WEEK #7
Support Vector Machines.

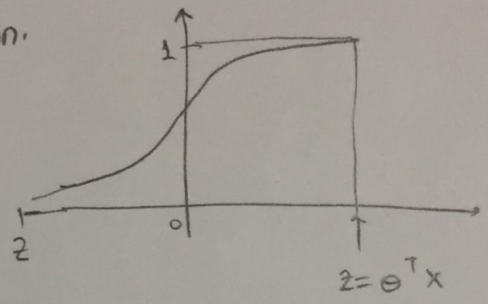
(1.) LARGE MARGIN CLASSIFICATN.

1.1. Obtimizatn Objective

SVM gives a cleaner & powerful way of learning complex non-linear fxns.

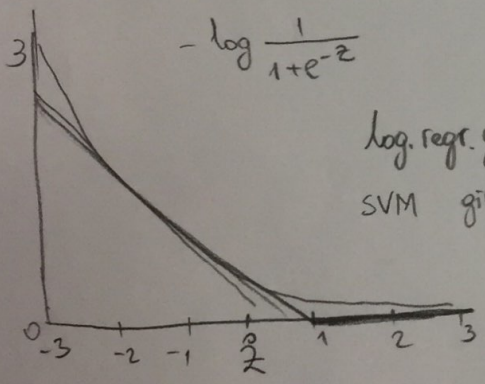
Alternative view of logistic regression.

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



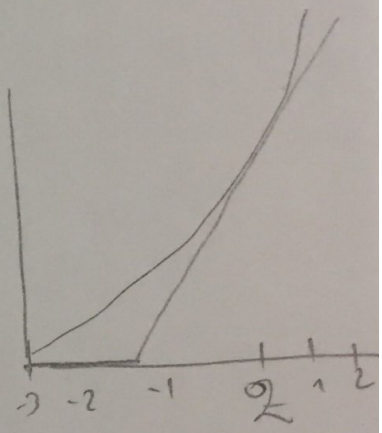
Cost of example: $-(y \log h_\theta(x) + (1-y)\log(1-h_\theta(x)))$

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



log. regr. gives cost of zero when $z \geq 3$

SVM gives cost of zero when $z \geq 1$



Logistic Regression:

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

(A)

SVM:

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

Log. Regr: $A + \lambda B$

SVM: $CA + B$; $C = \frac{1}{\lambda}$ (small)

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)} \ldots\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

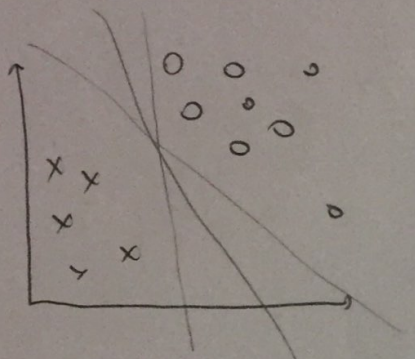
1.2. Large Margin Classifiers:

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

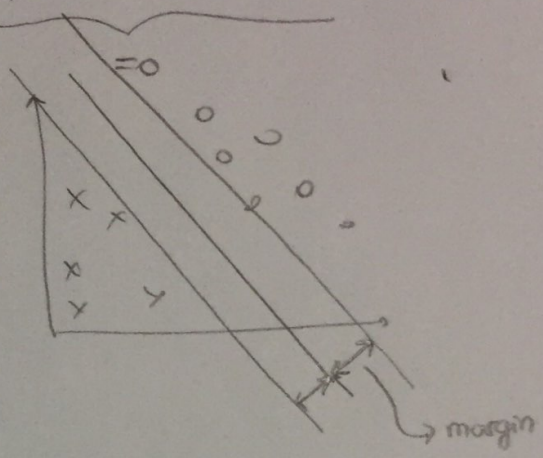
if $y=0$, we want $\theta^T x < -1$ (not just < 0)

when $C = 1000$ (very large)

$$\min_\theta C \sum_{i=1}^{m} [\ \ldots\ldots\ldots\ ] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$



several decision boundaries exist



if C is very large SVM will be sensitive to outliers.

if C is not too large, then SVM will be robust against outliers.

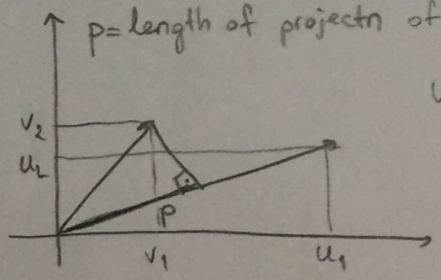
⇒ if the data are not linearly separable ⇒ SVM also do the right thing.

1.3. Mathematical Behind Large Margin Classificatn:

Vector Inner Product.

$u^T v$

$\|u\|$ = length of $u = \sqrt{u_1^2 + u_2^2}$



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

$= u_1 \cdot v_1 + u_2 \cdot v_2$

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

$\|\theta\|$

→ we want minimize $\|\theta\|$, thus we are going to maximize $p^{(i)}$

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

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

$\theta^T x^{(i)} = p^{(i)} \cdot \|\theta\| = \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$

$u^T v$

stands for margin.

② KERNELS

2.1. Kernel 1:

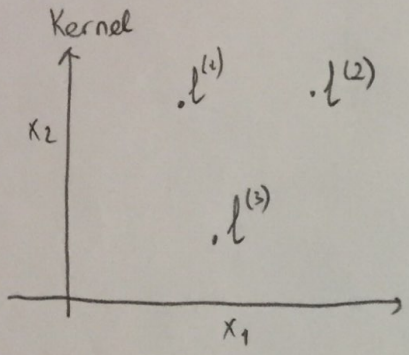
Non-linear decision boundary.

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

$$\theta_1 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 + \theta_5 f_5 + \ldots$$

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

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



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

(Gaussian Kernel:

Kernels & Similarity:

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

if x is far from $l^{(1)}$: $f_1 = \exp\left(-\frac{(\text{large \#})^2}{2\sigma^2}\right) \approx 0$

2.2. Kernel 2:

Choosing the landmarks:
→ put the m-landmarks to the places where the training examples exactly are.

SVM with Kernels:

$$\sum \theta_j^2 = \theta^T \theta = \|\theta\|^2$$

$C = \left(\frac{1}{\lambda}\right)$ Large C: Lower Bias, high variance
Small C: Higher Bias, lower variance

σ^2: Large σ^2: features f_i vary more smoothly → higher bias, lower variance
Small σ^2: " " less " → lower bias, higher variance

(3.) SVMs in Practice

3.1. Using an SVM: [liblinear, libsvm, ...]

→ Choice of paramtr C

→ Choice of kernel (similarity fxn) ex/ no kernel ("linear kernel")

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

n large, m small, do not try nonlinear decision boundary.

→ Choice of σ^2

Other choices of kernel:

- Polynomial kernel $k(x, l) = (x^T l)^2$
- More esoteric: string kernel, chi-square kernel, histogram intersectn kernel

---

if n is large (relative to m) → use logist. regr., or SVM without a kernel ("linear kernel")

if n is small, m is intermediate → use SVM with Gaussian kernel

if n is small, m is large → create/add more features, then use logrst. regr. or SVM without kernel.