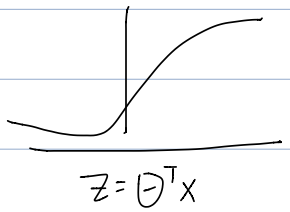
Support Vector Machines

Optimization objective

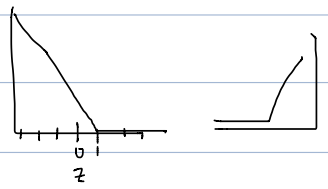
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



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

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

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



SVM: (cost func)

$$\min_\theta \frac{1}{m} C \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) + \frac{1}{2m} \sum_{j=0}^{n} \theta_j^2$$

$A + \lambda B$ (old)

$C A + B$ (new svm) $\quad C = \frac{1}{\lambda}$

hypothesis:

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

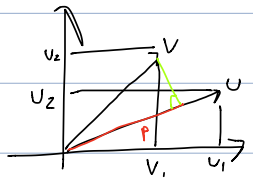
Large Margin Intuition

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)

Mathematics for LM

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



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

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

p = length of projection of v onto u

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

Signed (+/-) $= u_1 v_1 + u_2 v_2$ $\quad p \in \mathbb{R}$

SVM Decision Boundary

$w = (\sqrt{w})^2$

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

$= \|\theta\|$

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

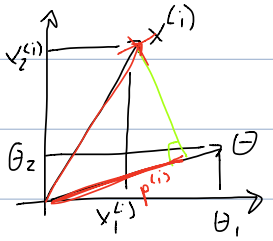
$\theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

$\begin{bmatrix} \cancel{\theta_0} \\ \theta_1 \\ \theta_2 \end{bmatrix}$ $\theta_0 = 0$

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

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

$u^T v$



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

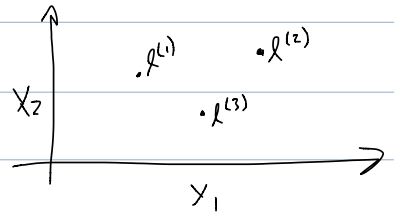
$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$ ⇐

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

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

Non-linear Decision Boundary — KERNELS

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots \geq 0$



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

$f_2 = \text{similarity}(x, l^{(2)}) = \exp \dots$

" ↑ (Kernel (gaussian kernels)) $K(x, l^{(i)})$

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

If $x \simeq l^{(1)}$:

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

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

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

Kernels II

Where to get landmarks? $l^{(1)}$ … $l^{(m)}$

training examples = landmarks

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

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

Given example x:

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

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

$f_2 = \text{sim}(x, l^{(2)})$ — fn

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

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

$\vdots$

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

$x^{(i)} \in \mathbb{R}^{n+1}$ $\quad f_0^{(i)} = 1$ $(\text{or } \mathbb{R}^n)$

$f_1^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp\left(-\frac{0}{2\sigma^2}\right) = 1$ $\quad f^{(i)} = \begin{bmatrix} f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$

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

Predict "$y = 1$" if $\theta^T f \geq 0$ $\quad \theta_0 f_0 + \theta_1 f_1 + \cdots + \theta_m f_m$

need θ — how?

Training:

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

($\theta_0 \leftarrow$ don't reg)

- $\sum_j \theta_j^2 = \theta^T \theta \leftarrow \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$ $[\text{no } \theta_0]$
- $\theta^T M \theta$ $\quad \|\theta\|^2$ $\quad m = 10{,}000$ $\quad \leftarrow$ computational efficiency

$\uparrow$ optimization

SVM parameters:

$C \left(= \frac{1}{\lambda}\right)$ Large C: Lower bias, high variance $\quad (\text{small } \lambda)$

Small C: Higher bias, low variance $\quad (\text{large } \lambda)$

σ^2 Large σ^2: features f_i vary more smoothly

Higher bias, lower variance

Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.

Using an SVM

Use SVM software package (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") if large # features, small # examples

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

Gaussian kernel: if small # features, large # examples

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

Need to choose σ^2 ← Use feature scaling!

Other kernels:

- Polynomial kernel: $k(x, l) = (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$ ... $(x^T l + \text{const})^{\text{degree}}$
- String kernel, chi-square kernel, histogram intersection kernel

Multi-class Classification

One vs. All

Logistic Regression vs SVMs

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

If n is large (relative to m): n ≥ m n=10,000 m=10...10,000

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

If n is small, m is intermediate (n=1-1000, m=10-10,000)

Use SVM with Gaussian kernels

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

(relative/add) more features than use logistic regression or SVM without a kernel

Neural network likely to work well for most of these, may be slow to train