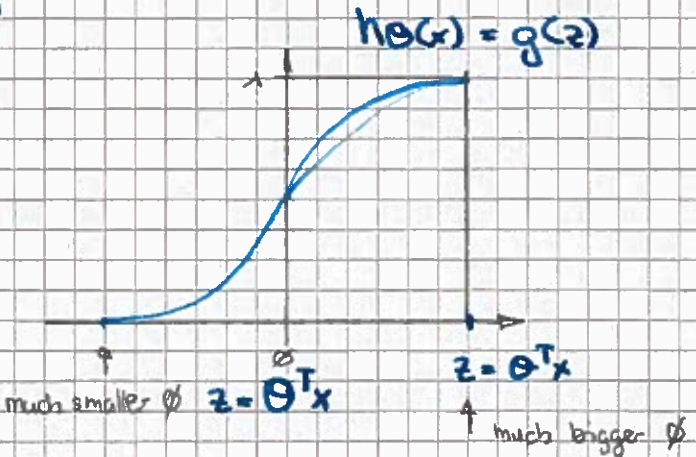
12. Support Vector Machines

Optimization Objective:

Alternative view of logistic regression

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



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

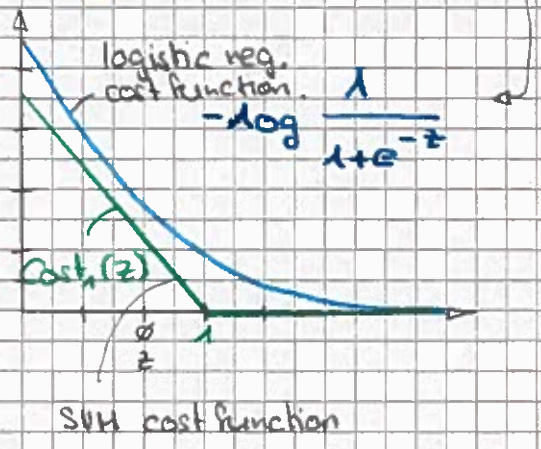
If $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

Cost function: $-(y \log h_\theta(x) + (1-y) \log (1 - h_\theta(x)))$

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

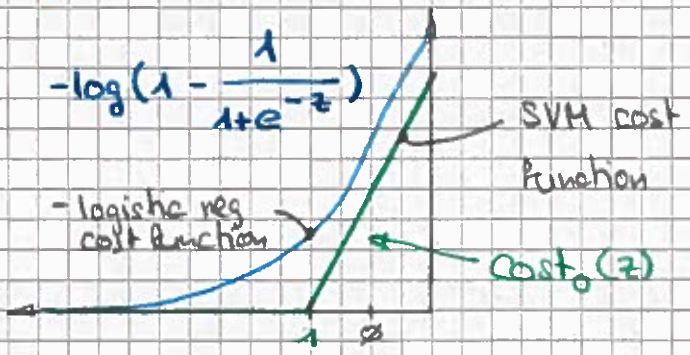
for single example

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



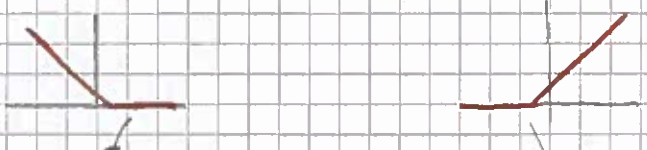
SVM cost function

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



Support Vector Machine:

Logistic Regression:



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

Support Vector Machine:

$$\min_\theta \underbrace{\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\,\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\,\text{cost}_0(\theta^T x^{(i)})}_{A} + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}_{B}$$

$\Rightarrow$ $A + \lambda B$

$CA + B$

$C = \frac{1}{\lambda}$

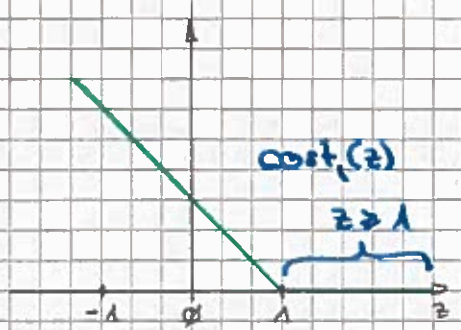
$\Rightarrow$ is basically the same if you want to control the regularization Param. Either give λ more weight or reduce the size of C to give the regularization more weight.

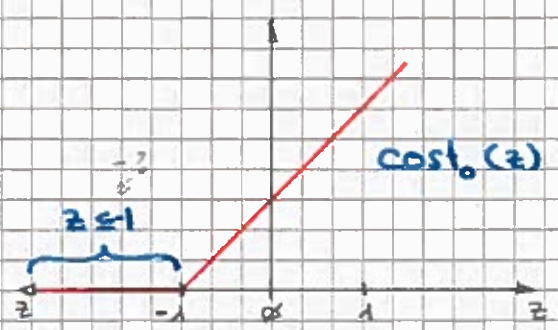
$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}\cdot\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\,\text{cost}_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

SVM Hypothesis:

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

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





if $y = 1 \rightarrow \theta^T x \geq 1$ (not just ≥ 0)

if $y = 0 \rightarrow \theta^T x \leq -1$ (not just < 0)

SVM decision Boundary — assume to be $C = 100'000$

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

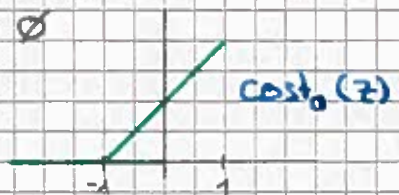
try to get this to 0

whenever $y^{(i)} = 1$

$\theta^T x^{(i)} \geq 1$

whenever $y^{(i)} = 0$

$\theta^T x^{(i)} \leq -1$

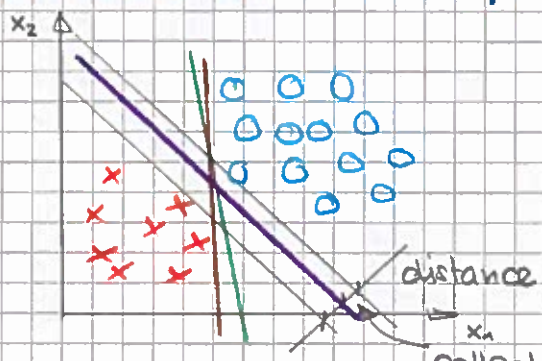


$$\min_\theta \; C \cdot 0 + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

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

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

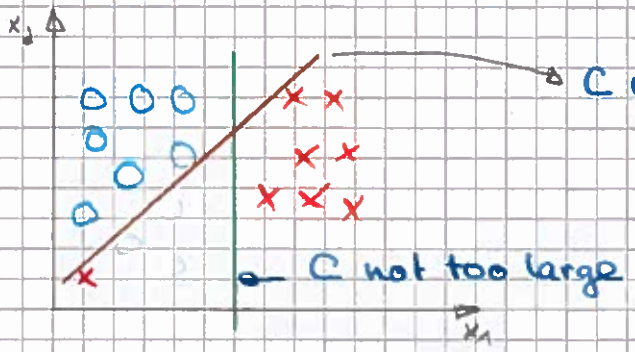
SVM Decision Boundary: linearly separated case



called margin ⇒ gives the SVM a robustness

Large margin classifier

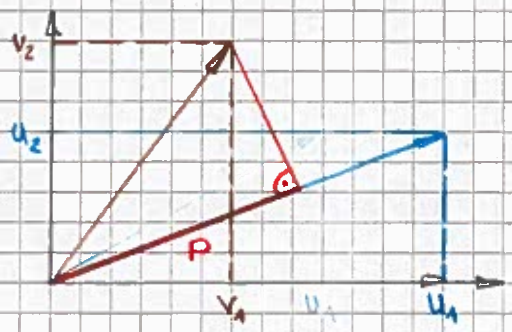
Large margin classifier in presence of outliners



C very large which is not so good
↳ equivalent to $\frac{1}{\lambda}$ $C = \frac{1}{\lambda}$

C not too large

Vector inner Product:



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

$u^T v = ?$

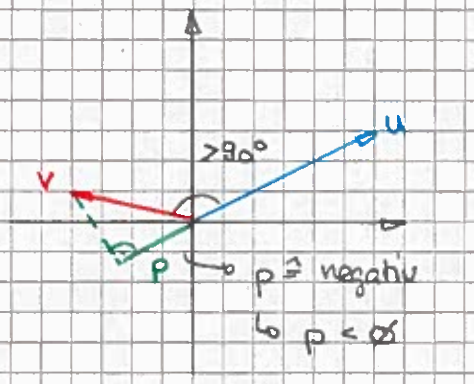
$\|u\|$ = length of vector u

$= \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$

p = length of projection of v onto u

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

$\Rightarrow [u_1 \; u_2] \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = u_1 v_1 + u_2 v_2 \quad p \in \mathbb{R}$



signed

SVM Decission Boundary:

$$\min_\theta \frac{1}{2} \sum_{i=1}^{n} \theta_i^2 \Rightarrow \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2} \cdot \left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 = \frac{1}{2}\|\theta\|^2$$

$\|\theta\|$ = vector u above

$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

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

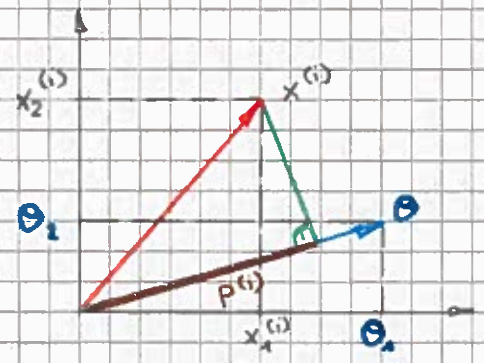
$\theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

For simplification: $\theta_0 = 0 \quad n = 2$

$\theta^T x^{(i)}$

$\uparrow \quad \uparrow$

$u^T v$



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

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

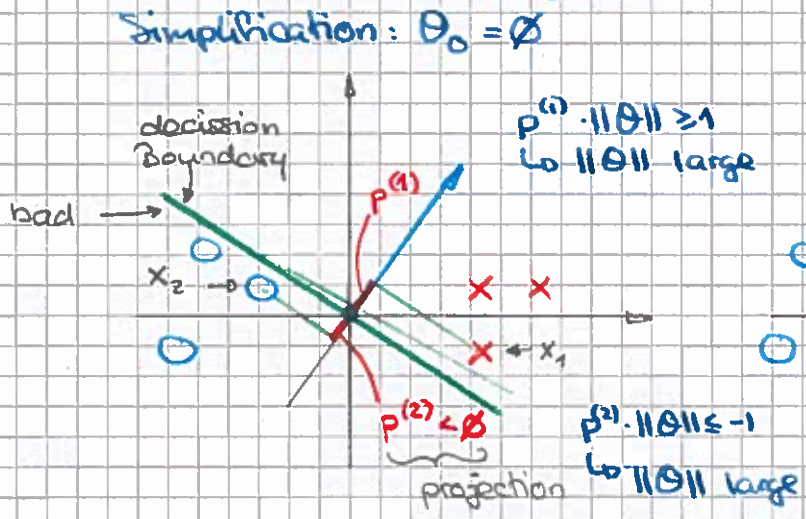
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\theta\|^2$$

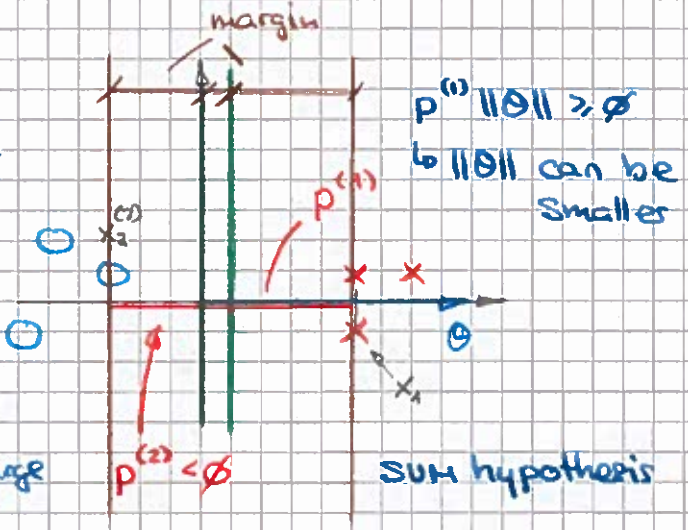
$$p^{(i)} \cdot \|\theta\| \geq 1 \quad \text{if } y^{(i)} = 1$$

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

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

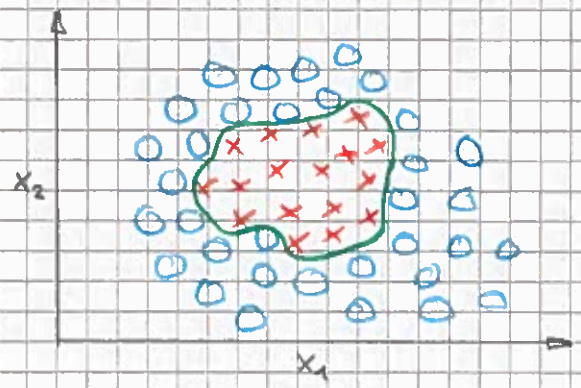
Simplification: $\theta_0 = 0$





⇒ need to find a value for θ close to 0

Non Linear Decision Boundaries:



Predict $y = 1$ if

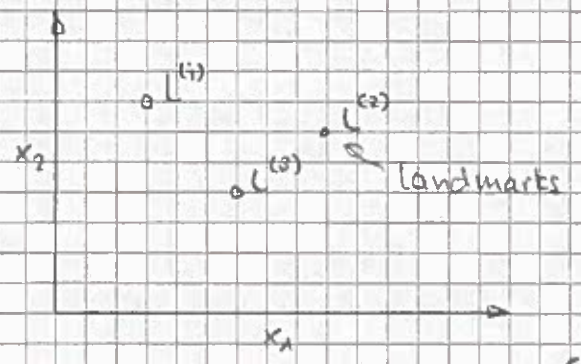
$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \dots \geq 0$

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

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

$f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$ ...

Is there a different / better way of features $f_1, f_2, f_3 \dots$?



~~Given new features~~

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

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

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

$f_3 = \dots$

Gaussian kernel (kernel) $= k(x, l^{(i)})$

kernel function

$\|\cdot\|$ Euclidian distance

Kernels and Similarity:

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

componente wise distance

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

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

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

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

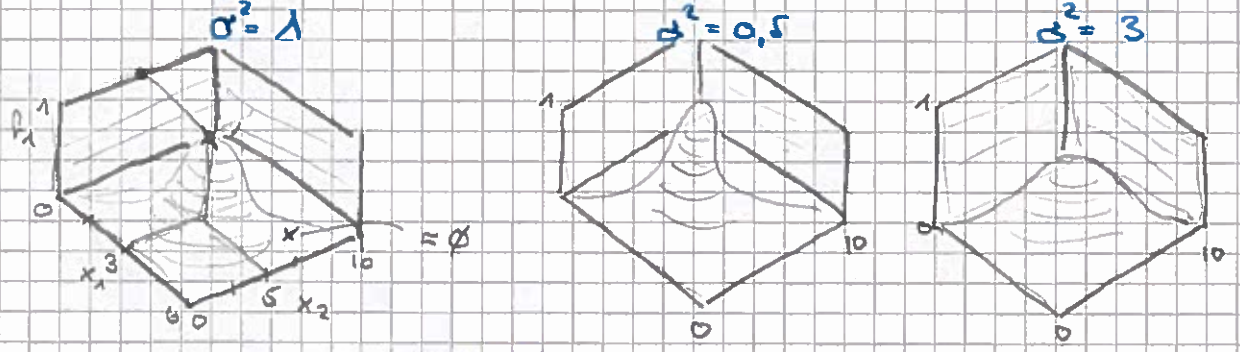
$l^{(1)} \to f_1$
$l^{(2)} \to f_2$
$l^{(3)} \to f_3$

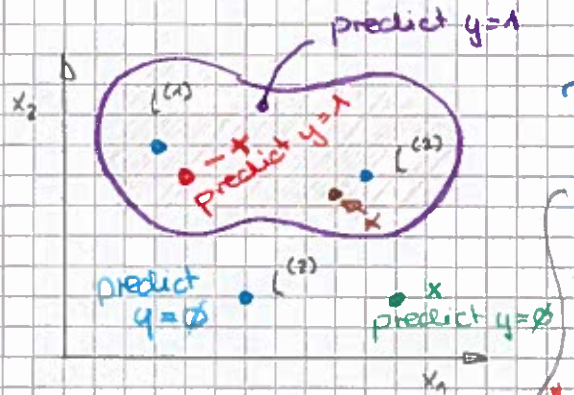
each landmark creates new feature

σ = standard deviation
σ^2 = variance

Example:

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





Predict "1" when

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

$\theta_0 = -0{,}5$, $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = 0$

if x is close to $l^{(1)}$

$f_1 \approx 1$, $f_2 \approx 0$, $f_3 \approx 0$

$$\Rightarrow \theta_0 + \theta_1 \cdot 1 + \theta_2 \cdot 0 + \theta_3 \cdot 0$$

$$= -0{,}5 + 1 = 0{,}5 \geq 0$$

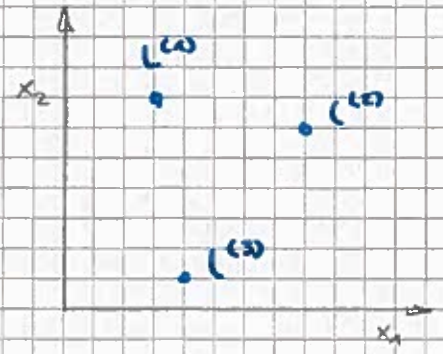
if x is far away from $l^{(i)}$

$f_1, f_2, f_3 \approx 0$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \approx -0{,}5 < 0$$

$$-0{,}5 + 0 + 0 + 0 = -0{,}5$$

Choosing the landmarks:



Given x:

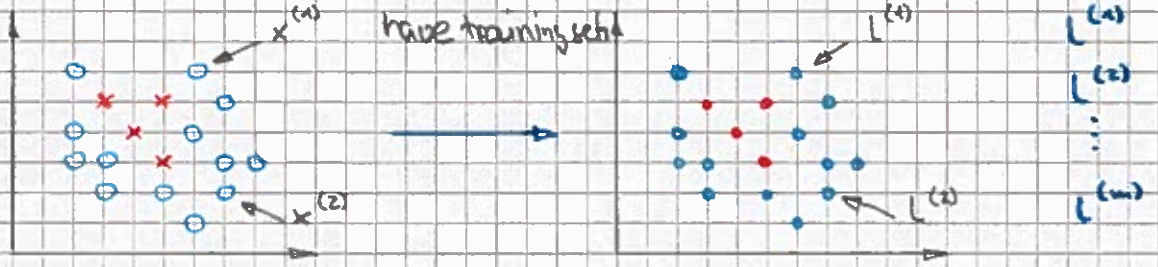
$f_i = \text{similarity}(x, L^{(i)})$

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

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

Where to get $L^{(1)}, L^{(2)}, L^{(3)}$

define as many landmarks as you have training ex.



SVM with kernels:

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

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

Given example x: ← can be training, cv or test set

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

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

$\vdots$

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

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

$x^{(i)} \Rightarrow$

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

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

$\vdots$

$f_i^{(i)} = \text{sim}(x^{(i)}, L^{(i)}) = \exp\left(-\frac{0}{2\sigma^2}\right) = 1$

$\vdots$

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

$L^{(i)} = x^{(i)}$

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

$f_0^{(i)} = 1$

SVM with kernels:

m training set → l landmarks

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

Predict "$y=1$" if $\theta^T f \geq 0$ $= \theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$ $\theta \in \mathbb{R}^{m+1}$

Training:

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

$n = m$

replacement of $\theta^T x^{(i)}$

θ_0 not regularized

$$-\sum_{j=1}^{n} \theta_j^2 = \theta^T \theta \quad \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \text{ (ignore } \theta_0\text{)}$$

$\theta^T \theta \to \|\theta\|^2$

$\theta^T M \theta$ used by some SVM implementation, allows to much bigger Training sets

SVM parameters:

$C (= \frac{1}{\lambda})$.

Large C: Lower bias, high variance (small λ) - overfit

Small C: Higher bias, low variance (large λ) - underfit

higher overfit

not much regularization

higher bias

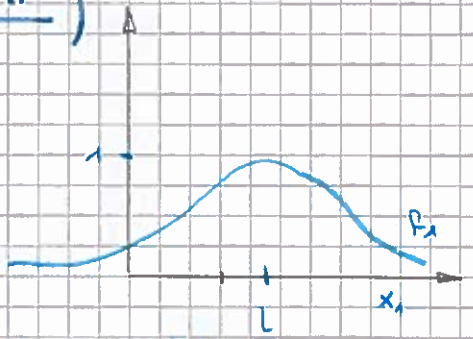
σ^2

Large σ^2: Features f_i vary more smoothly

Higher bias, lower variance

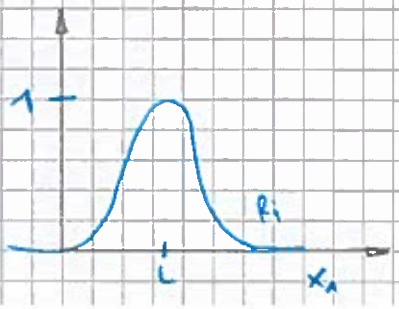
overfit → increase σ^2

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



Small σ^2: Features f_i vary less smoothly

Lower bias, higher variance



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

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

e.g no kernel (linear kernel) $\Rightarrow$ Standard linear classifier

Predict $y=1$ if $\theta^T x \geq 0$ $\Rightarrow$ $\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n \geq 0$

n large, m small $x \in \mathbb{R}^{n+1}$

use no kernel if above applies

↳ risk of overfitting when using Gaussian kernel

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.

$x \in \mathbb{R}^n$, n small and/or n large

Kernel (Similarity) function: ($x^{(i)}$, $l^{(i)} = x^{(i)}$)

```
function f = kernel(x1, x2) {
    f = exp(-||x1 - x2||^2 / (2σ^2))
    return
}
```

$\Rightarrow$ $x \Rightarrow$ $f_1, f_2, \dots, f_m$

Note: Do perform feature scaling before using Gaussian kernel

$\|x - l\|^2$ $\Rightarrow$ $\vec{v} = x - l$ (vector)

$$\|v\|^2 = v_1^2 + v_2^2 + \dots v_n^2$$

$$= (x_1 - l_1)^2 + (x_2 - l_2)^2 + \dots + (x_n - l_n)^2$$

(example: 1000 feat, 1-5 bedrooms)

req. feature scaling

Other choices of kernel:

Note: Not all similarity functions similarity (x, l) make valid kernels. (Need to satisfy technical conditions called "Mercer's Theorem" to make sure SVM packages optimizations run correctly and do not diverge.)

Many off-the-shelf kernels: (performs worse then Gaussian kernel)

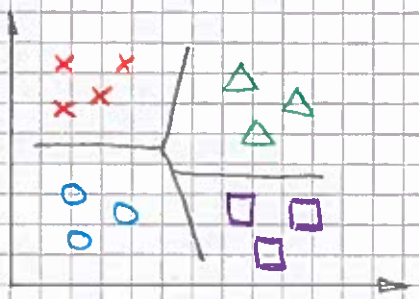
- Polynomial kernel $\Rightarrow k(x, l) = (x^T l) = (x^T l + \text{constant})^{\text{degree}}$

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

- More esoteric:

  String kernel, Chi-square kernel, histogramm intersection kernel, ...

Multi class classification:



$y \in \{1, 2, 3, \ldots, K\}$

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

Otherwise, use one vs. all method. (Train K SVM's, 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 vs. SVM's

n = number of features ($x \in \mathbb{R}^{n+1}$), m = number of training examples

if n is large (relative to m): (eg. $n \geq m$ $n = 10'000$, $m = 10 \ldots 1000$ Spam mails)

Use logistic regression, or SVM without a kernel (linear kernel)

if n is small and m is intermediate ($n = 1 - 1000$, $m = 10 - 10'000$)

Use SVM with a Gaussian kernel

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

Logistic Regression and SVM without a kernel are about the same!

Neural network likely to work well for most of these settings, but may be slower to train.

For SVM not to care about global optimum => usually always finds it

Power of SVM

-