Large Margin Classification

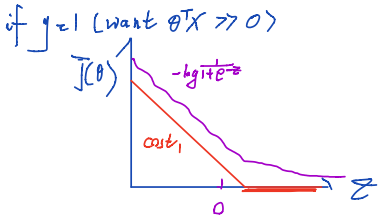
Optimization Objective

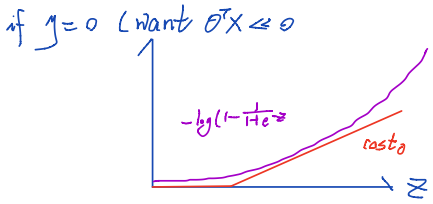
Alternative view of logistic regression

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

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

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





Support Vector machine (SVM)

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

$-\log h_\theta(x^{(i)}) \rightarrow \text{cost}_1(\theta^T x^{(i)})$, $-\log(1-h_\theta(x^{(i)})) \rightarrow \text{cost}_0(\theta^T x^{(i)})$

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

In logistic regression: $A + \lambda B$

In Support Vector machine $CA + B$ $(C = \frac{1}{\lambda})$

Cost function for SVM:

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

hypothesis of SVM

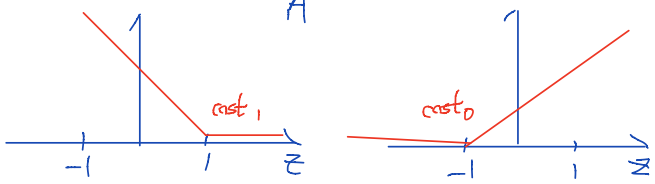
$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ \end{cases}$$

... 1 0 else

Large Margin intuition

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

A



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)

Suppose C very large, $\min_\theta CA + B \Rightarrow A \to 0$

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

$\theta^T x \geq 1$ $\quad \min C \times 0 + \frac{1}{2}\sum_{i=1}^{n} \theta_j^2$

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

$\theta^T x \leq -1$, $\quad \min C \times 0 + \frac{1}{2}\sum_{i=1}^{n} \theta_j^2$

SVM Decision Boundary: linearly seperable case

Large Margin Classifier

C very large

SVM is a Large margin classifier

Mathematics behind Large Margin Classification

SVM decision boundary when C is very large:

(for simplicity, assume $n = 2$, $\theta_0 = 0$)

$$\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}(\theta_1^2+\theta_2^2) = \frac{1}{2}\|\theta\| \qquad \theta=\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}$$

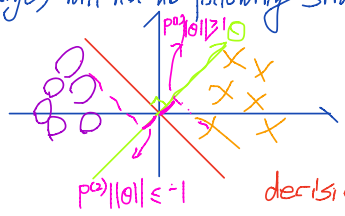
if $y^{(i)}=1$, $\theta^T x^{(i)} \geq 1 \Rightarrow p^{(i)}\cdot\|\theta\| \geq 1$

if $y^{(i)}=0$, $\theta^T x^{(i)} \leq -1 \Rightarrow p^{(i)}\cdot\|\theta\| \leq -1$

where $p^{(i)}$ is $x^{(i)}$ projection on to θ

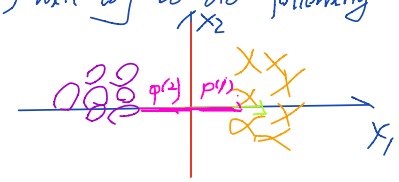
Now prove that SVM with large C with end up with large decision margin (large margin classification)

1) SVM (C large) will not do following small margin:



projection $p^{(i)}$ is small, means $\|\theta\|$ has to be large to make $p^{(1)}\|\theta\|\geq 1$ and $p^{(2)}\|\theta\|\leq -1$, but cost function want $\frac{1}{2}\|\theta\|$ to be small, so SVM (large C) will not end up with this situation

2) SVM (large C) will try to do following



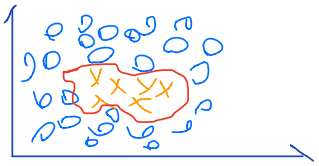
In this case $p^{(1)}$ is larger, to fulfil $p^{(1)}\|\theta\|\geq 1$ and $p^{(2)}\|\theta\|\leq -1$, $\|\theta\|$ doesn't need to be too large, which agrees the minimization of cost function $\min \frac{1}{2}\|\theta\|$

thus SVM (with large C) is a Large Margin Classification

Assumption $\theta_0 = 0$ make θ go through origin of above plot

Kernels

Nonlinear Decision Boundrary



predict $y=1$ if $\theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_1 X_2 + \theta_4 X_1^2 + \cdots \geq 0$

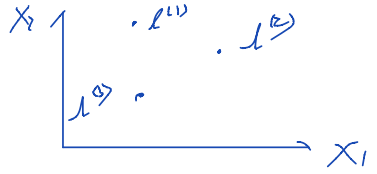
$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \cdots \geq 0 \\ 0 & \text{else} \end{cases}$$

Can rewrite: $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots$

where: $f_1 = X_1$, $f_2 = X_2$, $f_3 = X_1 X_2$ ...

Q: How to optimize ~~the~~ choice of $f_1, f_2, f_3 \cdots$?

Kernel



pick landmarks $l^{(1)}, l^{(2)}, l^{(3)}$, given x, Compute new feature depend on proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$

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

$$f_2 = \text{Similarity}(x, l^{(2)}) = \exp(\cdots)$$

$$f_3 = -$$

Kernel $k(x, l^{(1)})$ Gaussian kernel

Kernels and Similarity

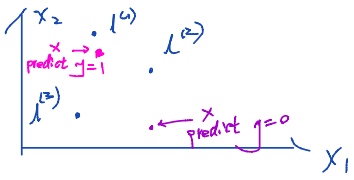
$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$$

if $x \approx l^{(1)}$: $f_1 \approx 1$

if x far from $l^{(1)}$, $f_1 \approx 0$

each of landmarks defines a new feature

Example:



predict 1 when $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 \geq 0$

say $\theta_0 = -0.5$, $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = 0$

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

$\theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0 = -0.5 + 1 = 0.5 \geq 0$ predict 1

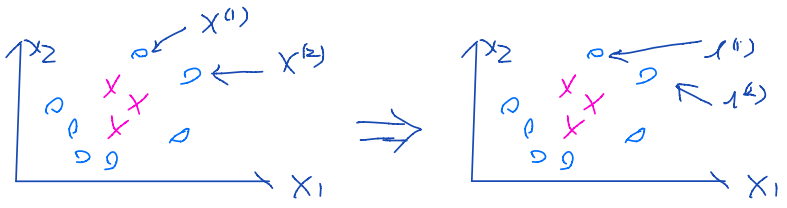
case 2: $f_1, f_2, f_3 \approx 0$

$\theta_0 + \theta_1 \times 0 + \theta_2 \times 0 + \theta_3 \times 0 = -0.5 < 0$ predict 0

Understand: for points close to landmarks, predict 1

for point far away from landmarks, predict 0

Choosing the landmarks



choose landmark exactly where the training data is

m training examples $\Rightarrow$ m landmarks

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

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

Given example x:

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

$f_2 = \cdots\cdots$

$f_m = \cdots\cdots$

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

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

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

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

$\vdots \qquad \vdots \longleftarrow f_i^{(i)} = \text{Sim}(x^{(i)}, l^{(i)}) = 1$

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

then: $x \in \mathbb{R}^{n+1} \rightarrow f^{(i)} = \begin{bmatrix} f_0^{(i)} = 1 \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$ new feature function

SVM with kernels ($n = m$ or $n = m+1$)

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

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

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

In real software/packages, it's actually $\frac{1}{2} \sum_{j=1}^{m} \theta^T M \theta$ instead of $\|\theta\|^2$

where M is some kernel related matrix, in order to let the algorithm scale to much large datasets and computes faster

* SVM with kernel usually runs slow

SVM parameters:

$C(=\frac{1}{\lambda})$: Large C: lower bias, high variance
Small C: higher bias, low variance

σ^2: large σ^2: features f_i vary more smoothly
higher bias, lower variance
Small σ^2: feature f_i vary less smoothly
lower bias, higher variance

Using An SVM

svm software package: liblinear, libsvm
Need to specify:
- choice of parameter C
- choice of kernel

e.g. No kernel ("linear kernel")
predict "y = 1" if $\theta^T x \geq 0$
- good for n (# of features) large, m (# of examples) small
to avoid overfitting

e.g. Gaussian kernel
need to choose σ^2
* good for n small, m large
** Note: Do preform feature scaling before using Gaussian kernel, reason is

$$\|x - l\|^2 = (x_1 - l_1)^2 + (x_2 - l_2)^2 + \cdots + (x_n - l_n)^2$$

then x_2 doesn't show much influence

other choice of kernel:

Note: Not all similarity functions make valid kernels

Need to satisfy "Mercer's Theorem" to make sure SVM packages' optimizations run correctly and do not diverge.

Many off-the-shelf kernels available:

- polynomial kernel: $(x^T l + \text{const})^{\text{degree}}$ (=2, 3, ……)
- string kernel
- chi-square kernel
- histogram intersection kernel

Multi-class classification

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 $i=1, 2, \cdots K$, get $\theta^{(1)}, \theta^{(2)}, \cdots \theta^{(K)}$, pick class i with largest $(\theta^{(i)})^T x$

Logistic regression VS. SVMs

How to make choice in between these two algorithms?

n: number of features ($x \in \mathbb{R}^{n+1}$)

m: number of training examples

- n is large relative to m (e.g. n ≥ m, n = 10,000, n = 10, ... 1000)
  - ↳ Use regression, or SVM w/o a kernel ("linear kernel")
- n is small, m is intermediate (e.g. n = 1–1000, m = 10 – 10,000)
  - ↳ Use SVM w/ Gaussian kernel
- n is small, m is large (n = 1 ~ 1000, m = 50,000 +)
  - ↳ Create/add more features, then use logistic regression or SVM w/o a kernel ("linear kernel")
- neural network likely to work well for most of these settings, but may be slower to train

* SVM is a convex problem, can always find the global min