ML

Andrew Ng

Coursera

Week 7

Date:

Optimization Objective

The performance of many supervised learning algorithms will be ~~not~~ similar, amount of data & skill in applying these algorithms will matter more.

Support vector machines (SVM) sometimes gives a cleaner way of learning complex non-linear functions

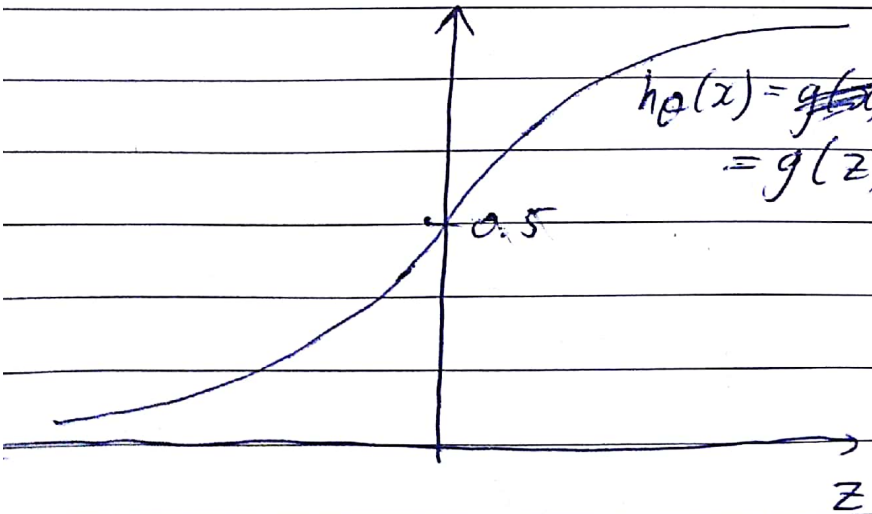
Alternative view of logistic regression

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

If $y=1$, we want $h_\theta(x) \approx 1$, $\theta^T x \gg 0$

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

This is what we want ideally



$h_\theta(x) = g(z)$

$z = \theta^T x$

Date:

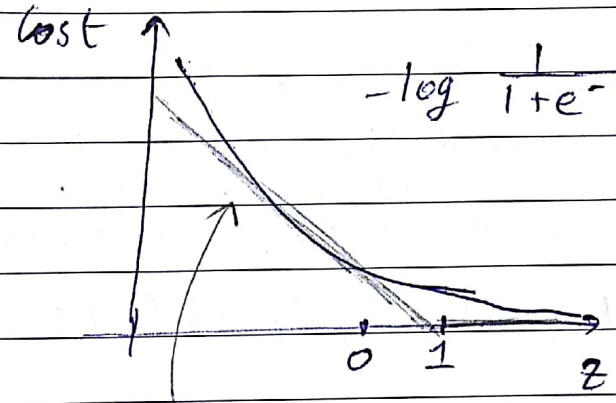
Cost of 1 example:

$$-(y \log h_\theta(x) + (1-y) \log(1-h_\theta(x))) =$$

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

if $y=1$ (want $\theta^T x \gg 0$):

$$z = \theta^T x$$

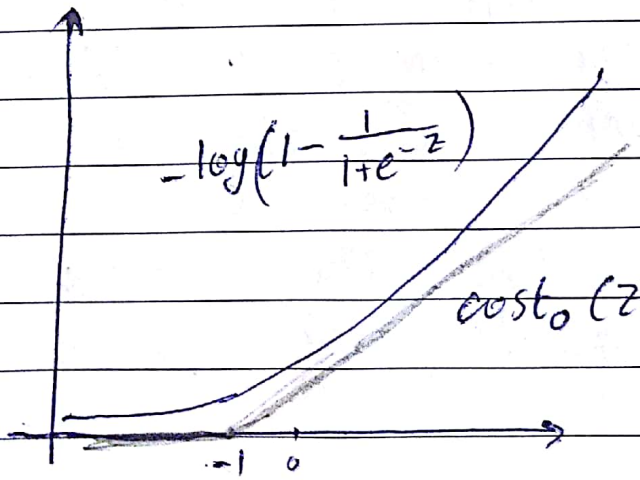


cost function of SVM is a close approximation to logistic regression cost function

cost function of SVM

↳ ~~cost~~ $\text{cost}_1(z)$

Simplified cost function of SVM gives computational advantages & give an easier optimi-sation problem



if $y = 0$ (want $\theta^T x \ll 0$)

Date:

subscript in $cost_0(z)$ & $cost_1(z)$ corresponds to the value of y the cost is modeled for.

Logistic regression:

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

A + B

SVM:

m is removed due to conventions used with SVMs, the minimum value does not change if constant is removed

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) + \frac{\lambda}{2m} \sum_{i=0}^{n} \theta_j^2$$

In logistic regression we have $A + \lambda B$ where λ would control the trade off between A & B. In SVMs we introduce $C = \frac{1}{\lambda}$ → $CA + B$ where C controls the trade off where the smaller the value of C, the greater the regularization occurs

Date:

SVM:

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

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

Hypothesis

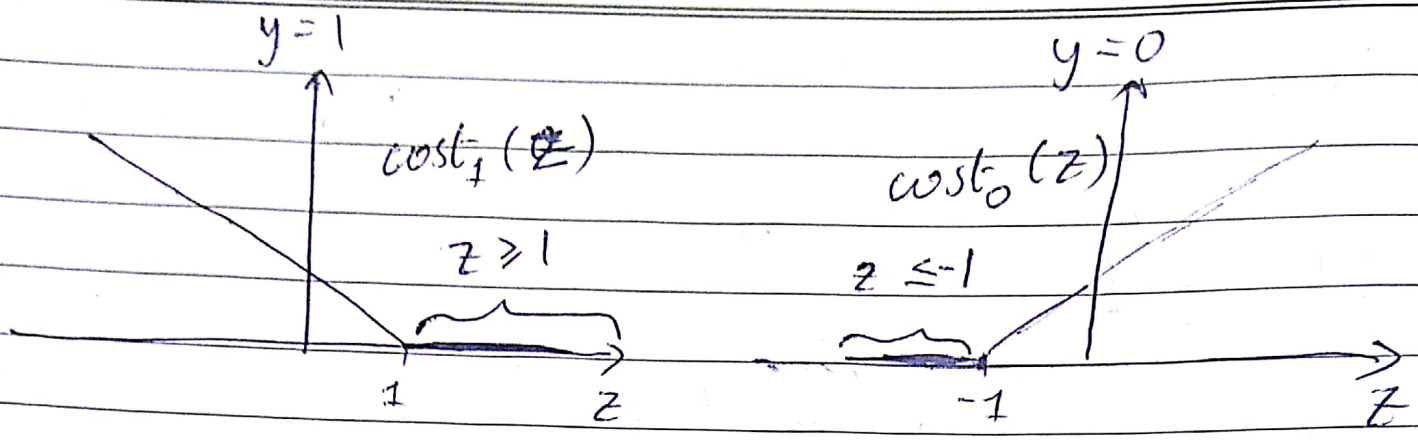
Large Margin Intuition

SVMs are sometimes called large margin classifiers

Support Vector Machine

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

Date:



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)

Even though the hypothesis requires that $\theta^T x \geq 0$ for $y = 1$ & $\theta^T x < 0$ for $y = 0$, the cost ensures that there is substantial distance from 0 ~~so that the predicted value is a strong 1 or strong 0 where it does not penalize~~ so that the predicted value is a strong 1 or strong 0

↳ builds in a safety margin factor into SVM

Suppose C is set to an extremely large value then when minimizing the optimization objective, the algorithm will be highly motivated to choose such a value so that the first term is ≈ 0

$=0$

Date:

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

$$+ \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

Whenever $y^{(i)} = 1$: $\theta^T x^{(i)} \geq 1$ where cost $= 0$

Whenever $y^{(i)} = 0$: $\theta^T x^{(i)} \leq -1$ where cost $= 0$

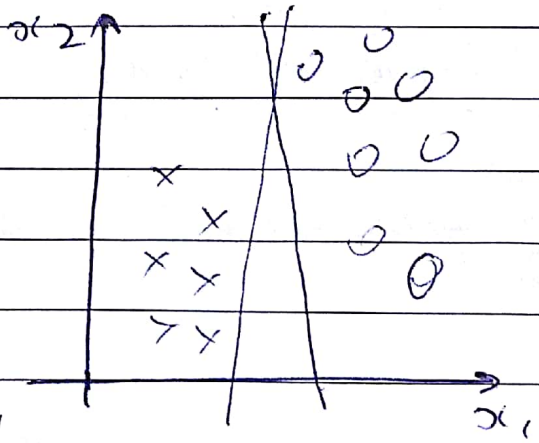
The optimization problem can further be simplified as such:

$$\min_{\theta} \; C \times 0 + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

such that $\theta^T x \geq 1$ if $y^{(i)} = 1$

$\theta^T x \leq -1$ if $y^{(i)} = 0$

The theta that is set due to the optimization problem above is such (decision boundary):

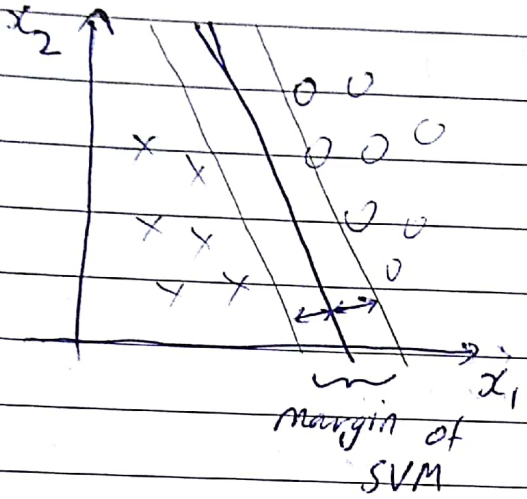


This data set is linearly separable - there exist many lines that can seperate the 2 classes

Date:

The lines drawn above, although are valid decision boundaries, do not represent good choices for seperation.

SVM decision boundary



The one chosen by SVM is a much better decision boundary than the lines drawn before.

↳ SVM chooses a more robust seperator

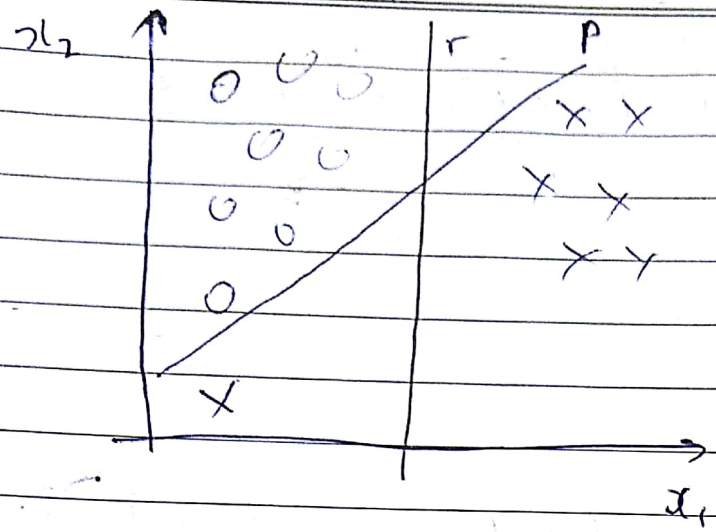
The decision boundary has a larger minimum distance from any of the training examples called a margin

↳ gives SVM robustness as it seperate the data with as large a margin as possible

SVM is also called a large margin classifier

↳ consequence of the optimisation problem

Date:



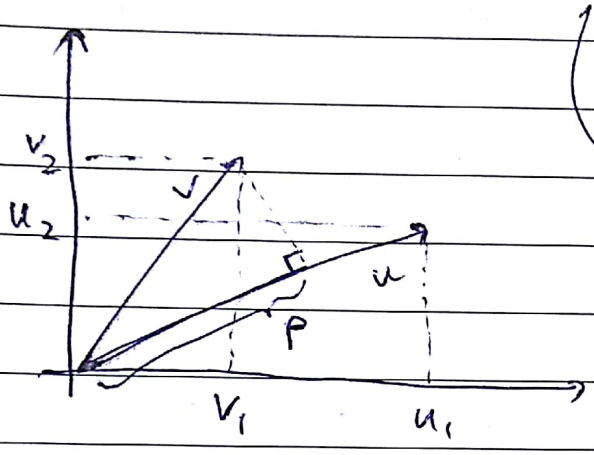
If C is very large then the SVM will choose the p decision boundary & if C is really small then SVM will choose the r decision boundary

If C is high then decision boundary is more sensative to outliers

The large margin classifier view of SVMs is only applicable when C is a large number.

Mathematics behind large Margin Classification

Vector inner product



$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$u^T v = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} =$$

$$= u_1 v_1 + u_2 v_2$$

$\|u\|$ = length of vector, $\|u\| = \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$

p = length of orthogonal projection of vector v on u (signed number)

$$u^T v = v^T u = p \cdot \|u\| = u_1 v_1 + u_2 v_2$$

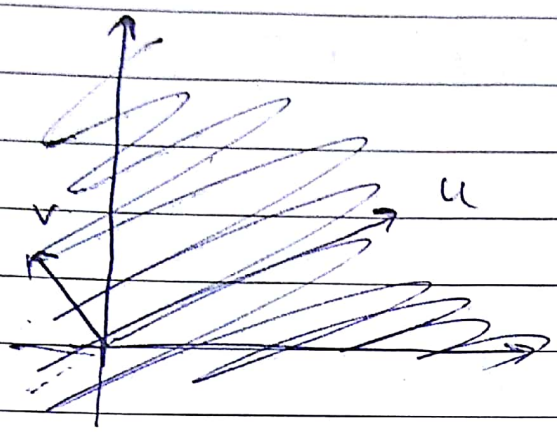
$p \in \mathbb{R}$

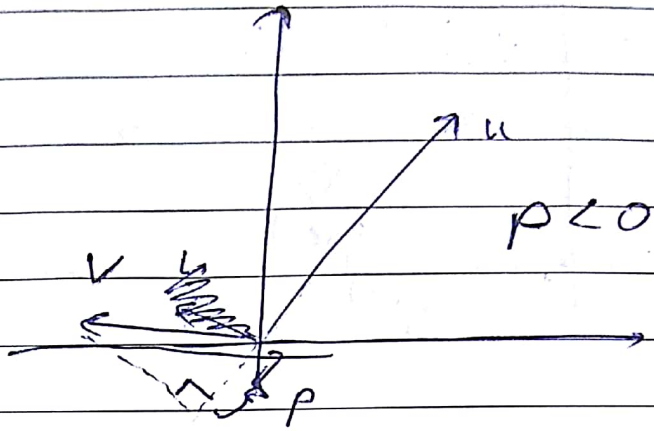
p is +ve in the graph above

S.t. = such that

Date:

p is +ve when the angle between u & v is less than 90°, otherwise it is negative





p is -ve here as angle between v & u is greater than 90°

SVM decision boundary

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

$$= \|\theta\| = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \text{ where } \theta_0 = 0$$

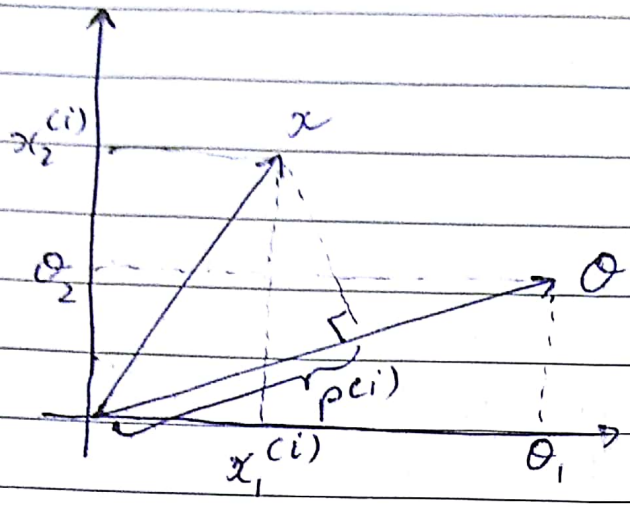
s.t. $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$

$\theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

Simplification: $\theta_0 = 0$, $n = 2$

thus $\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ is used

$\theta^T x$ is an inner product



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

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

$$\min_{\theta} \frac{1}{2} \sum \theta_j^2 = \frac{1}{2} \|\theta\|^2$$

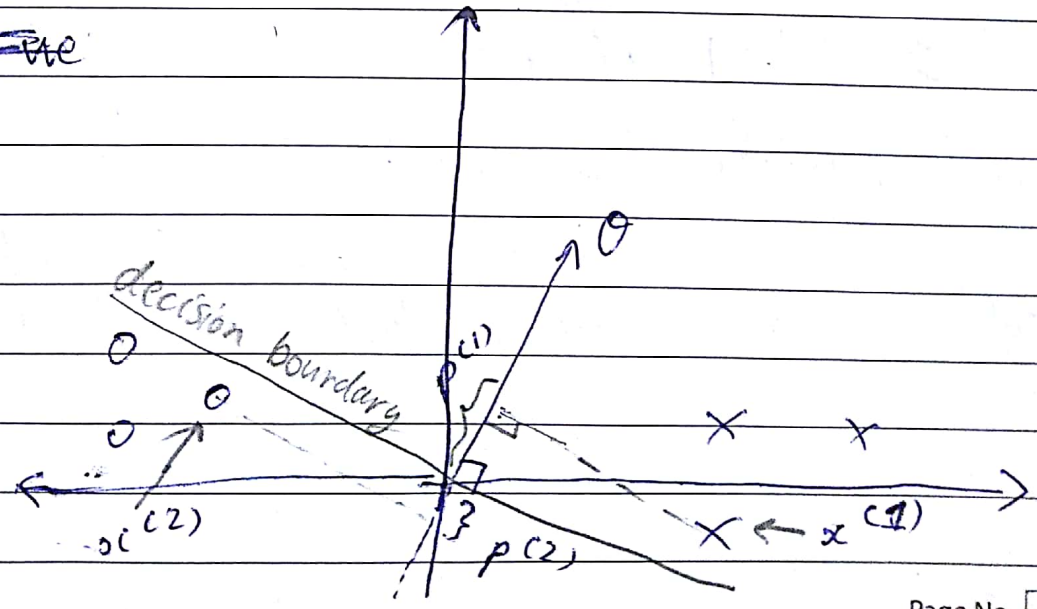
s.t. $p^{(i)} \cdot \|\theta\| \geq 1$ if $y^{(i)} = 1$

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

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ. Simplification: $\theta_0 = 0$

Suppose ~~the~~

Suppose this decision boundary is chosen. It has a really small margin



Victory

Page No.

Suppose $\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$ ← decision boundary line

$y = mx + c$

$\theta_2 x_2 = -\theta_1 x_1 - \theta_0$ if $\theta_0 = 0$ then:

$\theta_2 x_2 = -\theta_1 x_1$ & a valid (x_2, x_1) point is (0, 0)

Date:

& we will prove why SVM will not choose this decision boundary.

Note: for this choice of parameters it is possible to show that the parameter vector ~~theta~~ θ is at 90° to decision boundary

Note: $\theta_0 = 0$ means decision boundary must pass ~~over~~ across (0, 0)

$p^{(2)} < 0$ & $p^{(1)} > 0$

Both projections $p^{(2)}$ & $p^{(1)}$ are small in value

$p^{(1)} \cdot \|\theta\| \geq 1$

↳ since $p^{(1)}$ is small we need $\|\theta\|$ to be large

$p^{(2)} \cdot \|\theta\| \leq -1$

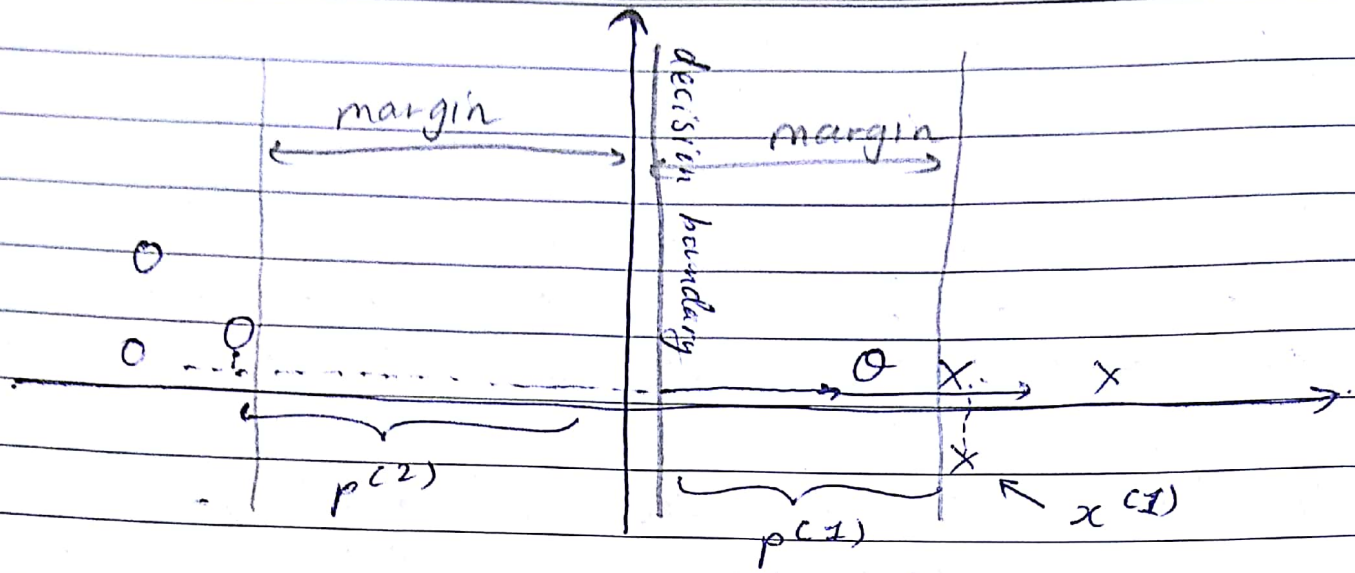
↳ since $p^{(2)}$ is small we need $\|\theta\|$ to be large as well

In the optimisation objective we are trying to find where $\|\theta\|$ is small → conflicts with the fact that we need a large $\|\theta\|$

Victory & hence SVM will not plot this line

Page No.

Date:



Now the ~~kg~~ lengths of $p^{(1)}$ & $p^{(2)}$ are much greater. Hence $\|\theta\|$ can be smaller. The decision boundary chosen here allows for $\|\theta\|$ to be small ~~hence~~ & does not conflict ~~&~~ the optimization objective

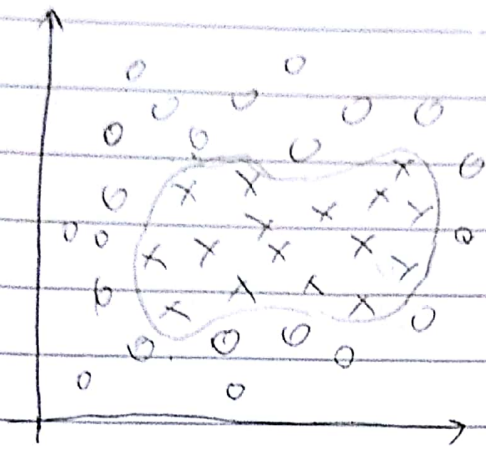
The SVM would plot $\|\theta\|$ like this

$p^{(i)}$ can only be large when there are large margins between the decision boundary & the examples → the magnitude of the gap ~~is the value~~ are the values of $p^{(i)}$

Kernels help SVMs learn complex non linear classifiers

Date:

Kernels



Predict $y = 1$ if

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \ldots \geq 0$

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \ldots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

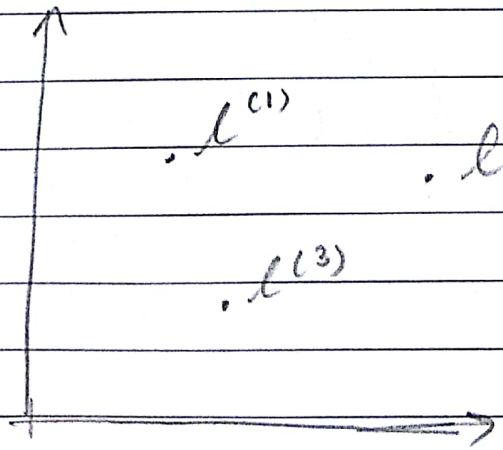
New notation:

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \ldots$

$f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, f_4 = x_1^2, f_5 = x_2^2$

Is there a different / better choice of features $f_1, f_2, f_3 \ldots$?

Kernel



Given x, compute new features depending on the proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$

line up

Date:

Given x:

$f_1 = \text{similarity}(x, \ell^{(1)}) = \text{ex}$

$= \exp\left(-\frac{\|x - \ell^{(1)}\|^2}{2\sigma^2}\right)$

only 1 value of x (one row)

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

$f_3 = \text{similarity}(x, \ell^{(3)}) = \exp(\ldots)$

similarity → kernel

specific type of kernel called Gaussian kernel

↳ how the similarity function is evaluated

There are many types of kernels i.e the similarity function can be evaluated in different ways

$$\text{similarity}(x, \ell^{(1)}) = k(x, \ell^{(1)})$$

Note: ignoring x_0 for the time being which is equal to 1

$\| x - \ell^{(i)} \|^2 = (x_1 - \ell_1^{(i)})^2 + (x_2 - \ell_2^{(i)})^2 + \ldots + (x_n - \ell_n^{(i)})^2$

$= (x_1 - \ell_1^{(i)})^2 + (x_2 - \ell_2^{(i)}) + \ldots + (x_n - \ell_n^{(i)})^2$

Date:

Kernels & similarity:

different dimensions of x are being subtracted to find distance

$f_1 = \text{similarity}(x, \ell^{(1)}) =$

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

If $x \approx \ell^{(1)}$:

$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$

$\exp(0) = e^0$

$\exp(x) = e^x$

If x is far from $\ell^{(1)}$:

$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0$

features (f_2, f_2 ...) measure how ~~close~~ similar x is from one landmark

Each landmark defines a new feature:

$\ell^{(1)} \Rightarrow f_1$

$\ell^{(2)} \rightarrow f_2$

$\ell^{(3)} \rightarrow f_3$

Example:

$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$, $f_1 = \exp\left(\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$

f_1 will be 1 if $x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$

$\sigma \uparrow$ then as x is moved away from $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$, the rate of drop from 1 to 0 is slower → slower drop in f

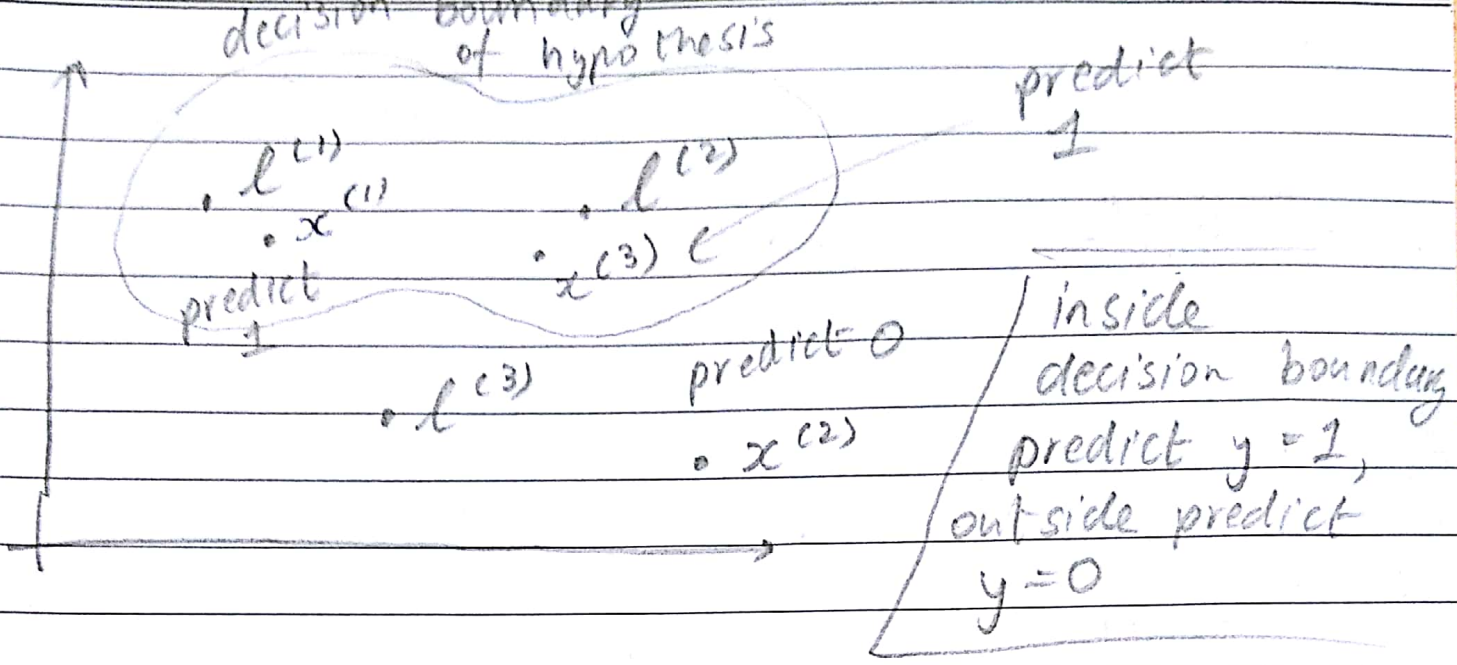
$\sigma \downarrow$ then as x is moved away from $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$, the rate of drop from 1 to 0 is faster → faster drop in f

---

Aside:

$x^{(i)} \rightarrow \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \end{bmatrix} \rightarrow \theta^T f$ evaluated

Date:



Predict 1 when:

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

plug in x into the features

Suppose $\theta_0 = -0.5$, $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = 0$ (which are already learned)

for $x^{(1)}$: $f_1 \approx 1$, $f_2 \approx 0$, $f_3 \approx 0$

$\theta_0 + \theta_2 \times 1 + \theta_3 \times 0 + \theta_3 \times 0 =$

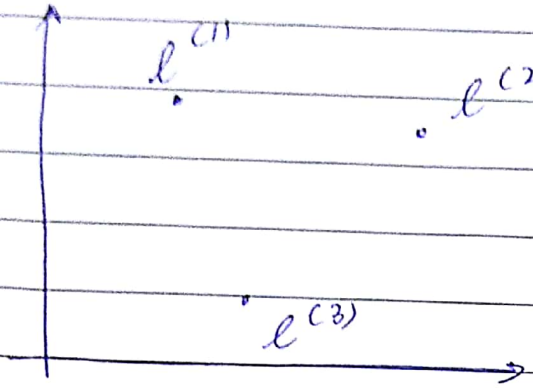
$= -0.5 + 1 = 0.5 \geq 0$ (assign 1)

for $x^{(2)}$: $f_1 \approx 0$, $f_2 \approx 0$, $f_3 \approx 0$

$\theta_0 + \theta_1 \times 0 + \theta_2 \times 0 + \theta_3 \times 0 = -0.5 < 0$ (assign 0)

Date:

Choosing the landmarks



given x:

f_i = similarity $(x, l^{(i)})$ =

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

hypothesis with kernels

- Predict $y = 1$ if $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Where to get $l^{(1)}, l^{(2)}, l^{(3)} \ldots$?

In practice, land marks are the ~~coordinates~~ data points themselves. Each data point is considered a landmark

Date:

~~Various values of ℓ : ℓ~~

Various values of ℓ : $[\ell^{(1)} \ \ell^{(2)} \ \ldots \ \ell^{(m)}]$

SVM with Kernels

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

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

Given example x :

$f_1 = \text{similarity}(x, \ell^{(1)})$

$f_2 = \text{similarity}(x, \ell^{(2)})$

$\vdots$

$f_m = \text{similarity}(x, \ell^{(m)})$

Construct an f vector:

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

Date:

For training example $(x^{(i)}, y^{(i)})$:

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

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

$x^{(i)} \rightarrow$ ⋮

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

$f_i^{(i)} = sim(x^{(i)}, l^{(i)}) = 1$

$x^{(i)}$

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \in \mathbb{R}^{m+1}$$

SVM with kernels

Hypothesis: given x, compute features $f \in \mathbb{R}^{m+1}$

predict "$y=1$" if $\underbrace{\theta^T f}_{\theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m} \geq 0$

Here $\theta \in \mathbb{R}^{n+1}$ ← not n but n

Training :

Date:

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

The value here would be n in the previous algorithm but here n=m hence m placed here

In actual SVM implementations the regularization parameter is implemented differently :

$$\sum_{j} \theta_j^2 = \theta^T \theta \rightarrow \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \text{ (ignore } \theta_0\text{)}$$

$$= \theta^T M \theta$$

↳ depends on the kernel used

↳ allows support vector machine to run more efficiently

Page No.

Date:

You can apply kernels to logistic regression as well but computational tricks that apply to support vector machines don't generalise to algos such as logistic regression

↳ using kernels with logistic regression becomes slow

SVMs & kernels tend to go well together because of these optimisation techniques

SVM parameters

$C = \frac{1}{\lambda}$ Large C: Lower bias, high variance
Small C: Higher bias, lower variance

σ^2 Large σ^2: features f_i vary smoothly
↳ higher bias, lower variance

Small σ^2: features f_i vary less smoothly
↳ lower bias, higher variance

Date:

Using an SVM

Use SVM software package (e.g. liblinear, libsvm ...) to solve for parameters θ

↳ there is no need to implement this yourself

Need to specify this though:
- Choice of parameter C
- Choice of kernel (similarity function):

E.g. No kernel ("linear kernel")
Predict $y = 1$ if $\theta^T x \geq 0$

↳ this can be used where n is large & m is small to avoid overfitting the data if you're trying to fit with a complicated function

E.g. Gaussian kernel:

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

Need to choose σ^2

The gaussian kernel can be used where n is small & m is large

so that a complex nonlinear decision boundary is learned

Victory

Page No.

Date:

Some software packages might ask you to implement a kernel function, or similarity function.

The ~~kernel~~ kernel similarity function can be implemented as such:

function f = kernel (x, ℓ)

$$f = \exp\left(\frac{\|x - \ell\|^2}{2\sigma^2}\right)$$

end

Note: Do perform feature scaling before using the Gaussian kernel

Reason: $\|x - \ell\|^2$

↳ suppose vector v as such:

$$v = x - \ell$$

Hence $\|v\| = v_1^2 + v_2^2 + \ldots + v_n^2 =$

$$= (x_1 - \ell_1)^2 + (x_2 - \ell_2)^2 + \ldots + (x_n - \ell_n)^2$$

$(x_1 - \ell_1)^2$: can be in 1000's of feet^2 if x_1 measures the size

$(x_2 - \ell_2)^2$: can be in single digits if x_2 measures ~~rooms~~ # of rooms

Date:

If feature scaling is not performed then house size will completely dominate the distance & # of rooms will be ignored. Perform scaling so that the # of rooms is not ignored

Other choices of kernel

Note: Not all similarity functions make valid kernels. They need to satisfy a technical condition called "Mercer's theorem" to make sure SVM packages' optimizations run correctly, and do not diverge

Many off the shelf kernels available:

- polynomial kernel: $k(x, l) =$

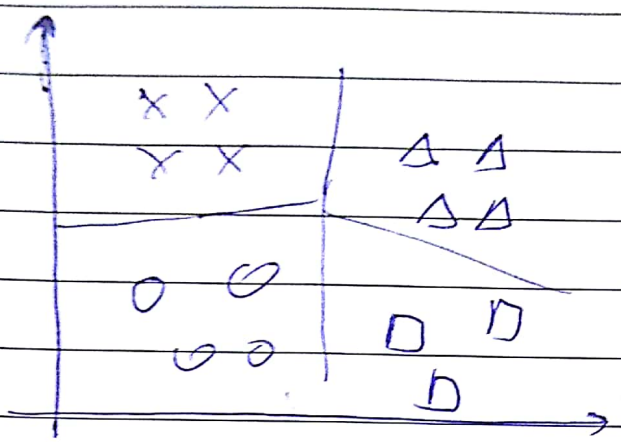
$= (x^T l)^2$ or $(x^T l + 1)^3$ or $(x^T l + 5)^5$

general form: $(x^T l + \text{constant})^{\text{degree}}$

Polynomial kernel almost always performs worse than gaussian kernel

- More esoteric kernels: String kernel, chi-square kernel, histogram kernel, etc

Multi-class classification



y can be of any given class

$y \in \{1, 2, 3 \dots k\}$

↳ ~~y can be~~ given ~~any of the values~~

Many SVM packages already have built-in multi class classification functionality.

Otherwise, use one-vs-all method
(Train k SVMs, one to distinguish $y = i$ from the rest for $i = 1, 2 \dots k$)

Get $\theta^{(1)}, \theta^{(2)}, \theta^{(3)} \dots \theta^{(k)}$

↳ $y=1$ ↳ $y=2$ ↳ $y=3$ ↳ $y=k$

Pick class i with largest $(\theta^{(i)})^T x$

Date:

Logistic regression vs SVMs

n = # of features
m = # of training examples

- if n is large (relative to m):
  n = 10'000, m = 10 ... 1000
  Use logistic regression, or SVM without a kernel ("linear regression")
  ↳ because there is not enough data to ~~learn~~ learn a complex non-linear function

- if n is small, m is intermediate:
  Use SVM with Gaussian kernel

  n = 1 ... 1000,
  m = 10 ... 10 000

- if n is small, m is large:
  ↳ (n = 1 ... 1000, m = 50 000+)
  ↳ create/add more features, then use logistic regression or SVM without a kernel
  ↳ if gaussian kernel is used then it becomes slow due to so many data points

Date:

Logistic regression & ~~SVM~~ SVM without a kernel are paired up together because they are pretty similar algorithms & will give similar performance & similar results

SVM with kernels tends to shine more when there are data points between 10'000 to 50'000 because they are capable of learning complex non-linear functions that are harder to do with logistic linear regression

Neural Networks are likely to work well for most of settings discussed above (the 3 classes) but may be slower to train

Informally, the C parameter is a +ve value that controls the penalty for misclassified training examples. A large C parameter tells the SVM to try to classify all the examples correctly. ~~It plays a role similar to $\frac{1}{\lambda}$ where λ~~