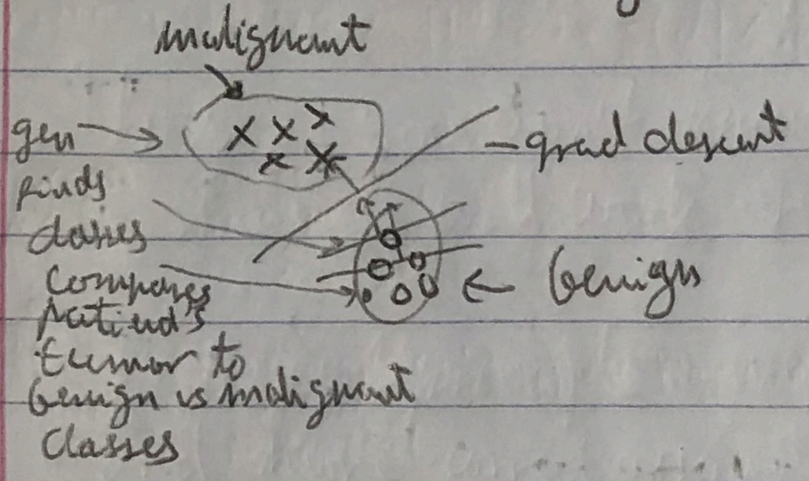
Lecture 5

Today: Generative Reasoning Algorithms
- Gaussian Discriminant Analysis (GDA)
- Generative vs Discriminative comparison
- Naive Bayes



Discriminative: Learn $p(y|x)$ (or learn: $h_\theta(x) = \{0, 1$ directly$\}$)
$x \to y$

Generative: learns $p(x|y)$ (class) $p(y)$ (class prior)
(x: feature)

Bayes Rule: $p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x)}$; $p(x) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)$

Gaussian discr. analysis (GDA)
Suppose $x \in \mathbb{R}^n$ (drop $x_0=1$ convention). Assume $p(x|y)$ is Gaussian.

$z \sim \mathcal{N}(\vec{\mu}, \Sigma)$, $\vec{\mu} \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$, $z \in \mathbb{R}^n$. $E(z) = \mu$ & $Cov(z) = E((z-\mu)(z-\mu)^T)$
$= E zz^T - E z E z^T$

$p(z) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$

GDA model; Assume:

$p(x|y=0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T \Sigma^{-1}(x-\mu_0)\right)$ Parameters: $\mu_0, \mu_1, \Sigma, \phi$.
(Σ: same, assumed typically)

& $p(x|y=1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_1)^T \Sigma^{-1}(x-\mu_1)\right]$

$p(y) = \phi^y (1-\phi)^{1-y}$ $\quad p(y=1) = \phi$.

Training set: $\{x^i, y^i\}_{i=1}^m$

Joint likelihood for generative:

$$\ell(\phi, \mu_0, \mu_1, \Sigma) = \prod_{i=1}^{m} p(x^i, y^i; \phi, \mu_0, \mu_1, \Sigma) = \prod_{i=1}^{m} p(x^i | y^i) p(y^i)$$

& Discriminative: $\ell(\theta) = \prod_{i=1}^{m} p(y^i | x^i; \theta)$ (conditional likelihood)

Maximum Likelihood Estimation:

$\max_{\phi, \mu_0, \mu_1, \Sigma} \underbrace{\log \ell(\dots)}_{\ell(\phi, \mu_0, \mu_1, \Sigma)} \Rightarrow$

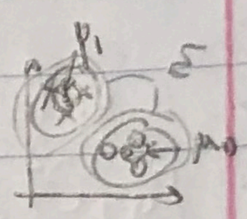
$$\phi = \sum_{i=1}^{m} y^i / m = \sum_{i=1}^{m} 1\{y^i = 1\} / m$$

$$\mu_0 = \sum_{i=1}^{m} 1\{y^i = 0\} x^i / \left(\sum_{i=1}^{m} 1\{y^i = 0\}\right)$$

$$\mu_1 = \sum_{i=1}^{m} 1\{y^i = 1\} x^i / \sum_{i=1}^{m} 1\{y^i = 1\}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^i - \mu_{y^i})(x^i - \mu_{y^i})^T$$

↖ indicator function: $1\{\text{true}\} = 1$, $1\{\text{false}\} = 0$.



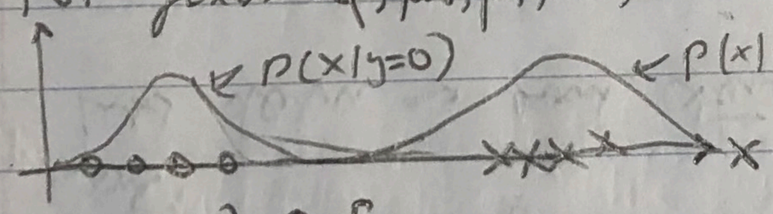
Prediction:

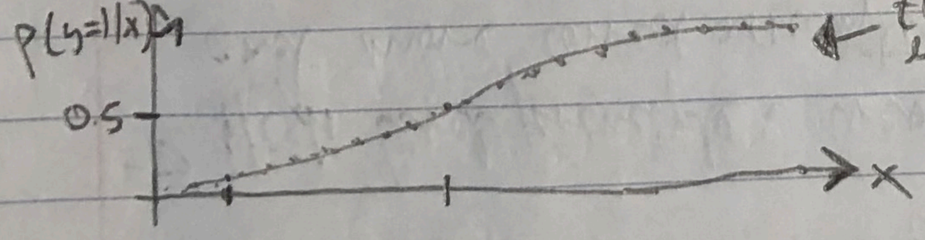
$$\arg\max_y p(y|x) = \arg\max_y \frac{p(x|y)p(y)}{p(x)}$$
$$= \arg\max_y p(x|y)p(y)$$

Compare GDA to logistic regression

For fixed $\phi, \mu_0, \mu_1, \Sigma$, lets plot $p(y=1 | x; \phi, \mu_0, \mu_1, \Sigma)$ as a function of x.

$$= \frac{p(x|y=1; \mu_1, \dots) p(y=1; \phi)}{p(x; \phi, \mu_0, \dots)} \quad (= \phi)$$





(generative) GDA assumes

$$\left.\begin{aligned} x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \\ y &\sim \text{Ber}(\phi) \end{aligned}\right\} \Rightarrow \not\Leftarrow$$

(discriminative) logistic regression

$$p(y=1|x) = \frac{1}{1 + e^{-\theta^T x}} \quad (\text{"}x_0 = 1\text{"}) \quad (\text{logistic})$$

Thus GDA makes stronger assumptions than logistic reg.

Note: In general, if your make stronger assumptions that are correct, your model will do better (you exploit structure). However, if they aren't, it'll do worse.

$X|y=1 \sim \text{Poisson}(\lambda_1)$

$X|y=0 \sim \text{Poisson}(\lambda_0)$

$y \sim \text{Bern}(\phi)$

$\Rightarrow p(y=1|x)$ is logistic

(or some other exponential family)

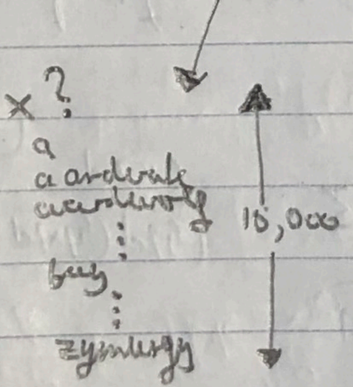
Thus if you don't know whether your data is Gaussian or Poisson, logistic regr. will work regardless.

$\Rightarrow$ High level principle: weaker assumptions → more robust to weaker assumption, and more correct assumptions = more exploitation of structure = better model accuracy. Algo has two sources of "truth": data & assumptions.

Important principle: Less data → need extensive assumptions more important.

Naive Bayes

Feature vector x?



$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$ $x \in \{0,1\}^n$

$x_i = 1\{\text{word } i \text{ appears in email}\}$

Want to model $p(x|y)$, $p(y)$

~~$2^{10,000}$~~ $2^{10,000} - 1$ possible values for x

Assume x_i's are conditionally independent given y. $p(x_1, \ldots, x_{10,000}|y) = p(x_1|y)$ fact

$p(x_2|x_1, y)\, p(x_3|x_1, x_2, y) \cdots p(x_{10,000}|\ldots)$

$\overset{\text{assume}}{=} p(x_1|y)\, p(x_2|y)\, p(x_3|y) \cdots p(x_{10,000}|y)$ ← Naive Bayes assumption. Not always great, but sometimes works.

$= \prod_{i=1}^{n} p(x_i|y)$

Parameters:

$\phi_{j|y=1} = p(x_j=1|y=1)$, $\phi_{j|y=0} = p(x_j=1|y=0)$, $\phi_y = p(y=1)$

Joint likelihood: $\mathcal{L}(\phi_y, \phi_{j|y}) = \prod_{i=1}^{m} p(x^i, y^i; \phi_y, \phi_{j|y})$.

MLE:

$$\phi_y = \sum_{i=1}^{m} \mathbb{1}\{y^i=1\}/m$$

$$\phi_{j|y=1} = \sum_{i=1}^{m} \mathbb{1}\{x_j^i=1, y^i=1\} / \sum_{i=1}^{m} \mathbb{1}\{y^i=1\}$$

It's not too shabby in practice... And extremely efficient.

Lecture 6

Outline:

- Naive Bayes
  - Laplace smoothing
  - Event models
- Comments on applying ML
- SVM intro

Recap:

$$x = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix} \begin{matrix} \text{a} \\ \text{aardvark} \\ \vdots \\ \text{buy} \\ \vdots \end{matrix} \quad x_j = 1\{\text{word } j \text{ appears in email}\}$$

Generative model: $\boxed{p(x|y)}$ $\boxed{p(y)}$

$p(x|y) = \prod_{j=1}^{n} p(x_j|y)$

Parameters: $p(y=1) = \phi_y$, $p(x_j=1|y=0) = \phi_{j|y=0}$,
$p(x_j=1|y=1) = \phi_{j|y=1}$

We found:

$$\phi_y = \sum_{i=1}^{m} 1\{y^i = 1\} / m$$

$$\phi_{j|y=0} = \sum_{i=1}^{m} 1\{x_j^i = 1, y^i = 0\} \Big/ \sum_{i=1}^{m} 1\{y^i = 0\}$$

At prediction time:

$$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1) + p(x|y=0)p(y=0)}$$

(major assumption)

Example: NIPS $\leftrightarrow j = 6017$ ($\Leftrightarrow$ spam = true)

$$\Rightarrow p(x_{6017}=1|y=1) = \frac{0}{\#\{y=1\}} = \phi_{6017|y=1}$$

$$p(x_{6017}=1|y=0) = \frac{0}{\#\{y=0\}} = \phi_{6017|y=0}$$ [Haven't seen this word yet.] ... i.e. $p_{6017} = 0$

$\Rightarrow$ When you suddenly start seeing NIPS in your emails, $p(x|y=1) = \prod_{i=1}^{10{,}000} p(x_i|y) = 0$ and so will $p(x|y=1)$ & $p(x|y=0)$, in denominator of $p(y=1|x) \Rightarrow 0/0$ problem.

$\Rightarrow$ Addres this problem with "Laplace Smoothing". But before that: an example.

		Won?
9/12	Wakeforest	0
10/10	Oregon State	0
10/17	Arizona	0
11/21	Caltech	0
12/31	Oklahoma	? ← what's the estimate for this? ← likely lost in the end lol

$P(x=1) = \frac{\#\text{"1"s}}{\#\text{"1"s} + \#\text{"0"s}} = \frac{0}{0+4} = 0$. ← not a good idea to estimate this.

Laplace smoothing: just add 1 to wins & losses, giving mathematically

$\underline{P(x=1)} = \frac{1}{1+5} = \underline{\underline{1/6}}$. Note: This trick is actually optimal in some Conditions.

More generally: $x \in \{1, \dots, k\}$ Estimate $p(x=j) = \frac{\sum_{i=1}^{m} 1\{x^i = j\} + 1}{m + k}$

$\phi_{j|y=0} = \left(\sum_{i=1}^{m} 1\{x_j^i = 1, y^i = 0\} + 1\right) / \left(\sum_{i=1}^{m} 1\{y^i = 0\} + 2\right)$ ← removes 0/0 problem

$x_i \in \{1, \dots, k\}$ rule of thumb is 10 buckets, else, not 4.

Size	<400 sqft	400-800	800-1200	>1,200
x	1	2	3	4

$P(x|y) = \prod_{i=1}^{4} p(x_j|y)$ multinomial

"Multivariate Bernoulli event model"

So far: $x = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ aardvark ← 1 "Drugs! Buy drugs now!" $x_i \in \{0, 1\}$
aardvark ← 2
buy ← 800
drugs ← 1600
now ← 6200

Multinomial event model

New rep: $x \in \begin{bmatrix} 1600 \\ 800 \\ 1600 \\ 6200 \end{bmatrix} \in \mathbb{R}^n$ $x_j = \{1, \dots, 10'000\}$

n_i = length of email i

$$P(x,y) = P(x|y)P(y) \overset{\text{assume}}{=} \prod_{j=1}^{n} P(x_j|y)$$ — how a "multinomial" instead of "Bernoulli" probability

Parameters: $\phi_{y=1} = P(y=1)$, $\phi_{k|y=0} = P(x_j = k | y=0)$ — change word k being k in $j=0$.

assume independent of j (location)

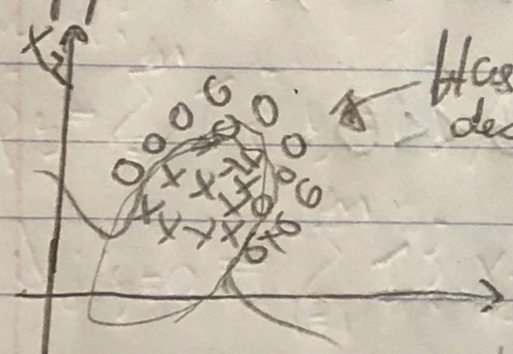
$\phi_{k|y=1} = P(x_j = k | y=1)$

MLE: $\phi_{k|y=0} = \left(\sum_{i=1}^{m} 1\{y^i=0\} \sum_{j=1}^{n_i} 1\{x_j^i = k\} + 1\right) \Big/ \left(\sum_{i=1}^{m} 1\{y^i=0\} \cdot n_i + 10{,}000\right)$

(non-spam; number of possible words)

Naive Bayes vs Logistic regression in practise:
LogReg typically performs better, but NaiveBayes much simpler to implement and far more computationally efficient (no need for eg gradDesc).
Naive Bayes is a nice "quick&dirty" starting point when investigating data.

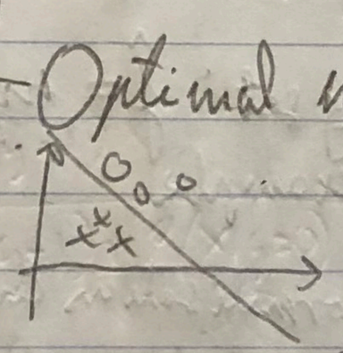
Support Vector Machines



Has a nonlinear decision boundary

SVM maps to higher dim features: $\phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_1 \\ x_1 x_2 \\ x_2 x_2 \end{bmatrix}$ & runs some linear classification

- Optimal margin classifier (seperable case) } today
- Kernel: $x \in \mathbb{R}^2 \to \psi(x) \in \mathbb{R}^5, \mathbb{R}^{10000}$ or even $\mathbb{R}^\infty$
- Inseperable case } next time

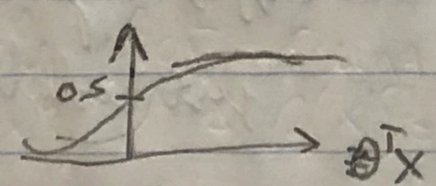


Functional Margin

$h_\theta(x) = g(\theta^T x)$

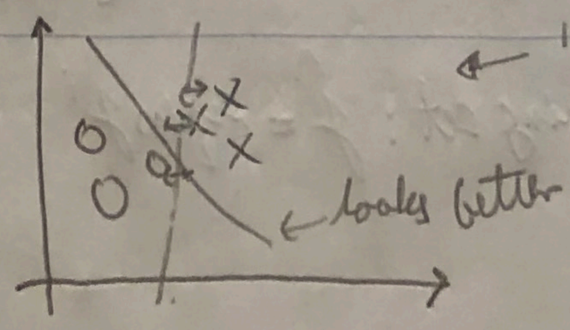
Predict "1" if $\theta^T x \geq 0$ ($g(\theta^T x) \geq 0.5$)

$\not\geq 0$ otherwise



If $y^i = 1$, hope that $\theta^T x^i >> 0$ & if $y^i = 0$, hope $\theta^T x^i << 0$

- Geometric Margin:



← "optMarClas." want to find the best line, ie geometric margin optimized

← looks better

Notation:

Labels $y \in \{-1, +1\}$. Have h output values in $\{-1, +1\}$

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ -1 & \text{otherwise} \end{cases}$$

Previously: $\theta \in \mathbb{R}^{n+1}$, $x_0 = 1$

$h_\theta(x) = g(\theta^T x)$

Now: $\mathbb{R}^n$ $\mathbb{R}$

$h_{w,b}(x) = g(\underbrace{w^T x + b}_{\sum_{i=1}^{n} w_i x_i + b})$

Drop the $x_0 = 1$ constraint.

Eg: $\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$ $\begin{matrix} \} b \\ \} w \end{matrix}$

Functional margin of hyperplane defined by (w, b) wrt (x^i, y^i):

$\hat{\gamma}^i = y^i (w^T x^i + b)$. If $y^i = 1$, want $w^T x^i + b \gg 0$; &

If $y^i = -1$, want $w^T x^i + b \ll 0$.

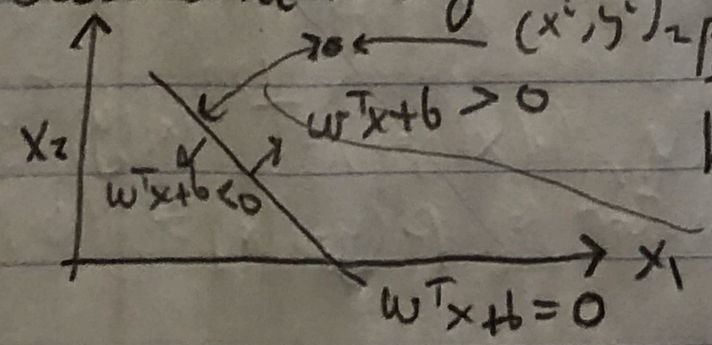
$\Rightarrow$ Want $\hat{\gamma}^i \gg 0$. If $\hat{\gamma}^i > 0$, then $h(x^i) = y^i$

Functional margin wrt training set: $\hat{\gamma} = \min_i \hat{\gamma}^i$, $i = 1, \ldots, m$

Ie its defined by the "worst" training example

$|w| = 1$; $(w, b) \rightarrow \left(\frac{w}{|w|}, \frac{b}{|b|}\right)$ Nice normalisation convention

Geometric Margin



Geometric margin of hyperplane (w, b) wrt (x^i, y^i): $\gamma^i = \frac{w^T x^i + b}{|w|}$

just Euclidian distance

Turns out: $\gamma^i = \frac{\hat{\gamma}^i}{|w|}$ [note: $\hat{\gamma}^i$ = functional margin, γ = geometric margin]

Geometric margin wrt training set: $\hat{\gamma} = \min_i \gamma^i$

Optimal margin classifier:
Choose w, b to maximise γ.

$$\max_{\gamma, w, b} \gamma \quad \text{s.t. } y^i(w^T x^i + b)/|w| \geq \gamma, \; i = 1, \dots, m$$

nonconvex

This is convex

$$\min_{w,b} |w|^2 \quad \text{st } y^i(w^T x + b) \geq 1$$

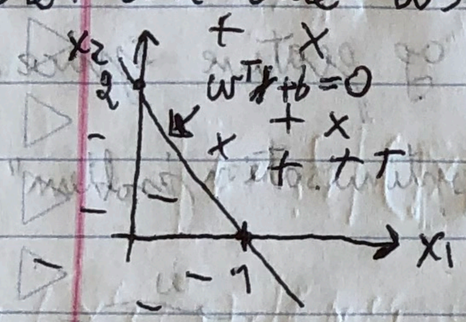
examples

Lecture 7 : SVMs. Optimisation problem; Representer theorem; kernels.

$\gamma^i = \frac{y^i(w^T x^i + b)}{|w|}$ (geometric margin) ; $\gamma = \min_{i=1,\dots,n} \gamma^i$; $y = g(w^T x + b)$ classifier

$\max_{\gamma, w, b} \gamma$ st $\frac{y^i(w^T x^i + b)}{|w|} \geq \gamma$ ← equiv. to "every example has GM $\geq \gamma$"

Note: Can scale w, b without affecting decision boundary. Eg, take $y = g([2\ 1]^T x - 2) = g([20\ 10]^T x - 20)$.



$\Rightarrow$ Optimisation objective becomes $\max \frac{1}{|w|}$ st $y^i(w^T x^i + b) \geq \gamma$

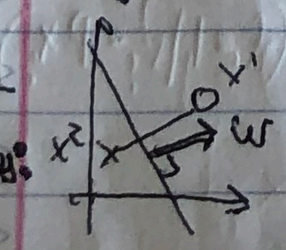
$\Leftrightarrow \min_{w,b} \frac{1}{2}|w|^2$ st $y^i(w^T x^i + b) \geq 1, i = 1, \dots, n$

Say $x^i \in \mathbb{R}^{100}$. Suppose $w = \sum_{i=1}^n \alpha_i x^i$ (y^i) ← ± 1. The "Representer theorem" proves this to be an optimal value, best proof is quite involved. Here's an intuitive explanation:

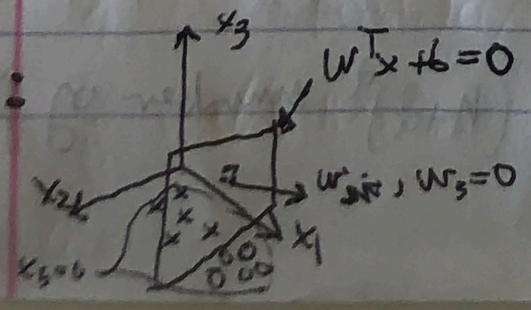
Intuition #1: Logistic regression: $\theta_0 = 0$, then $\theta := \theta - \alpha(h_\theta(x^i) - y^i)x^i$. Thus θ is a linear combination of the training examples. (α: learning rate) $\Rightarrow \begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_n\end{bmatrix} \leftrightarrow \begin{bmatrix}b\\ w_1\\ \vdots\\ w_n\end{bmatrix}$

Also true for batch gradient descent: $\theta := \theta - \alpha \sum_{i=1}^n (h_\theta(x^i) - y^i) x^i$

#2: Eg: $g([2\ 1]^T x - 2)$ (w^T, b) $\Rightarrow$ w is always normal to decision boundary. Then w lies in the span of the training examples.



#3:



$\Rightarrow$ w is correctly placed in the span of the training examples i.e its solely represented in the features $x_1 \& x_2$, where the training examples lie.

So, let's assume $w = \sum_{i=1}^{n} \alpha_i y^i x^i$. $\min_{w,b} \frac{1}{2}\underbrace{\|w\|^2}_{=w^T w}$ st $y^i(w^T x^i + b) \geq 1 \; \forall i$.

$\Leftrightarrow$ $\min \frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y^i x^i\right)^T \left(\sum_{j=1}^{n} \alpha_j y^j x^j\right) = \min \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^i y^j \underbrace{x^{iT} x^j}_{\langle x^i, x^j \rangle}$

[$\langle x, z \rangle = x^T z$ is the inner product]

While the limitation becomes: $y^i(\sum_j \alpha_j \underbrace{\langle x^j, x^i \rangle}_{} + b) \geq 1$ $\Rightarrow$ feature vectors only ever appear in inner product, and so we only need to be able to evaluate it very efficiently, regardless of the dimensionality of feature space.

Anyways, can simplify that optimisation problem to:

"Dual optimisation problem"

$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_i \sum_j y^i y^j \alpha_i \alpha_j \langle x^i, x^j \rangle$ st $\alpha_i \geq 0$, $\sum_{i=1}^{n} y^i \alpha_i = 0$

To predict: 1) Solve for α's, b; 2) Predict by: $h_{w,b}(x) = g(w^T x + b) = g(\underbrace{(\sum_i \alpha_i y^i x^i)^T}_{=w} x + b)$

$= g(\sum_i \alpha_i y^i \langle x^i, x \rangle + b)$.

Kernel trick:

1) Write algorithms in terms of $\langle x^i, x^j \rangle$ ($\langle x, z \rangle$)
2) Let there be some mapping from $x \longmapsto \phi(x)$
3) Find way to compute $K(x,z) = \phi(x)^T \phi(z)$
4) Replace $\langle x, z \rangle$ in algorithm with $K(x,z)$.

Let $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^n$, $\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ \vdots \\ x_3 x_3 \end{bmatrix} \in \mathbb{R}^{n^2}$, $\phi(z) = \begin{bmatrix} z_1 z_1 \\ z_1 z_2 \\ z_1 z_3 \\ z_2 z_1 \\ \vdots \\ z_3 z_3 \end{bmatrix}$ n^2 elements $\Rightarrow$ need $O(n^2)$ time to compute $\phi(x)$, or $\phi(x)^T \phi(z)$ explicitly.

$K(x,z) = \phi(x)^T \phi(z) = (x^T z)^2$ is an $O(n)$ time computation.

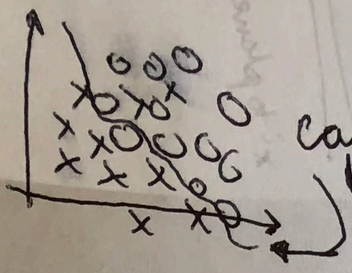
$= (\sum_{i=1}^{n} x_i z_i)(\sum_{j=1}^{n} x_j z_j) = \sum_{i,j} x_i z_i x_j z_j = \sum_{i,j} (x_i x_j)(z_i z_j)$. $\Rightarrow$ Go from $O(n^2)$ to just $O(n)$!

($\mathbb{R}^n$, $\mathbb{R}^n$, $x^T z$, $(\cdot)^2$)

Another ex: $K(x,z) = (x^T z + c)^2$, $c \in \mathbb{R}$ $\Rightarrow$ $\phi(x) \rightarrow \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_3 x_3 \\ \sqrt{2c}\, x_1 \\ \sqrt{2c}\, x_2 \\ \sqrt{2c}\, x_3 \\ c \end{bmatrix}$ and ditto for $\phi(z)$

Or: $K(x,z) = (x^T z + c)^d$ $\Rightarrow$ $\phi(x)$ has all $\binom{n+d}{d}$ features of monomials up to order d. $\sim (n+d)^d$ number of features

Optimal margin classifier + kernel trick = SVM.



can learn very nonlinear decision boundaries.

How to make kernels. Heuristic:
If x, z are "similar", then $K(x,z) = \phi(x)^T \phi(z)$ is "large". Else if x, z are "dissimilar", $K(x,z)$ should be small. Ie, $\phi(x)$ and $\phi(z)$ are [illegible] is

$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$, eg.

Does there exist ϕ st $K(x,z) = \phi(x)^T \phi(z)$? This is the goal. Must satisfy: $K(x,x) \geq 0$.

Theorem outlining when a given kernel is appropriate:

Let $\{x^1, \ldots, x^d\}$ be d points, and let

$K \in \mathbb{R}^{d \times d}$ "kernel/Gram matrix"

$K_{ij} = K(x^i, x^j)$. Given any vector z,

$$z^T K z = \sum_{i,j} z_i K_{ij} z_j = \sum_{i,j} z_i \phi(x^i)^T \phi(x^j) z_j$$

$$= \sum_{i,j} z_i \sum_k [\phi(x^i)]_k [\phi(x^j)]_k z_j = \sum_{k,i,j} z_i [\phi(x^i)]_k$$

$$[\phi(x^i)]_k z_j = \sum_k \left(\sum_i z_i \phi(x^i)_k\right)^2 \geq 0 \Rightarrow K \text{ is psd}$$

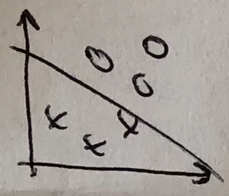
Theorem (Mercer): K is a valid kernel function (ie $\exists \phi$ st $K(x,z) = \phi(x)^T \phi(z)$) iff for any d points $\{x^1, \ldots, x^d\}$, the corresponding kernel matrix K is psd. The above algebra proves this theorem. The above algebra $K(x,z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$, Gaussian kernel, does indeed turn out to be a valid kernel (and is commonly used).

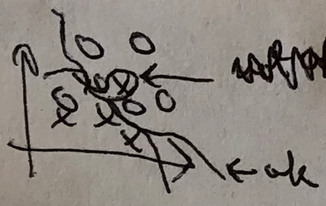
Most widely used kernel is the linear kernel.

$k(x,z) = x^T z$; $\phi(x) = x$. (Trivial).

After that, the Gaussian kernel is the most widely used, whose $\phi(x) \in \mathbb{R}^\infty$. Uses all monomial features: $x_1, x_1x_2, x_1^2x_2$, ... etc.

Basically, allows us to do linear regression in infinite-dimensional feature spaces.

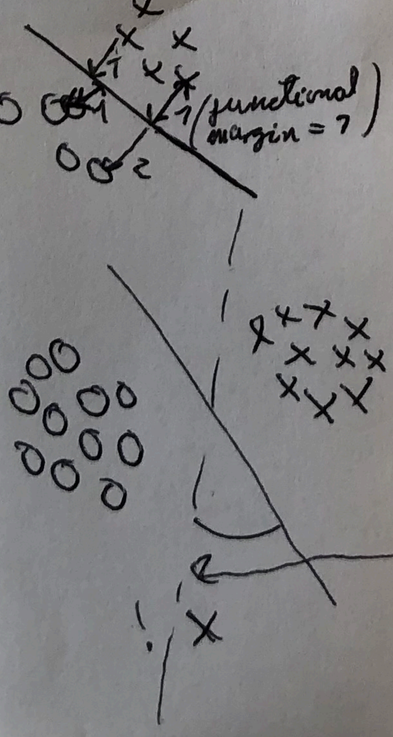




L_1 norm soft margin SVM

$$\min \frac{1}{2}|w|^2 + C\sum_{i=1}^{m} \xi_i \quad \text{s.t.} \quad \underbrace{y^i(w^T x^i + b)}_{\text{functional margin}} \geq 1 - \xi_i, \quad i = 1, \ldots, m$$

where $\xi_i \geq 0$



"Setting ξ_i to eg 0.5 lets you get away with having a functional margin a bit less than 1".

auto margin classifier optimises for the worst case margin, and so will cause a dramatic swing in the decision boundary even just due to a single outlier.

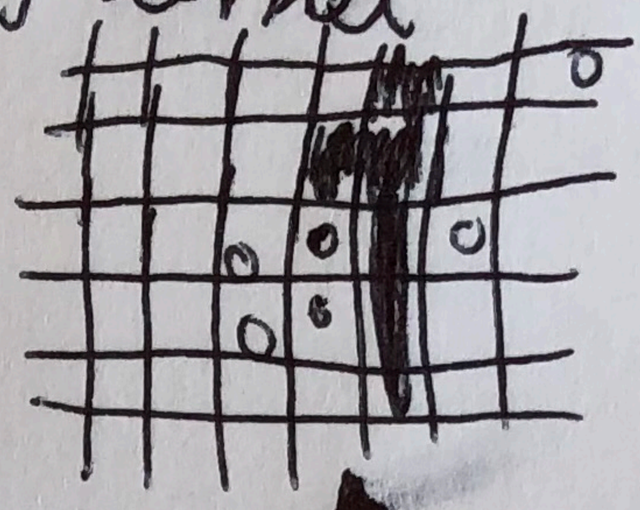
Soft margin SVM is less vulnerable to this.

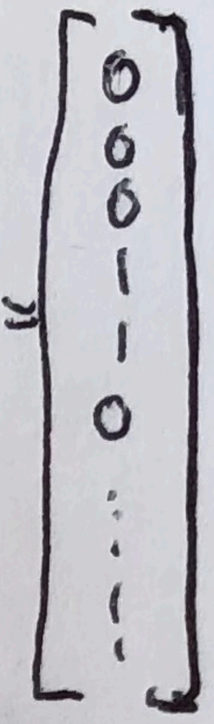
$$\max \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m}\sum_{j=1}^{m} y^i y^j \alpha_i \alpha_j \langle x^i, x^j \rangle \quad \text{s.t.} \quad \sum_{i=1}^{m} y^i \alpha_i = 0,$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m$$

$k(x,z) = (x^T, z)^d$ poly kernel

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





SVMs hold the record for best image classification for a while.

Protein Sequence classifica

A.....Z

BAJTSDAJBAJTAU....

$\phi(x) = ?$ $x \longmapsto y$

AAAA
AAAB
AAAC
AAAD
:
AAAZ
:
BAJT
TSTA
ZZZZ

$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 2 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \phi(x)$,

20^4 length

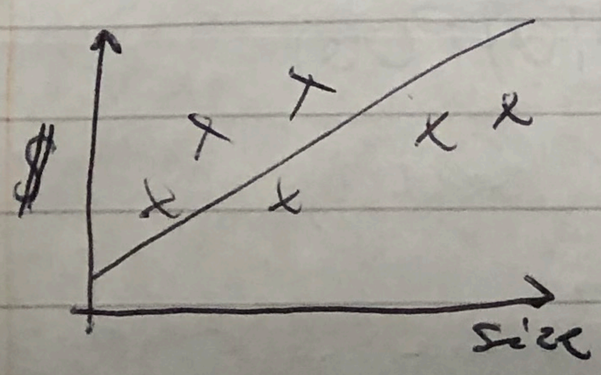
$\phi(x)^T \phi(z) = k(x,z)$

There is a dynamic programming algo to compute this Kernel.

Lecture 8

Outline
- Bias/variance
- Regularisation
- Train/dev/test splits
- Model selection & cross-validation

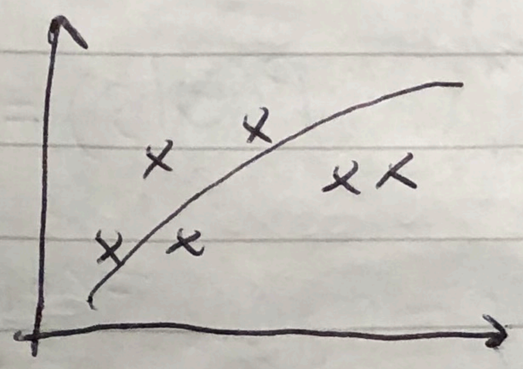
Bias/variance:



$\theta_0 + \theta_1 x$

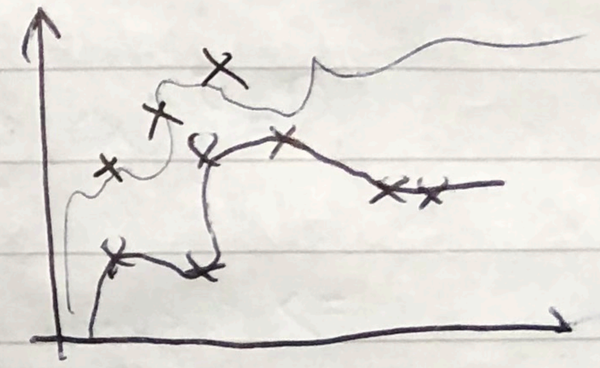
"underfit"; high bias

bias = "strong preconception of model", want to avoid



$\theta_0 + \theta_1 x + \theta_2 x^2$

"just right"

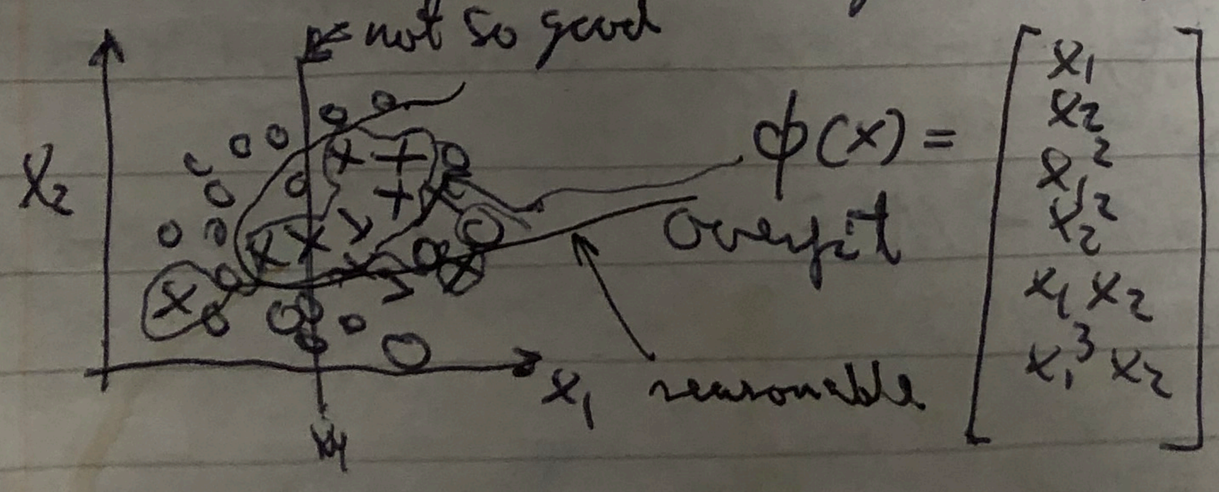


$\theta_0 + \dots \theta_5 x^5$

overfit; high variance

For new data, this algorithm would get a completely different result!

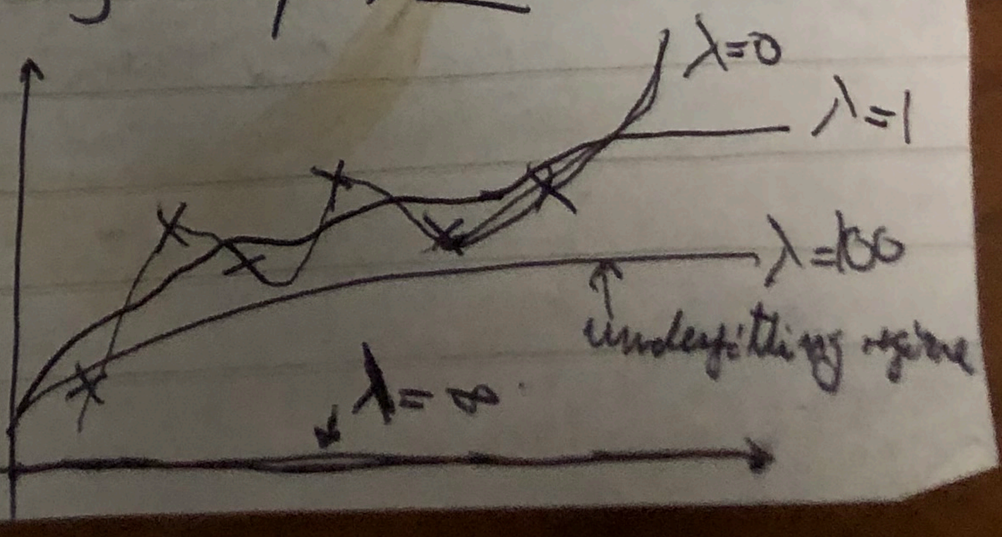
Also hold true in classification problems



Regularisation: sounds deceptively simple, but extremely widely used.

$$\min_\theta \frac{1}{2} \sum_{i=1}^{m} |y^i - \theta^T x^i|^2 + \underbrace{\frac{\lambda}{2} |\theta|^2}_{\text{regul. term}}$$

penalising large θ_k terms & makes overfitting harder



$$\arg\max_{\Theta} \sum_{i=1}^{m} \log p(y^i | x^i; \Theta) - \frac{\lambda}{2}|\Theta|^2$$

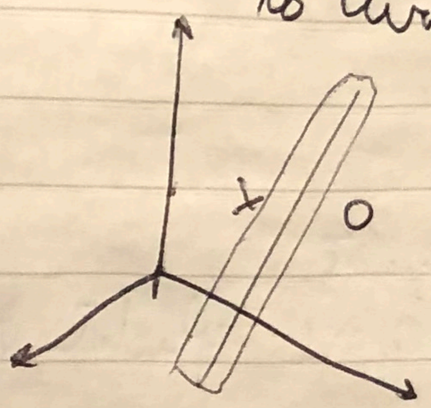
Note: SVM don't overfit too much despite working in extremely large feature spaces because $|w|^2 \propto |\theta|^2$ but formal proof quite complicated.

Text classification:

$m = 100$; $n = 10{,}000$, $X = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ \vdots \end{bmatrix}$ standards

* examples * features

Need regularization here to avoid (severe) overfitting.



Note: always ensure features are scaled similarly during preprocessing before applying regularization.

$$S = \{(x^i, y^i)\}_{i=1}^{m} \qquad P(\Theta|S) = \frac{P(S|\Theta)P(\Theta)}{P(S)}$$

$$\Rightarrow \arg\max_{\Theta} P(\Theta|S) = \arg\max_{\Theta} P(S|\Theta)P(\Theta)$$

$$= \arg\max_{\Theta} \left(\prod_{i=1}^{m} P(y^i | x^i, \Theta)\, P(\Theta) \right)$$

↑ eg log. reg. model.

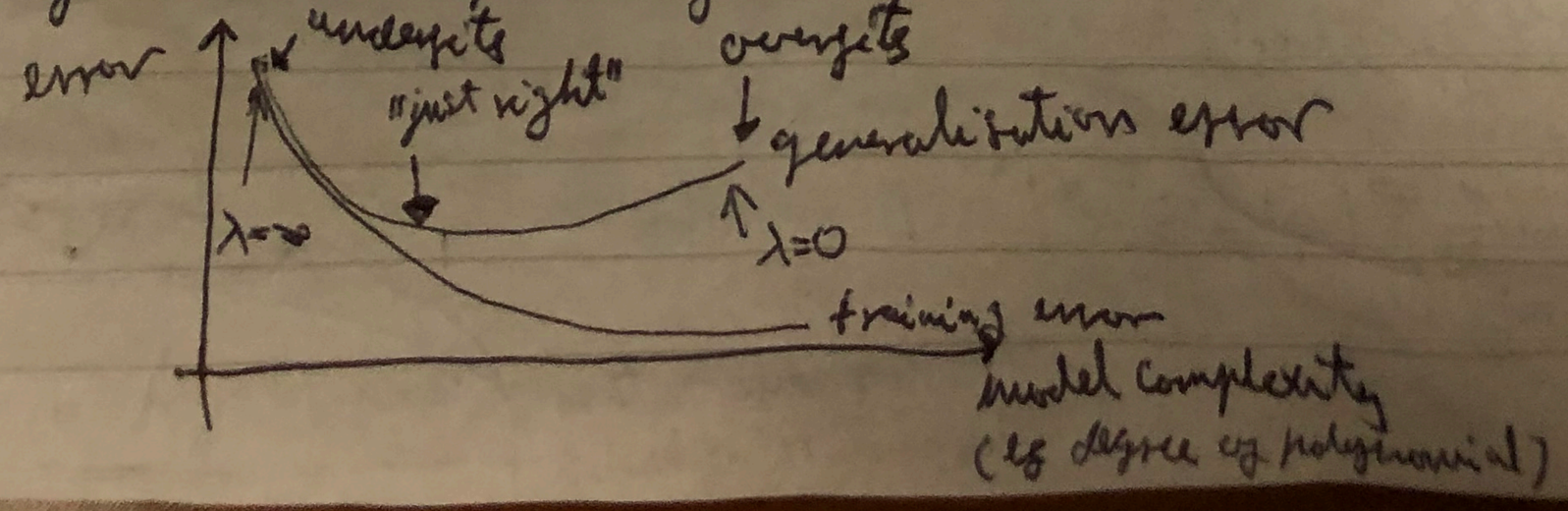
assume Gaussian

$$\rightarrow P_G(\Theta): \Theta \sim \mathcal{N}(0, \tau^2 \mathbb{I}) \Leftrightarrow P_G(\Theta) = \frac{1}{\sqrt{2\pi}\,[\tau^2 \mathbb{I}]^{1/2}} \exp\left(-\Theta^T (\tau^2 \mathbb{I})^{-1} \Theta\right)$$

As it turns out: Now, if you plug in $P_G(\Theta)$ for P in the above optimization problem and solve it, our regularization technique falls out.

Frequentist: $\arg\max_{\Theta} P(S|\Theta)$ – MLE : non-regularized per definition (but one can apply it after)

Bayesian: Prior distribution $P_G(\Theta) \rightarrow \arg\max_{\Theta} P(\Theta|S)$ – MAP estimator – regularized per definition.



Train/dev/test sets

10,000 examples: Eg $\Theta_0 + \Theta_1 x$ vs $\Theta_0 + \Theta_1 x + \Theta_2 x^2$, etc., or choose λ, or choose $\vec{c}$ or C

$S \to S_{train}, S_{dev}, S_{test}$

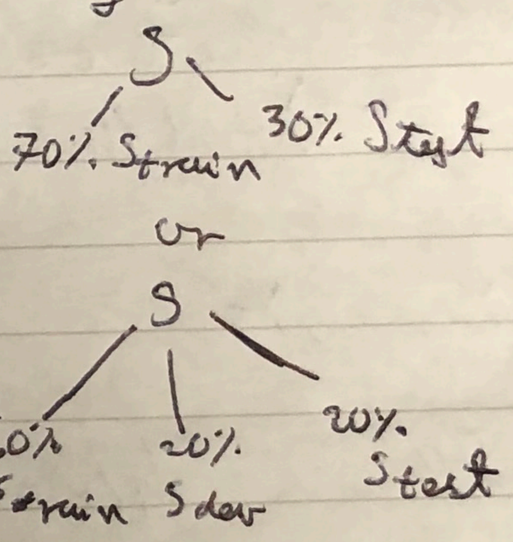
- Train each model (Option for degree of polynomial) on S_{train}. Get some hypothesis h_i.
- Measure error on S_{dev}. Pick model with lowest error on S_{dev}.
  => S_{dev} is there to select hyperparameters.
- Optional: Evaluate algorithm on test set S_{test} and report error.

Holdout cross validation (simple)

degree	S_{dev} error	S_{test}
1	10	10
2	5.1	3.0
3	5.0	5.0
4	4.9	5.0
5	7	7
6	10	10
⋮		⋮

Note: sometimes, people call S_{dev} = $S_{cross.val.}$

Rules of thumb for "reasonably sized datasets":



But for enormous datasets, need smaller % on S_{dev}, S_{test} (in general, just pick S_{test}/S_{dev} large enough to differentiate your various models reliably, but not much larger than that.

Note: never base any decisions on S_{test}, to remain unbiased.

Small datasets: For 70-30 split: do we really want to use "just" 70 training examples?

Not usually, hence K-fold CV:

S
$\left|\begin{matrix} x^1, y^1 \\ \vdots \\ x^{100}, y^{100} \end{matrix}\right|$

$k=5$ only for illustration, $k=10$ is the typical.
For $i=1,\dots,k$ train (fit parameters) on $k-1$ pieces and test on remaining 1 piece, and then average all these errors.

Optional Final step: Refit model on 100% of data.

Advantage: Uses all data. Disadvantage: Computational expensive

20-100:
$m=20$ extreme version of k-CV is "leave-out-one-CV":
$k=m$.

Feature Selection
Start with $\mathcal{F}=\emptyset$
Repeat:
1) Try adding each feature i to $\mathcal{F}$, and see which single feature addition most improves dev set performance.
2) Add that feature to $\mathcal{F}$

$x_1,\dots,x_5$

$\left[\begin{matrix} \emptyset & h(x)=\Theta_0 \\ \emptyset+x_1 & \\ \emptyset+x_2 & \Rightarrow \mathcal{F}=\{x_2\} \\ \vdots & \\ \emptyset+x_5 & h_\theta(x)=\Theta_0+\Theta_1 x_5 \end{matrix}\right.$

$h_\theta(x)=\Theta_0+\Theta_1 x_2$ new model. Turns out Θ_2 adds best performance $\Rightarrow$ new model is $\Theta_0+\Theta_2^T\begin{bmatrix} x_2 \\ x_2^2 \end{bmatrix}$.

This is a reasonable, <u>greedy algorithm</u>.