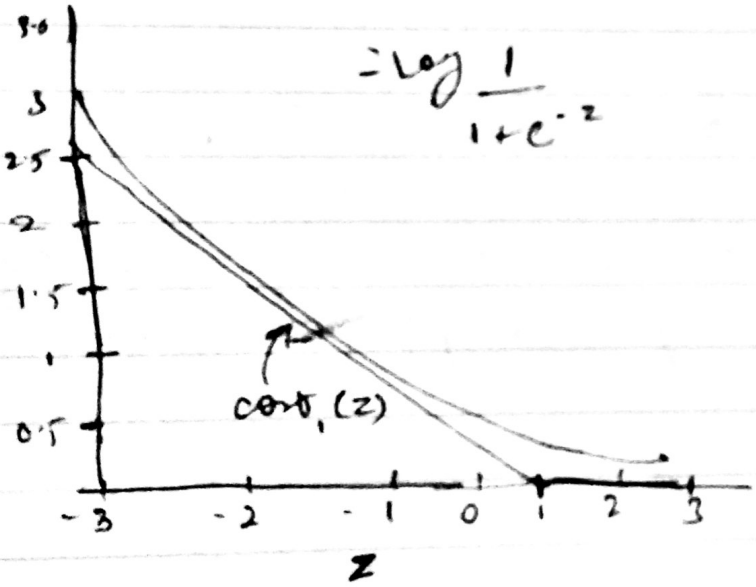
SUPPORT VECTOR MACHINES

cost function for logistic regression

$$= -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log \left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

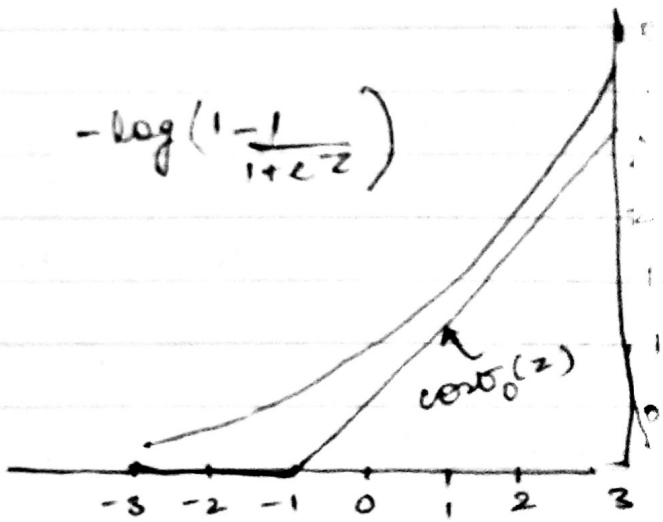
for $y = 1$, $\theta^T x >> 0$

$$= \log \frac{1}{1+e^{-z}}$$



for $y = 0$; $\theta^T x < 0$

$$-\log \left(1 - \frac{1}{1+e^{-z}}\right)$$



Logistic Regression,

$$\min_\theta \frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \left(-\log h_\theta(x^{(i)})\right) + \left(1-y^{(i)}\right) \left(-\log \left(1 - h_\theta(x^{(i)})\right)\right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

$\theta_2 x_2$ - $y = mx + c$. $x = 3$

$\theta_2 x_2 + \theta_1 x_1 + \theta_0 \geq 0$

for support vector machines,

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

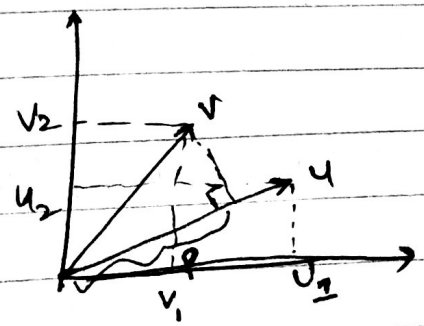
$$h_\theta(x) \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{if otherwise.} \end{cases}$$

SVM is a large margin classifier.

if C is very large, the decision boundary will try to incorporate every training example. (gives margin)

Maths behind SVM

Vector inner product



$\|u\|$ = length of vector u.

$= \sqrt{u_1^2 + u_2^2}$

$u^T v = p \cdot \|u\|$

$= u_1 v_1 + u_2 v_2$.

$\theta_1^2 + \theta_2^2$

$x = 0$

Kernels

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

for all $l^{(1)}, \ldots l^{(n)}$ $\quad l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$

$$\therefore f_j = \exp\left(-\frac{\sum_{j=1}^{n} (x_j - l_j^{(1)})^2}{2\sigma^2}\right)$$

If $x \approx l^{(1)}$ ← Gaussian kernel

$$\therefore f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right)$$

$$\approx 1$$

← is a parameter of gaussian kernel. = 1 almost all bin

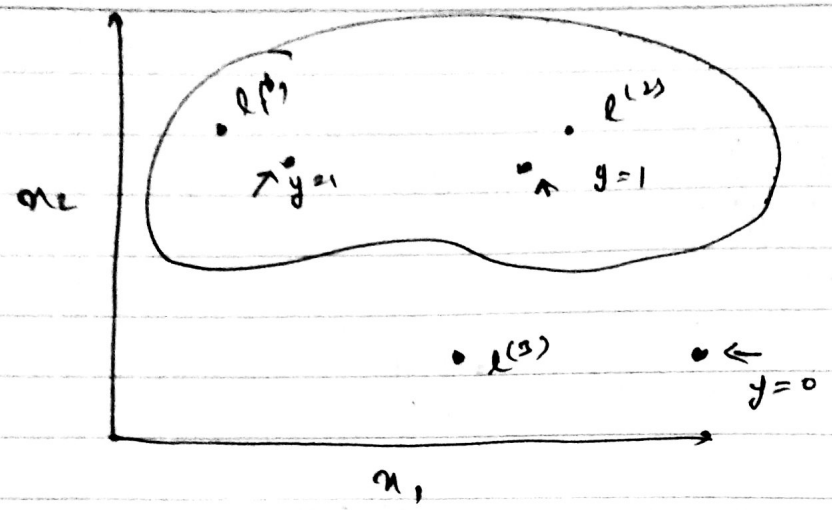
If x if far from $l^{(1)}$

$$f_1 = \exp\left(-\frac{(\text{big no.})^2}{2\sigma^2}\right) \approx 0.$$

$$\approx 0$$

If $\sigma^2 = 0.5$ → sharp contour

$\sigma^2 = 3.0$ → easy contour.



predict 1 , when

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

suppose , $\theta_0 = -0.5$

$\theta_1 = 1$

$\theta_2 = 1$

$\theta_3 = 0$

Examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})$

choose $l^{(1)} = x^{(1)}, l^2 = x^{(2)} \ldots l^{(m)} = x^{(m)}$

$f_1 = \text{sim}(x, l^{(1)})$

$f_2 = \text{sim}(x, l^{(2)})$

$\vdots$

$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$ $f_0 = 1$

for example $(x^{(i)}, y^{(i)})$

$x^{(i)} \in R^{n+1}$

$x^{(i)} = f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)})$

$f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)})$

$f_3^{(i)} = \text{sim}(x^{(i)}, l^{(3)})$

$\leftarrow f_1^{(1)} = \text{sim}(x^{(1)}, l^{(1)}) = 1$

$f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)})$

$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$

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

$\theta_0 f_0 + \theta_1 f_1 + \theta_2 f_2 \ldots$

$m = n$.

Training

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

this could be written as $\theta^T \theta$

$C \left(= \frac{1}{\lambda}\right)$; Large C : High variance

Small C : Higher bias.

σ^2 ; large σ^2 - Vary not smoothly
Higher bias.

; small σ^2 - vary sharply
Lower bias.

★ Use SVM package (eg., liblinear, libsvm, ...) to solve for θ.

Need to specify
→ Choice of C
→ Choice of kernel.

[if n is large & m is small]

Eg: No kernel = predict $y=1$ if $\theta^T x \geq 0$
(in others, if it is $\theta^T x \geq 1$)

[n is small / m is large]

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.

perform feature scaling, if there are too many differences in the features. suppose size of rooms (1000 sq ft), no of bedrooms (range 1-5).

★ They need to satisfy "Mercer's theorem", to make sure that θ is calculated correctly.

Multi-class classification

Many SVM have already built-in multi-class classification
or use one-vs-all method (like logistic regression).

we 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 largest $(\theta^{(i)})^T x$.

Logistic regression V/s SVM

(n = 10,000) (m = 10, ..., 1000)

if n is large (relative to m),
use logistic regression or SVM without a kernel.

(n = 1–1000) (m = 10 – 10,000)

* If n is small, m is intermediate
→ use SVM with gaussian kernel.

(n = 1 – 1000) (m = 50,000+)

If n is small, m is large
(Create/add more features, then use logistic regression or SVM without a kernel)

* Neural network will work well, however it is slower to train.

* In SVM, it will find global minima, not sure about neural network.

UNSUPERVISED LEARNING

learning from unlabelled data.
There's no y.

Clustering can be used for market analysis.