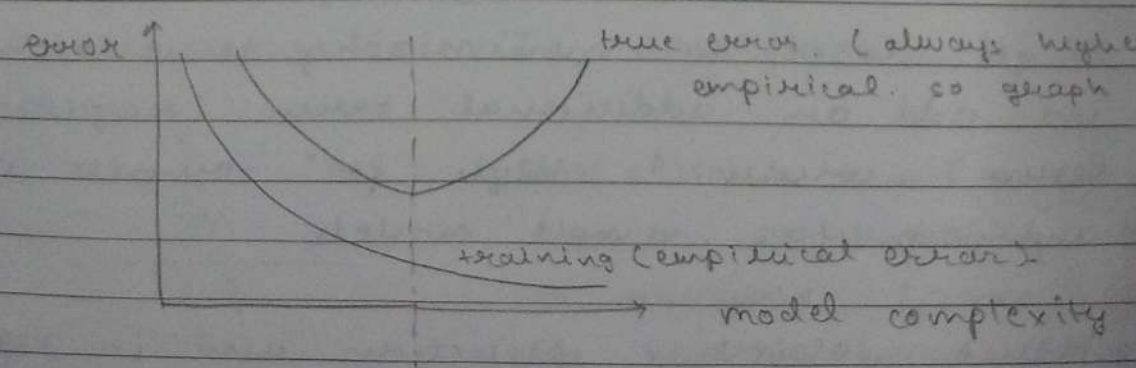
This term represents complexity of model; small when complexity is small large when complexity is high

→ $\frac{d\hat{f}_0}{dy_0}$ is high when small perturbation in y_0 leads to large change in $\hat{f}_0$, hence a complex model.

→ For a simple model, $\hat{f}_0$ does not change a lot with small perturbation in y_0

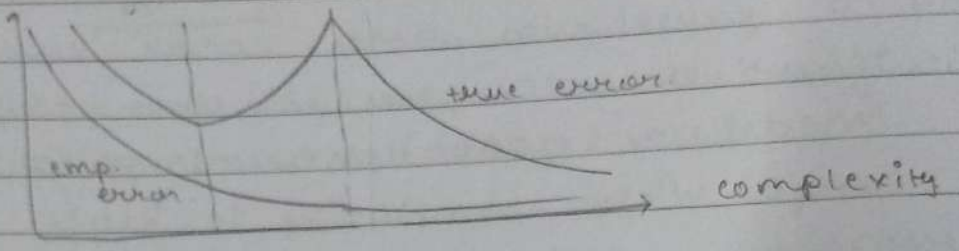
$$\text{True error} = \text{Empirical error} - n\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} D_i$$

→ For a complex model, overfitting occurs and hence true error will increase



Page No.:
Date:

→ Neural networks are extremely complex models (classical ML theory). There the graph of error can changes when complexity is increased tremendously.



→ When the points are in the training set, empirical error is not a good estimate of true error. Its always less than true error.

→ so to get true error, we add a term that takes care of model complexity.

→ If model complexity is less, true error is less; if mode complexity is more, true error is high.

→ That is the reason for overfitting. This happens because we are minimizing empirical error while we should have been minimizing true error (which we can't).

→ To get closer to minimizing true error, we add an additional term (regularisation term) which is high for complex model and low for simple models.

→ Hence, objective function used in regularisation

$I(\theta, x, y) + \Omega(\theta)$

(empirical error term) (Regularisation term)

Anima Anandkumar
Prof at Caltech.

Page No.:
Date:
YOUVA

Eq. $\sum \left(y^{(i)} - h_\theta(x^{(i)})\right) + \lambda \|\theta\|^2$

↓ empirical error ↓ regularisation

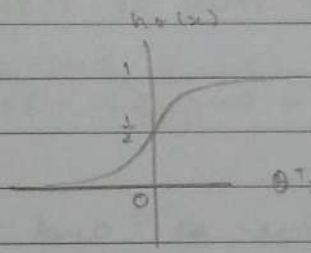
Support Vector Machine (SVM).

→ Supervised learning.
→ Binary linear classifier. (SVM - hyperplane)

→ In logistic regression;

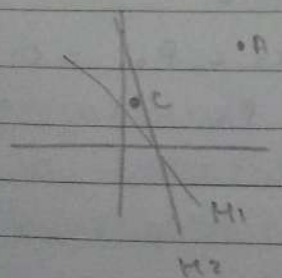
$$P(y=1 \mid x, \theta) = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

$$\underline{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \underline{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$



$\theta^T x > 0 \rightarrow y = 1$
$\theta^T x < 0 \rightarrow y = 0$

→ For better accuracy, we train the model such that $\theta^T x >>> 0$ when $y = 1$ and $\theta^T x <<< 0$ when $y = 0$. This gives more confidence in classification



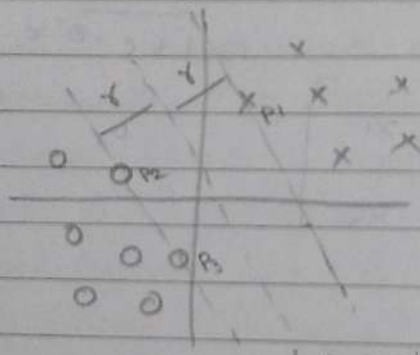
We want the hyperplane decision boundary as far as possible from even the closest point. Else a small change in hyperplane would change the classification of this close point

→ Distance between hyperplane and the closest point is called margin.

→ The objective of SVM is to maximize the margin.

→ There are infinitely many separating hyperplanes. The best hyperplane as far as SVM is concerned is the one that maximizes the distance between closest data points from both classes. (i.e. the margin).

→ Hence, SVM is a maximum margin classifier



Margin — γ

Support vector - The closest data points (p_1, p_2, p_3) which may be N-dimensional

hyperplane = $\theta^T x + \theta_0 = \beta^T x + \beta_0 = 0$.

→ Lets label +ve examples as +1 and -ve examples as -1

→ Consider an example of hyperplane ...

$x_1 + x_2 = 3$

$x_1 + x_2 - 3 = 0$

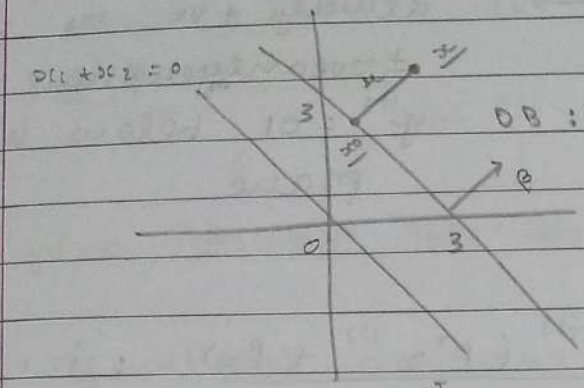
→ Compare with $\underline{\beta}^T \underline{x} + \beta_0 = 0$

$\beta_1 x_1 + \beta_2 x_2 + \beta_0 = 0$

and get $\beta_0 = -3$, $\beta_1 = 1$, $\beta_2 = 1$.

$\beta^T x + \beta_0 = 0$

$1 \times n \quad n \times 1 \quad 1 \times 1$



↓

β is $\perp^r$ to x and hence β is $\perp^r$ to the hyperplane

→ $(\underline{x} - \underline{x_0}) = (r) \frac{\underline{\beta}}{\|\beta\|}$

magnitude = r
direction = unit vector in direction of $\underline{\beta}$

→ $(\underline{x} - \underline{x_0}) \cdot \frac{\underline{\beta}}{\|\beta\|} = r$

→ $r = \frac{\beta^T \underline{x} - \beta^T \underline{x_0}}{\|\beta\|}$ $\quad [x \cdot \beta = \beta^T x]$

Now $\underline{x_0}$ lies on hyperplane so

$\beta^T \underline{x_0} + \beta_0 = 0$.

$r = \frac{\beta^T \underline{x} + \beta_0}{\|\beta\|}$

→ Now, we can use this formula of r to calculate distance of any x from hyperplane.

$r > 0$

$r < 0$

** constraint optimisation problem

→ $\frac{y^{(i)}[\beta^T x^{(i)} + \beta_0]}{\|\beta\|}$ → always +ve as $y^{(i)} = 1$ above hyperplane and $y^{(i)} = -1$ below hyperplane

(Hence $y^{(i)}$ choosen to be ±1 only)

→ Margin: $\min_{\forall x^{(i)}} \frac{y^{(i)}(\beta^T x^{(i)} + \beta_0)}{\|\beta\|}$; $i = 1, \ldots, m$

→ The problem formulation for SVM is to maximize the margin.

$$\max_{\beta, \beta_0} \left[\min_{\forall x^{(i)}} \frac{y^{(i)}(\beta^T x^{(i)} + \beta_0)}{\|\beta\|} \right]$$

(cannot be = 0 as then the example lies on decision boundary)

such that $y^{(i)}(\beta^T x^{(i)} + \beta_0) > 0$ **

* → Since β is not dependent on x, we have

$$\max_{\beta, \beta_0} \frac{1}{\|\beta\|} \left[\min_{\forall x^{(i)}} y^{(i)}(\beta^T x^{(i)} + \beta_0) \right]$$

(min possible value = 1.)

such that $y^{(i)}(\beta^T x^{(i)} + \beta_0) > 0$.

→ Taking $y^{(i)}(\beta^T x^{(i)} + \beta_0) \geq c$ will give. (inequality constraint)

$$y^{(i)}\left(\frac{\beta^T}{c} x^{(i)} + \frac{\beta_0}{c}\right) \geq 1$$ (+ve constant)

$$y^{(i)}(\beta^{*T} x^{(i)} + \beta_0^*) \geq 1$$

β^*: new set of parameter

Just a scaled version of β so hyperplane $\beta^T x + \beta_0 = 0$ will not change.

* → By taking that as min = 1, we get.

→ $\max_{\beta, \beta_0} \frac{1}{\|\beta\|}$ such that $y^{(i)}(\beta^T x^{(i)} + \beta_0) \geq 1$

→ $\min_{\beta, \beta_0} \|\beta\|$ (equivalent)

(Not > 0)

→ $\min_{\underline{\beta}, \beta_0} \frac{1}{2}\|\beta\|^2$ is also equivalent.

so that it cancels when we take derivative.

↓

Objective function

→ If atleast any one of objective function or constraint is non linear, then its called NLP (Non linear programming)

→ Non linear problems are usually difficult to solve.

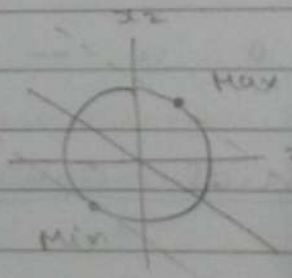
→ Lagrange Multiplier Approach is used to solve for $\underline{\beta}, \beta_0$.

EG Optimize $2x_1 + 3x_2$ such that $x_1^2 + x_2^2 = 1$

Linear objective function

Non linear constraint

$$2x_1 + 3x_2 = 0$$
$$3x_2 = -2x_1$$
$$x_2 = -\frac{2}{3}x_1$$



→ Lagrange method....

$$L(x_1, x_2, \lambda) = (2x_1 + 3x_2) + \lambda (x_1^2 + x_2^2 - 1)$$

Lagrange multiplier $f(x)$ $g(x)$

→ Differentiate wrt x_1, x_2, λ

$$2 + 2x_1\lambda = 0 \Rightarrow x_1 = -1/\lambda$$
$$3 + 2x_2\lambda = 0 \Rightarrow x_2 = -3/2\lambda$$
$$x_1^2 + x_2 - 1 = 0$$
$$\frac{1}{\lambda^2} + \frac{9}{4\lambda^2} = 0 \Rightarrow \frac{13}{4\lambda^2} = 1 \Rightarrow \lambda = \pm\frac{\sqrt{13}}{2}$$

[Constraint $y^{(i)}(\underline{\beta}^T x^{(i)} + \beta_0) \geq 1$ is equivalent to $\underbrace{1 - y^{(i)}(\underline{\beta}^T x^{(i)} + \beta_0)}_{g^{(i)}(\underline{\beta}, \beta_0)} \leq 0$]

→ λ is called the lagrange multiplier.

→ $\lambda = \frac{\sqrt{13}}{2}$. $\Rightarrow$ $x_1 = -\frac{2}{\sqrt{13}}$, $x_2 = -\frac{3}{\sqrt{13}}$

$\lambda = -\frac{\sqrt{13}}{2}$ $\Rightarrow$ $x_1 = \frac{2}{\sqrt{13}}$, $x_2 = \frac{3}{\sqrt{13}}$

Here, the second pair of (x_1, x_2) will give the max $f(x_1, x_2)$ [because for first pair (x_1, x_2), $f(x_1, x_2)$ is -ve]

λ +ve — func min

λ -ve — func max

→ θ, β, w — all notations equivalent

Primal Optimisation Problem

→ min $f(\underline{w})$ s.t. $g_i(w) \leq 0$ and $h_i(w) = 0$

for $i = 1 \ldots k$ for g_i

for $i = 1 \ldots l$ for h_i

To solve this problem, define generalised lagrangian $L(\underline{w}, \underline{\alpha}, \underline{\beta}) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$

α_i $i = 1$ to k

β_i $i = 1$ to l

→ lagrange multipliers

$$\theta_p(w) = \max_{\underline{\alpha}, \underline{\beta}, \alpha_i \geq 0} L(\underline{w}, \underline{\alpha}, \underline{\beta})$$

↓

equivalent to λ used earlier

→ Let us say some $\underline{w}$ is given.

→ If $\underline{w}$ violates any primal constraints, i.e. $g_i(w) > 0$ and/or $h_i(w) \neq 0$, then $\theta_p(w) = \infty$.

→ When constraints are satisfied, we get some $\theta_p(w)$.

→ So the original primal problem is written as $\min_{\underline{w}} (\theta_p(\underline{w}))$ i.e.

$$\min_{\underline{w}} \left[\max_{\underline{\alpha}, \underline{\beta}, \alpha_i \geq 0} L(\underline{w}, \underline{\alpha}, \underline{\beta}) \right]$$

→ This will give the required parameters $\underline{w}^*$, $\underline{\alpha}^*$, $\underline{\beta}^*$ and on putting those values back we get p^*.

→ Suppose we interchange the max, min order, then we land into the dual problem. $\max_{\underline{\alpha}, \underline{\beta}, \alpha_i \geq 0} \left[\min_{\underline{w}} L(\underline{w}, \underline{\alpha}, \underline{\beta}) \right]$, d^*

Note : $\underline{w}$ - primal variable

$\underline{\alpha}, \underline{\beta}$ - dual variable.

→ Now, we try understanding primal problem with an example ...

Take some $\underline{w_i} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}_{(n+1) \times 1}$

→ So now $\underline{w_i}$ is fix. Take different pairs of $\underline{\alpha}$, $\underline{\beta}$ and obtain values of function. Choose the max out of these and let that be associated with w_i.

→ We do the same with multiple $\underline{w_s}$.

$\underline{w_1}$ – 25
$\underline{w_2}$ – 26
$\underline{w_3}$ – 27
$\underline{w_4}$ – 88. and so on ... (A)

Now p^* will be the min of all these (corresponding to all $\underline{w_s}$). So here $p^* = 25$

→ Now if try using dual approach we have some α, β as constant and obtain functional value with different w_s. Then choose min of those for each α, β

$\underline{\alpha_1}, \underline{\beta_1}$ – 2
$\underline{\alpha_2}, \underline{\beta_2}$ – 3
$\underline{\alpha_3}, \underline{\beta_3}$ – 4
$\underline{\alpha_4}, \underline{\beta_4}$ – 1.

→ Intuitively these values must be lower than those in (A)

Now d^* is the max of these i.e. $d^* = 4$

→ Clearly $d^* \leq p^*$

→ We get the correct solution by primal method, not dual (when $d^* < p^*$)

→ However, under some condition, we get $d^* = p^*$

→ There are conditions called Karush-Kuhn-Tucker (KKT) under which $d^* = p^*$ so that we can get $\underline{w}$ either by using primal or dual formulation of problem

→ KKT conditions:

① $\frac{\partial}{\partial w_i} L(\underline{w}^*, \underline{\alpha}^*, \underline{\beta}^*) = 0 \quad i = 0 \ldots n$

② $\frac{\partial}{\partial \beta_i} L(\underline{w}^*, \underline{\alpha}^*, \underline{\beta}^*) = 0 \quad i = 1 \ldots l$

③ $\alpha_i^* g_i(\underline{w}^*) = 0 \quad i = 1 \ldots k$

$g_i(\underline{w}^*) \leq 0$

$\alpha_i^* \geq 0.$

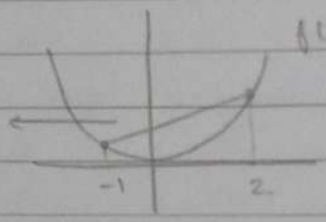
Only for s.v. α_i's exist

(condition true only when $\alpha \neq 0$)

→ The conditions will be satisfied for f and g are convex function h are affine.

→ Function f is convex if
$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$
where $\alpha \in [0,1]$, $x \in R^n$

Eg. $f(x) = x^2$ is a convex function



$x_1 = -1$
$x_2 = 2$
$\alpha = 0.5$

→ $f((0.5)(-1) + (1-0.5)(2)) = f(0.5) = 0.25$
→ $(0.5) f(-1) + (1-0.5) f(2) = 2.5$
Hence convex.

→ $f(w)$ here is $\frac{1}{2}\|\beta\|^2$

$g(w)$ here is $1 - y^{(i)}(\beta^T x^{(i)} + \beta_0)$

$$L(\underline{\beta}, \beta_0, \lambda) = \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{m} \left[\lambda_i (1 - y^{(i)}(\beta^T x^{(i)} + \beta_0))\right]$$

$\beta_1^2 + \beta_2^2 + \cdots + \beta_n^2$

→ We solve using dual approach i.e.

$$\max_{\lambda_i} \left[\min_{\underline{\beta}, \beta_0} L(\underline{\beta}, \beta_0, \underline{\lambda}) \right]$$

↓

gives $\underline{\beta}^*$ and β_0^* Put that in actual function to get a function of λ and then maximize over λ

* These always appear as dot product (while using dual approach)

Page No.: Date: YOUVA

→ So consider minimizing $L(\underline{\beta}, \beta_0, \underline{\lambda})$ $\forall\ \beta_0, \underline{\beta}$

→ Differentiate L wrt. each β_i ; $i = 0$ to n and equate to 0. i.e.

$$\frac{\partial}{\partial \beta_1} L(\underline{\beta}, \beta_0, \underline{\lambda}) = 0$$

$$\frac{\partial}{\partial \beta_2} L(\underline{\beta}, \beta_0, \underline{\lambda}) = 0 \quad \text{and so on....}$$

which is $\nabla_{\underline{\beta}} L(\underline{\beta}, \beta_0, \underline{\lambda}) = 0$.

$$\beta_1 - \sum_{i=1}^{m} \lambda_i (y^{(i)} x^{(i)}) = 0.$$

$$\underline{\beta} - \sum_{i=1}^{m} \lambda_i (y^{(i)} x^{(i)}) = \underline{0}$$

($\underline{\beta}$: $n \times 1$, $x^{(i)}$: $n \times 1$)

Hence $\underline{\beta}^* = \sum_{i=1}^{m} \lambda_i (y^{(i)} x^{(i)})$ → Derivate wrt to β_1 to β_n

$$0 = \sum_{i=1}^{m} \lambda_i y^{(i)} \rightarrow \text{Derivate wrt } \beta_0$$

→ Using $\|x\|^2 = x^T x$ ...

$$L = \frac{1}{2} \left[\sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)} \right]^T \left[\sum_{j=1}^{m} \lambda_j y^{(j)} x^{(j)} \right] + \sum_{i=1}^{m} \lambda_i \left[1 - y^{(i)} \left(\sum_{j=1}^{m} \lambda_j y^{(j)} x^{(j)} \right)^T x^{(i)} + \beta_0 \right]$$

$$= \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y^{(i)} y^{(j)} \boxed{x^{(i)T} x^{(j)}}$$

(Combine like terms and use $\sum_{i=1}^{m} \lambda_i y^{(i)} = 0$)

* Quadratic objective function with linear constraints (This is a convex optimization problem).

for λ_i s

* This has to now maximised (because this is now a function of λ) on $\lambda_i \geq 0$ for $i = 1, \ldots, m$ with constraint $\sum \lambda_i y_i = 0$ for $i = 1 \ldots m$

↓

Formulate problem to solve for λ_is (using lagrangian approach — lagrangian multiplier μ, α)

① → Once we get λ_i s, find $\underline{B} = \sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)}$

② → Convex problem — We will have a unique solution (do not worry about multiple minima)

→ This is a quadratic optimisation problem (with quadratic objective function and linear constraints).

→ This is a convex optimisation problem (so we do not have to worry about multiple minimas). We will have a unique solution.

→ Once we get λ_i's, find $\underline{\beta} = \sum_{i=1}^{m} \lambda_i y^{(i)} x^{(i)}$

$\downarrow$

$\beta_1 \dots \beta_n$

⇒ Now our target is to find value of β_0 from $\underline{\beta}^T x + \beta_0 = 0$.

→ For this we use the KKT conditions

$\lambda_i^* \; g(\underline{\beta}^*, \beta_0^*) = 0 \qquad i = 1 \dots m$

$g(\underline{\beta}^*, \beta_0^*) = 1 - y^{(i)} (\beta^{*T} x^{(i)} + \beta_0^*)$

(This is ≤ 0)

→ we have $\lambda_i^* \geq 0$

→ so if $\lambda_i^* > 0$, then $y^{(i)} (\beta^{*T} x^{(i)} + \beta_0^*) = 1$

→ This shows that $\lambda_i > 0$ only for support vectors (i.e. λ_i's exist i.e. $\lambda_i \neq 0$ for support vector only

→ For all the other examples, $\lambda_i = 0$

So far solving our initial problem,

$\min_{\underline{\beta}, \beta_0} \frac{1}{2} \|\beta\|^2 \quad$ s.t. $\quad y^{(i)} (\underline{\beta}^T x^{(i)} + \beta_0) \geq 1$

only a few data points are used (i.e. only the support vectors are used because they have non zero λ_i)

$L = (\dots) + \sum_{i=1}^{s} \lambda_i (\dots)$ if there are s support vectors

→ The research problem here is identifying these support vectors.

→ Once we know the S.V. and also know whether it belongs to class 1 or class -1, we have -

For +ve example : $\underline{\beta}^{T*} x^{(i)} + \beta_0^* = 1$

$\beta_0^* = 1 - \underline{\beta}^{T*} x^{(i)}$

For -ve example : $\beta_0^* = -1 - \underline{\beta}^{T*} x^{(i)}$.

Here all is correspond to support vector.

$$L = \sum_{i=1}^{m} \lambda i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda i \lambda j \, y^{(i)} y^{(j)} x^{(i)T} x^{(j)}$$

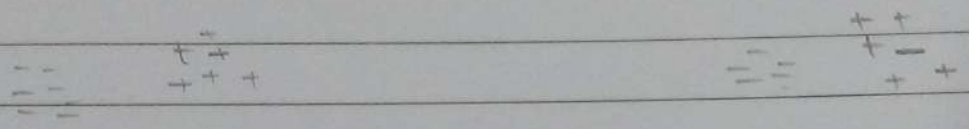
The dimension of solution of this problem is R^m ($\lambda_1 \dots \lambda m$) irrespective of the size of data ($x_0 \dots x_n$), because $x^{(i)T} x^{(i)}$ is always a single value

Advantage of dual approach - Even if the data is not linearly separable, we move from $x \to \phi(x)$ but the problem still remains the same.

Page No.:
Date:
YOUVA

→ Here we are imposing a strict constraint that no data point is closer than 1 from the D.B. This is called hard margin classifer.

→ With primal approach, if data is not linearly separable by hyperplane, then this method gives no solution



Linearly separable

Not linearly separable

↓

At least for some points, the constraint $y^{(i)}(\beta^T x^{(i)} + \beta_0) \geq 1$ is not satisfied.

→ In order to solve this problem when data is not linearly separable, we use softmax classifier.

$z = xc$ ← channel behavior

↑ data of sent signal ↑ received data

(needed apriori for determining c but once c is obtained z is calculated from x and c)

ML analogy

$\rightarrow y = X\theta$

$$X = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(n)} \end{bmatrix}$$

$$MSE = (y - X\theta)^T (y - X\theta)$$

$$= (y^T - \theta^T X^T)(y - X\theta)$$

$$= y^T y - y^T X\theta - \theta^T X^T y + \theta^T X^T X\theta$$

$$\frac{\partial MSE}{\partial \theta} = 0$$

$$= -y^T X - y^T X + \theta^T X^T X + \theta^T X^T X = 0$$

$$y^T X = \theta^T X^T X$$

$$X^T y = X^T X \theta$$

$$\theta = (X^T X)^{-1} X^T y$$

$y \rightarrow z$

$\theta \Rightarrow c$

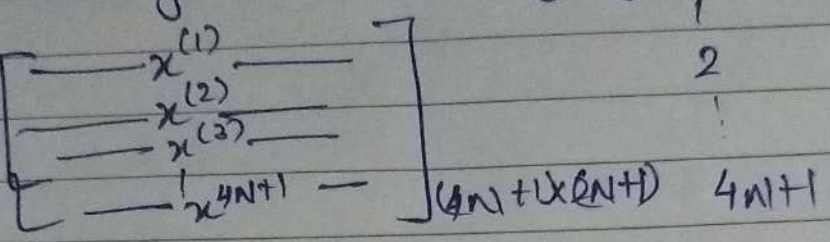
$X \rightarrow X$

$$c = (X^T X)^{-1} X^T z$$

$$= R_{xx}^{-1} R_{xy}$$

↑ autocorrelation

zero forcing → is like saying, we ignore



X(k-N) to X(k+N) ← features.

R_{XX} and R_{XZ} are correlations between the samples.
But X is actually data that we ~~send~~ received so we can calculate X only if the samples are ergodic in nature
(Not sure what I mean by this but the analogy says so)