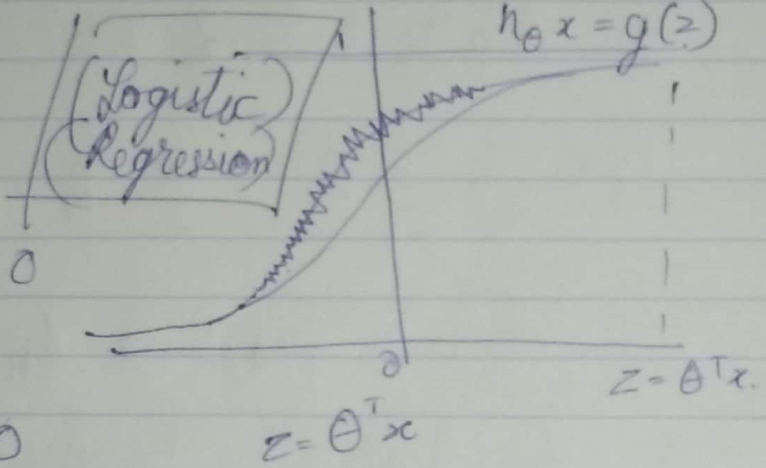
WEEK - 7

SUPPORT VECTOR MACHINE
(supervised Learning).

$\Rightarrow$ $h_\theta(x) = \dfrac{1}{1+e^{-\theta^T x}}$



if $y=1$, we want $h_\theta(x) \approx 1 \rightarrow \theta^T x >> 0$

if $y=0$, we want $h_\theta(x) \approx 0$, $\theta^T x << 0$

$\Rightarrow$ Optimization $\rightarrow$ Cost f^n $\rightarrow$ for SVMs

$$\Rightarrow \min_\theta C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) \right]$$

$$\left(C = \frac{1}{\lambda}\right) \qquad + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

Hypothesis $\rightarrow$ $h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$

$\Rightarrow$ If $y=1$, we want $\theta^T x \geq 1$ (not just ≥ 0)
If $y=0$, we want $\theta^T x \leq -1$ (not just < 0)

⇒ Margin of SVM → the distance b/w the closest points of both classes from the separation line

⇒ SVM are also called Large Margin Classifier.

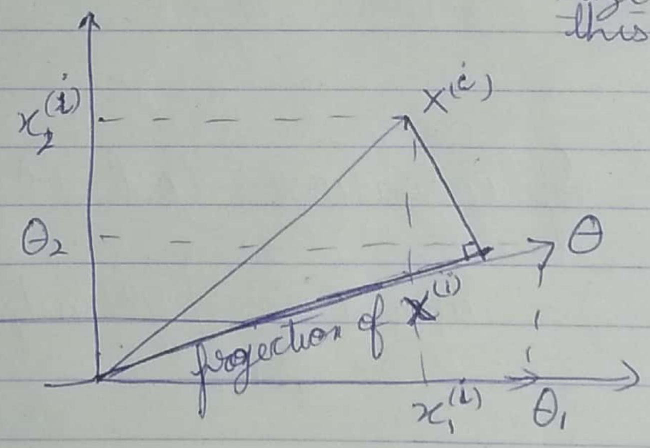
⇒ **Maths behind SVM**

$$\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}\left(\theta_1^2+\theta_2^2\right) = \frac{1}{2}\left(\sqrt{\theta_1^2+\theta_2^2}\right)^2 = \frac{1}{2}\|\theta\|^2$$

we have to minimize this

$\|\theta\|$ → length of vector θ

⇒ $\theta^T x$ →



θ

⇒ $\theta^T x^{(i)} = p^{(i)} \cdot \|\theta\|$

$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$

∴ in $p^{(i)}\|\theta\|$

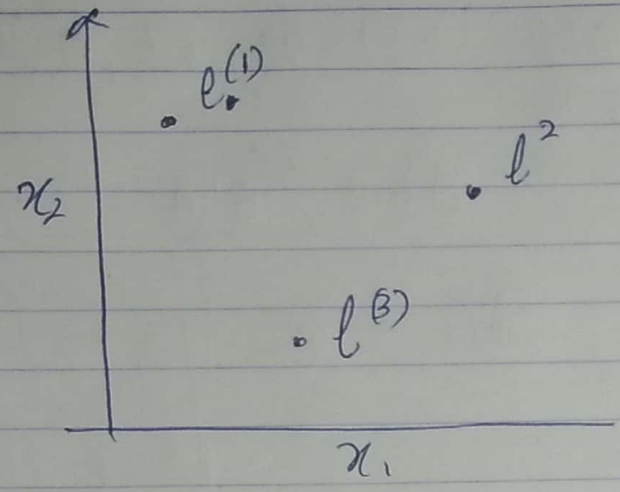
→ to minimize this we need to make $p^{(i)}$ large because they have to

$p^{(i)} \cdot \|\theta\| \geq 1$ for $y^{(i)} = 1$

$p^{(i)} \cdot \|\theta\| \leq -1$ for $y^{(i)} = 0$

⇒ Optimization problems of SVM are convex in nature.

⇒ <u>Kernels.</u>



(3 random points are choosen)

these are landmarks.

given x: $f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\dfrac{\|x - l^{(1)}\|^2}{2(\sigma)^2}\right)$

: $f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\dfrac{\|x - l^{(2)}\|^2}{2(\sigma)^2}\right)$

: $f_3 = \text{similarity}(x, l^{(3)}) = \exp\left(\dfrac{-\|x - l^{(3)}\|^2}{2(\sigma)^2}\right)$

kernels

(Gaussian Kernel) also → $k(x, l^{(i)})$

$$\Rightarrow \exp\left(\frac{-\|x - l^{(1)}\|^2}{2(\sigma)^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2(\sigma)^2}\right)$$

∴ if $x \approx l^{(1)}$ (close) → $f_1 \approx \exp\left(-\dfrac{0^2}{2\sigma^2}\right) \approx \underline{\underline{1}}$

if x is far from $l^{(1)}$ → $f_1 \approx \exp\left(-\dfrac{(\text{large no})^2}{2\sigma^2}\right) \approx \underline{\underline{0}}$

⇒ New Cost f^n :-

$$\min_\theta \; C\sum_{i=1}^{m} y^{(i)}\text{cost}_1(\theta^T f^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T f^{(i)}) + \underbrace{\frac{1}{2}\sum_{j=1}^{m}\theta_j^2}$$

$\theta^T M \theta$ / $\theta^T\theta$

some matrix

$\theta = \begin{bmatrix}\theta_1 \\ \vdots \\ \theta_m\end{bmatrix}$ [ignoring θ_0]

⇒ SVM parameters.

$C \Rightarrow \frac{1}{\lambda}$, large $C \rightarrow$ lower bias, high variance
Small $C \rightarrow$ Higher bias, low "

$\sigma^2 \Rightarrow$ ~~large~~ Large σ^2 : → features f_i vary more smoothly → Higher bias, lower variance

⇒ Lower σ^2 → features f_i vary less smooth. Lower bias, higher variance.

⇒ SVM libraries → liblinear, libsvm

⇒ ~~If~~ using libraries, we need to specify
→ Choice of parameter C
→ Choice of Kernel (similarity function)

⇒ We can ~~also~~ also choose not to use a kernel → i.e. use a linear kernel SVM (no kernel SVM)

Scanned with CamScanner

⇒ No kernel ("Linear Kernel") (A Standard Classifier)

predict "$y = 1$" if $\theta^T x \geq 0$

↑ for a large no of features and small number of training examples.

⇒ Gaussian Kernel

$$f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2(\sigma)^2}\right) \text{ where } l^{(i)} = x^{(i)}$$

Need to choose σ^2

↑ when less no of features and high no. of examples (training)

→ Do perform feature scaling before using the Gaussian Kernel.

⇒ All the SVM ~~functions~~ kernel functions should ~~stat~~ satisfy the condition called 'Mercer's Theorem'.

⇒ Some other Kernels → Polynomial Kernel

$\therefore k(x, l) \Rightarrow (x^T l + 0)^2, (x^T l)^3, (x^T l + 1)^3, (x^T l + 5)^4 \ldots$

Form $k(x, l) = (x^T l + \text{constant})^{\text{degree}}$

Intuition → x^T & l will be close and so their inner product will be large.

Scanned with CamScanner

⇒ Esoteric Kernels —
1)= String Kernel
2)= chi-square Kernel
3)= Histogram Kernel
4)= Intersection kernel

⇒ Multiclass SVM classification

→ We can use one-vs-all method (Train K SVMs one to distinguish $y = i$ from the rest, for $i = 1, 2 \ldots K$), get $\theta^{(1)}, \theta^{(2)}, \ldots \theta^{(k)}$. Pick class i with the largest $(\theta^{(i)})^T x$.

⇒ ~~If~~

n = number of features
m = no of training examples.

⇒ if n is large (relative to m): ($n = 10{,}000$, $m = 10 - 1000$)
we can use ~~logistic~~ logistic regression or linear kernel SVM.

⇒ If n is small, m is intermediate.
→ Use SVM with gaussian kernel.

($n = 1 - 1000$, $m = 10 - 10{,}000$)

Scanned with CamScanner

⇒ If n is small, m is large ($n = 1-1000$, $m = 50{,}000+$)

→ create/add more features, then use logistic regression or SVM + linear kernel

⇒ All of the above scenarios, Neural Network can work well, but may be slower to train.