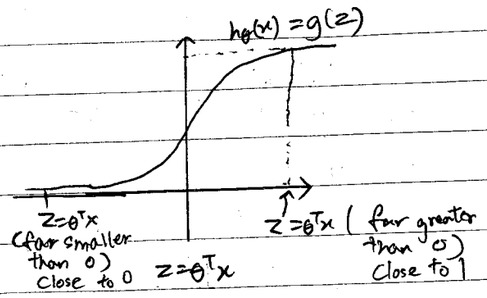
No. Date

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 $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

Alternative view of logistic regression

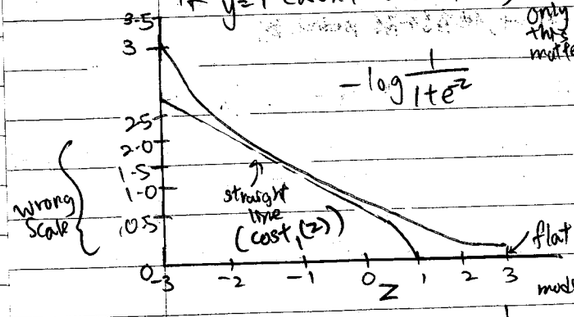
Cost of example: $-y \log h_\theta(x) - (1-y)\log(1-h_\theta(x))$

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

(single (x,y))

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

$$-\log \frac{1}{1+e^{-z}}$$

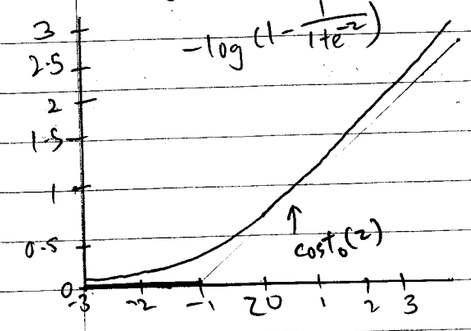


modify the cost function

when $z \uparrow$, $-\log \frac{1}{1+e^{-z}} \downarrow$

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

$$-\log\left(1 - \frac{1}{1+e^{-z}}\right)$$



No. Date

Support vector machine

Logistic regression:

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

(未来在前面的，指数在后面)

A, B

Support vector machine:

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

$\Rightarrow$ cut out the constant m

$\min_u ((u-5)^2+1) \times 10 \longrightarrow u=5$ (get smallest)

$\min_u 10(u-5)^2+10 \longrightarrow u=5$ (still u=5 to get smallest)

So multiply something that you're minimizing over by some constant, 10 in this case, it does not change the value of u that gives us, that minimizes this function.

LR: $A + \lambda B$ — in logistic regression, λ can act as a control to trade off the relative weight btwn how much we wanted the training set fit well.

$\lambda \uparrow$, $B \uparrow$

SVM: $CA + B$ — C playing a role as $\frac{1}{\lambda}$

$C \downarrow$, $B > A, C$

overall optimization objective function for svm.

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

if minimize this function, what you have is the parameters learned by SVM

SVM hypothesis

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

Hypothesis

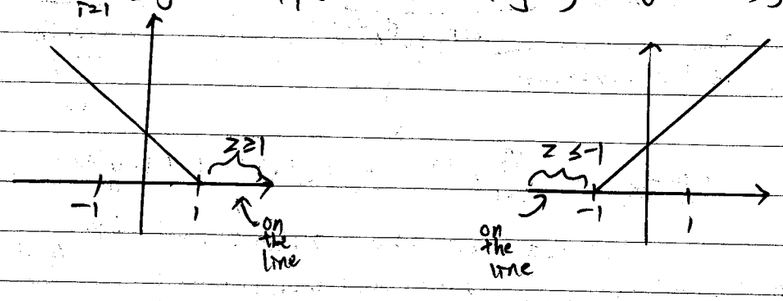
$$h_\theta(x) \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Support Vector Machines

Large Margin Intuition

SVM:

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



If $y=1$, we want $\theta^Tx\geq 1$ (not just ≥ 0)

If $y=0$, we want $\theta^Tx\leq -1$ (not just <0)

↗ This higher value to determine $y=1$ / $y=0$ builds in an extra safety factor or safety margin factor into SVM.

SVM Decision Boundary

$$\min_\theta C\underbrace{\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1(\theta^Tx^{(i)})+(1-y^{(i)})\text{cost}_0(\theta^Tx^{(i)})\right]}_{\text{we have to choose the value so this }=0}+\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

if C very very large like 100,000

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

$\theta^Tx^{(i)}\geq 1$

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

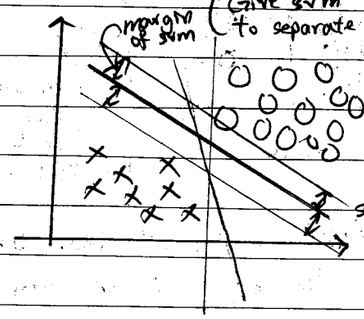
Subject to $\theta^Tx^{(i)}\geq 1$ if $y^{(i)}=1$

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

Whenever $y^{(i)}=0$:

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

SVM Decision Boundary: Linearly separable case

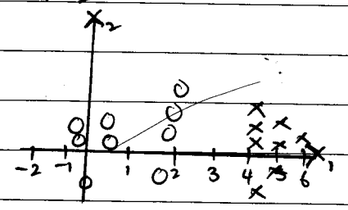


(Give svm a certain robustness as it tries to separate data with as large the margin as possible.)

svm will choose this DB which seems better than the green one, this DB has a larger distance which is known as margin, from any of the training examples, whereas green line is close to examples.

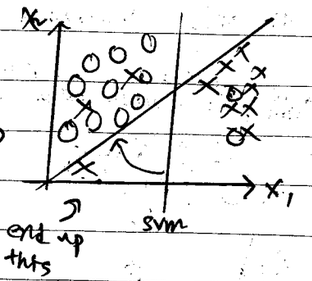
svm sometimes called large margin classifier

Quiz: Consider the training set to the right, where "x" denotes +ve examples ($y=1$) and "o" denotes -ve examples ($y=0$). Suppose you train an SVM (which will predict 1 when $\theta_0+\theta_1x_1+\theta_2x_2\geq 0$). What values might SVM give for θ_0, θ_1, and θ_2?



Answer: $\theta_0=-3$, $\theta_1=1$, $\theta_2=0$

Large margin classifier in presence of outliers



→ if C very large

svm will change like this

→ If C not too large, svm DB no change

end up this DB

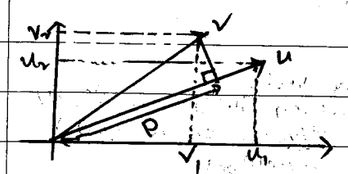
⇒ It's really not clear based on the single outliner, and it's not a good idea to change DB.

If got training examples like green "o" and "x" appear, svm will also do the right thing

Support vector Machines

The mathematics behind large margin classification

Vector Inner Product



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

$\|u\|$ = length of vector u | $u^T v$ (called as inner product b/twn vectors u and v)

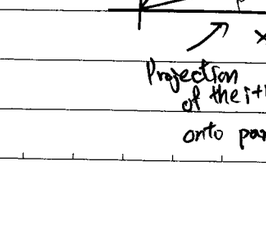
$= \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$

p = length/magnitude of the projection of vector v onto the vector u

Signed (could be +ve or -ve)

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

$= u_1 v_1 + u_2 v_2$ $p \in \mathbb{R}$



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

$p < 0$

* In inner product, if angle b/twn u and v is less than 90°, then p is +ve

if angle is greater than 90°, p is -ve.

SVM Decision Boundary

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

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

$\hookrightarrow \|\theta\|$

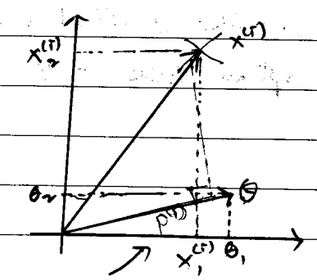
subject to $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$

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

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

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

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



Projection of the i-th training example onto parameter vector theta

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

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

⇒ Continue at the back.

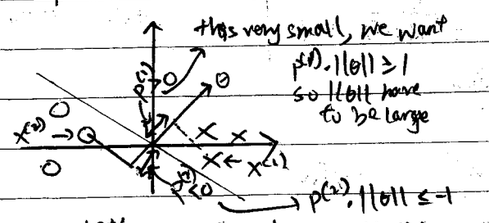
SVM Decision Boundary

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

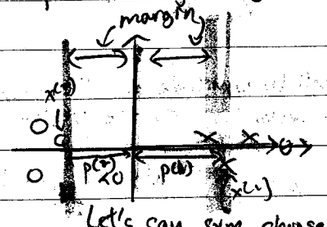
s.t. $p^{(i)} \cdot \|\theta\| \geq 1$ if $y^{(i)}=1$
$p^{(i)} \cdot \|\theta\| \leq -1$ if $y^{(i)}=0$
} ← very large

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.
(if $\theta_0 \neq 0$, DB can be not passing through origin(0,0))
Simplification : $\theta_0 = 0$ (means DB must pass through origin (0,0))

Note: vector theta be 90° to DB



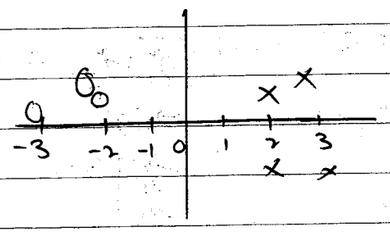
Let's say svm choose green DB
→ not a good choice, margin too small
∴ But svm will not do this
⇒ In this, $p^{(i)}$ will be pretty small numbers. And the $\|\theta\|$ must be big



Let's say svm choose this pink DB
⇒ $p^{(1)}$ and $p^{(2)}$ here much more bigger
$p^{(1)} \cdot \|\theta\| \geq 1$
$\|\theta\|$ can be smaller

(Svm will choose this)
Margin is the value of $p^{(1)}$, $p^{(1)}$ bigger, margin bigger and $\|\theta\|$ can be smaller which is what we trying to do

Quiz:



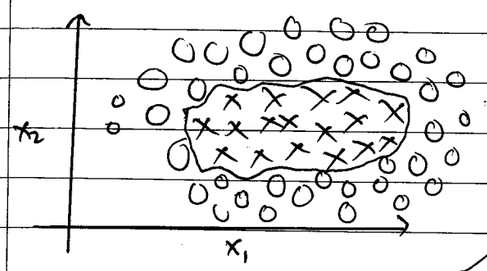
where $p^{(i)}$ is the (signed- +ve/-ve) projection of $x^{(i)}$ onto θ. Consider the training set above. At the optimal value of θ. What is $\|\theta\|$?

Answer : $\frac{1}{2}$

Support Vector Machines

Kernels I

Non-linear Decision Boundary



Predict y=1 if
$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \ldots \geq 0$

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

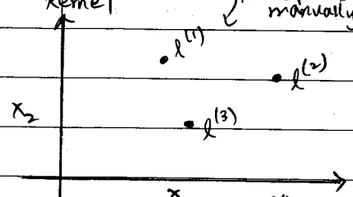
$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \ldots$

$f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$, $f_4 = x_1^2$, $f_5 = x_2^2$, ...

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

Idea on how to define new features f_1, f_2, f_3

Example:



Put 3 points manually and name and choose them manually

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

Given x: example $x^{(i)}$, 1st feature, measure of similarity between training example x and first landmark

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

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

$f_3 = \text{similarity}(x, l^{(3)}) = \exp\left(-\frac{\|x-l^{(3)}\|^2}{2\sigma^2}\right)$

Kernel function (Gaussian Kernels) ← specific example here is named this

⇒ $k(x, l^{(i)})$

No. _____ Date _____

Kernels and Similarity

$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$ ← ignoring x_0

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

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

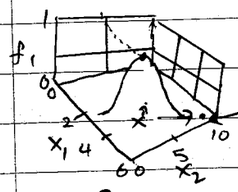
$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0$

Example: 2 features → x_1, x_2

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

σ^2: parameter of Gaussian Kernel

$\sigma^2 = 1$

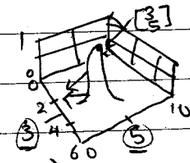


when $x = \begin{bmatrix}3\\5\end{bmatrix}$, $f_1 = 1$

≈ 0 if x more away

nice bump (凸起来的)

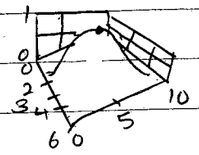
$\sigma^2 = 0.5$



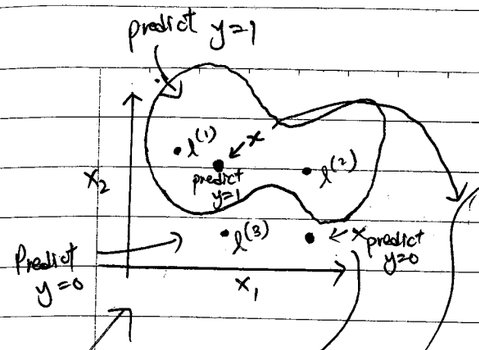
f_1 falls to zero much more rapidly

the bump become narrower

$\sigma^2 = 3$



as you move away from $l^{(1)}$, value of feature falls away much more slowly



$f_1, f_2, f_3 \approx 0$

$\theta_0 + \theta_1 x_0 + \theta_2 x_0 + \theta_3 x_0$

$= \theta_0$

$\approx -0.5 < 0$

Because is < 0, for this training example x, predict $y=0$

Predict "1" when $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Put x as

Let's say

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

Because x is near to $l^{(1)}$, but far from $l^{(2)}$ and $l^{(3)}$,

So, $f_1 \approx 1$, $f_2 \approx 0$, $f_3 \approx 0$ (not exactly 1 or 0, just close to)

$\theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$

$= -0.5 + 1$

$= 0.5 \geq 0$

Because is ≥ 0, for training example x, predict $y=1$

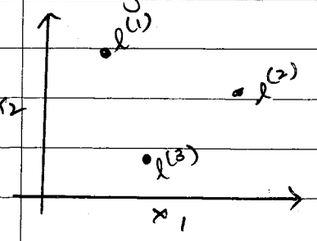
We end up predicting those example near to $l^{(1)}$, $l^{(2)}$ as $y=1$ and far from $l^{(1)}$, $l^{(2)}$ as $y=0$. So we got a DB like this.

Support Vector Machines

Date

Kernels II

Choosing the landmarks



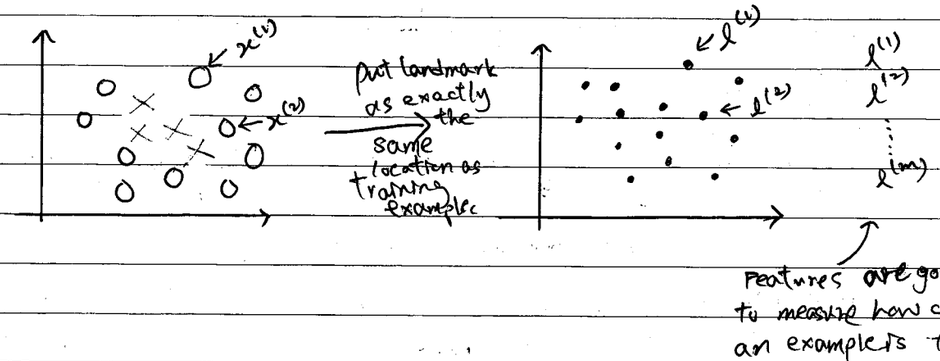
Given x:

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

$= \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$ ← 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 $l^{(1)}, l^{(2)}, l^{(3)}, \ldots$?



features are going to measure how close an example is to one of the things in training set.

SVM with kernels

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

Choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$

~~can~~ be in the training set / CV set / test set

Given example x:

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

$f_2 = \text{similarity}(x, l^{(2)}) \Rightarrow f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$ $f_0 = 1$ (x_0, interceptor) (as a feature vector)

$\ldots$

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

$x^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_i^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$ $= \text{sim}(x^{(i)}, l^{(1)})$

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

$\leftarrow f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)})$ ($l^{(i)} = x^{(i)}$)

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

Gaussian Kernel: $= \exp\left(-\frac{0}{2\sigma^2}\right) = 1$

[$f_i^{(i)}$ is just be similarity btwn x and itself]

Sim = abbr of similarity

→ Can take these and group them into a feature vector

Instead use $x^{(i)} \in \mathbb{R}^{m+1}$ (or $\mathbb{R}^n$) to represent training example

can use $f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$ → new feature vector to represent training example

SVM with Kernels

Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$ (m training example/m landmarks)

Predict "$y=1$" if $\theta^T f \geq 0$ ($\theta \in \mathbb{R}^{m+1}$)

$\theta_0 f_0 + \theta_1 f_1 + \ldots + \theta_m f_m$

Training:

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

($\theta^T f^{(i)}$; Not regularizing θ_0)

$\sum_{j=1}^{n} \theta_j^2 = \theta^T \theta$ ← $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$ (ignoring θ_0)

replace SVM implementation with $\theta^T M \theta$ instead of minimizing exactly $\|\theta\|^2$

rescale version → The parameter vector theta depends on kernel

this matrix (depends on kernel you use)

this kind of mathematical detail allows SVM software to run more efficiently → this modification allow to scale to much bigger training sets

Can use kernel idea on logistic regression, but will very slow. Because of computational tricks like that in body and how it modifies and details of how SVM software is implemented, so SVM and kernels tend to go particularly well together.

- Use the software/packages that developed by others is better than writing them yourself.

SVM parameters: Bias and Variance Trade-off

$C\ (=\frac{1}{\lambda})$ Large C: Lower bias, high variance (small λ)

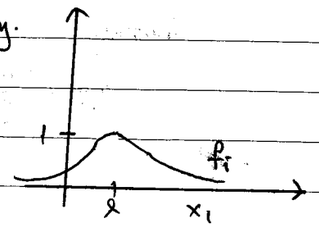
Small C: Higher bias, low variance (large λ)

In Gaussian kernels

σ^2 Large σ^2: Features f_i vary more smoothly.

Higher bias, lower variance.

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

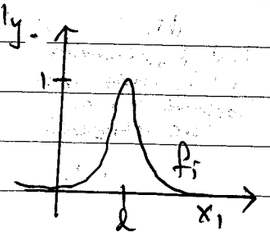


Gaussian kernel falls off smoothly, will get a hypothesis that varies smoothly as you change the input x

* σ determine the width of the Gaussian kernel

Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.



The features vary less smoothly as higher slopes / higher derivatives, end up fitting hypothesis of lower bias and higher variance

Quiz: Suppose you train an svm and find it overfits your training data. Which of these would be a reasonable next step?

- Decrease C
- Increase σ^2

Support Vector Machines

Using an SVM

Use svm software package (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")

Predict "y=1" if $\theta^T x \geq 0$ ($\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n \geq 0$)

why want to do this?

n large, m small

↳ maybe want to just fit a linear DB and not try to fit a very complicated nonlinear function, because might not have enough data.

Gaussian kernel:

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

Need to choose σ^2

When to use?

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

and want to use kernel to fit more complex non-linear DB

kernel (similarity) functions: $x^{(i)} \xrightarrow{f_i} l^{(i)} = x^{(i)}$

Code → function f = kernel(x_1, x_2)

$$f = \exp\left(-\frac{\|x_1-x_2\|^2}{2\sigma^2}\right)$$

return

Note: Do perform feature scaling before using Gaussian kernel.

$\|x-l\|^2$ → $v = x-l$ $\quad x \in \mathbb{R}^n$

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

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

Use housing problem: if this is 1000 ft² and this is #bedrooms (1-5) ⇐ the difference will very large, so need feature scaling

No. _____

Date _____

Other choices of kernel

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

kernel other than the 2 that mentioned previously, maybe will use this

Many off-the-shelf kernels available:

- Polynomial kernel: $k(x, l) = (x^T l)^2$ (+0 one of the version). ← one measure how similar x and l are. If x and l very close to each other, the inner product tend to be large

Another version: $(x^T l)^3$, $(x^T l + 1)^2$, $(x^T l + 5)^4$ ← degree of polynomial

⇒ has 2 parameters (the constant, and the degree); number you add here

In general: $(x^T l + \text{constant})^{\text{degree}}$

But almost always perform worse than Gaussian kernel

Usually used only for data where x and l are all strictly non negative, so that ensure these inner products are never -ve

- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, ...

to measure similarity between different object

use if your input data is text strings or other types of strings.

The more similar two strings/two objects are, the higher the value of K(a, b) will be.

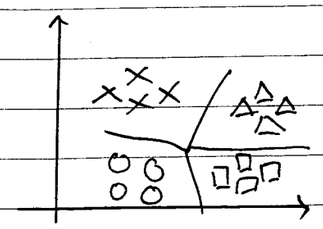
Quiz:

Suppose you are trying to decide among a few different choices of kernel and are also choosing parameters such as C, σ^2, etc. How should you make the choice?

Answer: choose whatever performs best on cross-validation data.

Date _____

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 SVMs, one to distinguish y=i from the rest, for i = 1, 2, ..., K), get $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$

Pick class i with largest $(\theta^{(i)})^T x$ (i: y=1, y=2)

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): (eg. n ≥ m, n = 10,000, m = 10 ... 1000) (text classification problem)

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

⇒ if n is small, m is intermediate: (n = 1 — 1000, m = 10 — 10,000)

Use SVM with Gaussian kernel (Create/Add more polynomial feature)

⇒ 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 usually do pretty similar things and give pretty similar performance, but depending on the implementational details, one may be more efficient than the other.

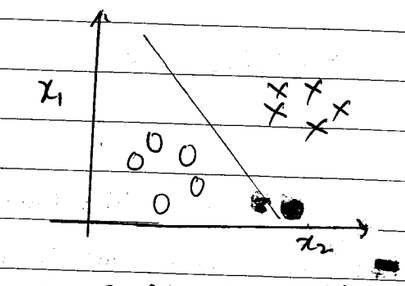
⇒ Neural Network likely to work well for most of these settings, but may be slower to train.

SVM has a convex optimization problem, and good SVM optimization software packages will always find the global min. or something close to it. So for SVM, no need to worry about local optima.

Clustering

Unsupervised learning introduction (learned from unlabeled data)

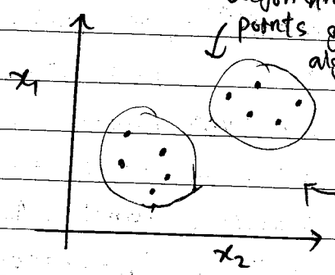
Supervised learning



Training set : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots, (x^{(m)}, y^{(m)})\}$

Goal : To find DB that separates +ve label examples and -ve label examples

Unsupervised learning



algorithm find, looks like this data set has points grouped into 2 separate clusters, algorithm that find clusters like the circled one is called a clustering algorithm.

← Data that has no labels

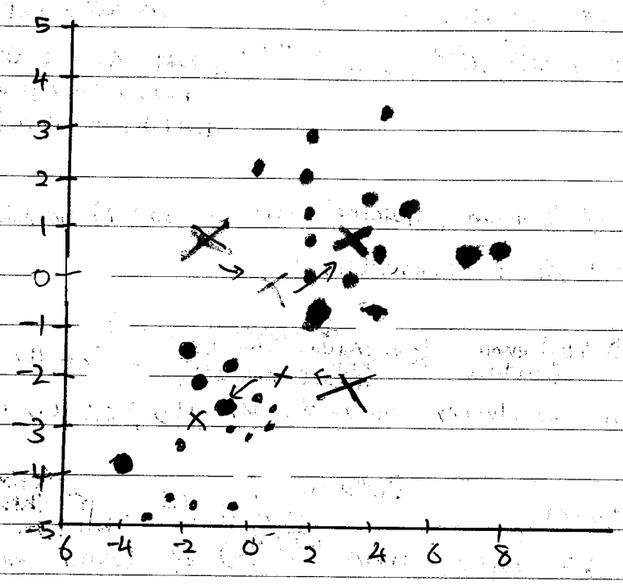
Training set : $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)}\}$ ← don't get label y

Give this sort of unlabeled training set to an algorithm and ask algorithm find some structure in the data for us.

Application of clustering
- Market segmentation
- Social network analysis
- Organize computing clusters
- Astronomical data analysis

Clustering

K-means algorithm ⇒ to automatically group unlabeled data into coherent subsets / coherent clusters for us.



← First, randomly initialize two points, called cluster centroids.

Got 2 cluster centroids, because going to group data into 2 clusters.

K-Mean ⇒ iterative algorithm does 2 things — cluster assignment step
— move centroid step

- Going through each of the example (dots ·), depend which cluster centroid is closer to, black or blue, assign each of the data points to one of it.

(first thing) – Those blue belong to blue cluster centroid, while black belong to black cluster centroid

(second thing) – Look at the mean of blue dot, move the blue cluster centroid there, and look at the mean of black dot, move the black cluster centroid there. (arrow indicate moving)

(first thing again) – After moving the cluster centroid, do the cluster assignment step again, which colour them either blue/black, so the colour of some dots change again

(first thing again) – Move to the mean of new trainy data again, and recolour those data which are closer to either blue/black cluster centroid again.

- Keep running additional iterations of K-mean until a final mean, it will not change any further, and had done finding 2 clusters in data.

Date

K-mean algorithm

Input:

- K (number of clusters) ← we want certain number of cluster, our job tell how many we want in data set.
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ ← just xs, as this is unsupervised learning, we don't have labels y.

For unsupervised learning, of K-mean, training examples are n-dimensional rather than n+1 $x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

What it does — K ⇒ total num. of centroids; k ⇒ index (num. of between 1 and K) into different centroid ⇒ location of the cross on last page

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots \mu_K \in \mathbb{R}^n$

Repeat {

Cluster assignment step (that colour the points):

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$ — Distance → $\min_k \|x^{(i)} - \mu_k\|^2$ [value of k that minimize this is what to set in $c^{(i)}$]

Move centroid step:

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

let's say $x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$ assigned to cluster 2, $c^{(1)}=2, c^{(5)}=2, c^{(6)}=2, c^{(10)}=2$

moving μ_2 ⇒ (location) $\mu_2 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}\right] \in \mathbb{R}^n$ ← because all $x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$ are n-dimensional vector (location of cluster centroid k)

}

Quiz: Suppose you run K-means and after the algorithm converges, you have $c^{(1)} = 3, c^{(2)} = 3, c^{(3)} = 5 \dots$

True statement: — Third example $x^{(3)}$ has been assigned to cluster 5

- First and second training examples $x^{(1)}$ and $x^{(2)}$ have been assigned to the same cluster.
- Out of all the possible values of $k \in \{1, 2, \dots, K\}$ the value $k=3$ minimizes $\|x^{(3)} - \mu_k\|^2$.

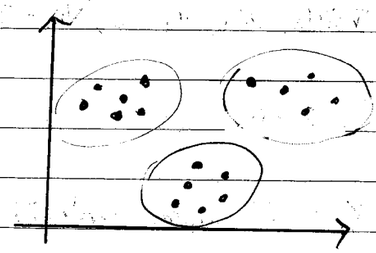
Note:

- Think of $c^{(i)}$ is picking the cluster centroid with the smallest squared distance to training example $x^{(i)}$
- Cluster centroid with no points / zero points assigned to it, can just eliminate this cluster centroid.

(Sometimes if you really need K cluster, can randomly reinitialize the cluster that don't have points, but more common to just eliminate it)

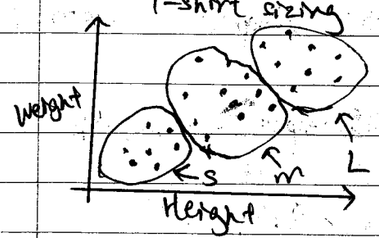
Date

K-means for non-separated clusters



⇐ 3 well separated clusters, we can just let algorithm to find 3 clusters for us.

very often k-mean also apply to data set like this



For example, want to sell T-shirt in 3 sizes ⇒ S, M, L

K-Mean will group these data into 3 clusters ⇐ kind of market segmentation

- Tries to fit the needs of each of the 3 separate sub-populations well.

Clustering — Optimization objective (knowing this help us in debugging learning algorithm, make sure it is running correctly and find better costs and avoid local optima)

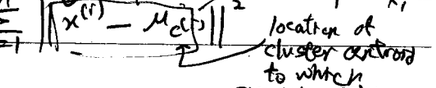
K-means optimization objective

$c^{(i)}$ = index of cluster (1, 2, …, K) to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

(Let's say $x^{(i)}$ assigned to 5 (cluster), $c^{(i)} = 5$, $\mu_{c^{(i)}} = \mu_5$)



Optimization objective:

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Try to find c and μ to try to minimize cost function J → $\min_{c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

Distortion cost function / Distortion of the K-means algorithm.

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

  for i = 1 to m (cluster assignment step)

    $c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

  for k = 1 to K

    μ_k := average (mean) of points assigned to cluster k

}

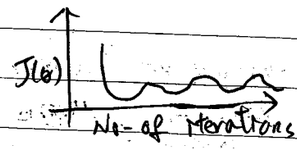
move centroid step — minimize $J(\dots)$ with respect to $\mu_1, \dots, \mu_K$

[minimize $J(\dots)$ with respect to $c^{(1)}, c^{(2)}, \dots, c^{(m)}$] while (holding $\mu_1, \dots, \mu_K$ fixed)

Note: First, minimize $J(\dots)$ with respect to $c^{(1)}, c^{(2)}, \dots c^{(m)}$

Then, minimize $J(\dots)$ with respect to $\mu_1, \dots, \mu_K$

Quiz: Suppose you have implemented k-means and to check that it is running correctly, you plot the cost function $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$ as a function of the number of iterations. Your plot looks like this:



What does this mean?

⇐ It is not possible for the cost function to sometimes increase. There must be a bug in the code.

Clustering

Random initialization (initialize k-mean)

First step in K-means algorithm

↳ Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Random initialization

Should have $K < m$ ($K = 2$)

Randomly pick K training examples.

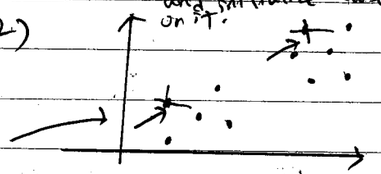
Set $\mu_1, \dots, \mu_K$ equal to these K examples.

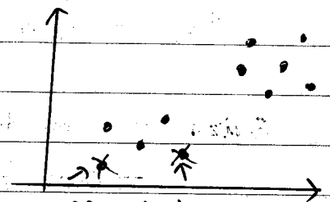
$\mu_1 = x^{(i)}$ randomly choose 2 value i and j

$\mu_2 = x^{(j)}$

⋮

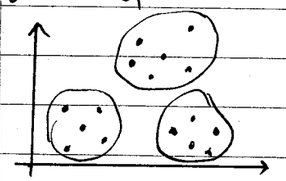
and go on as you have more clusters centroids.



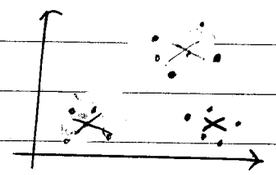


if unlucky, might pick 2 near together one

Local optima



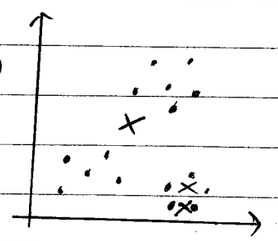
Run K-mean

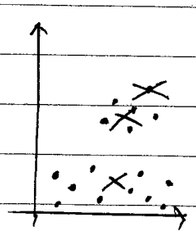


ends up at a good local optima, might be really global optima, might end up with that cluster ring.

nicely

If unlucky random initialization, K-mean also will get stuck at different local optimas.





Term local optima refers to local optima of distortion function J

$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

↗ Solutions where K-means got stuck at local optima and not doing good job minimizing distortion function J.

Note: Want to increase the odd of K-means finding the best possible clustering like on top, we can try multiple, random initialization, to keep running K-mean T to try, so we can get the good local/global optima as possible

Random initialization — may range from 50–1000

For i=1 to 100 {

Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.

Compute cost function (distortion)

$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

}

After running 100 times, will have 100 different ways of clustering data

- Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

Quiz: Recommended way to initialize k-means.

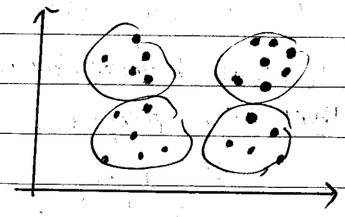
- Pick k distinct random integers $i_1, \dots, i_k$ from $\{1, \dots, m\}$

Set $\mu_1 = x^{(i_1)}, \mu_2 = x^{(i_2)}, \dots, \mu_k = x^{(i_k)}$

Clustering

Choosing the number of clusters

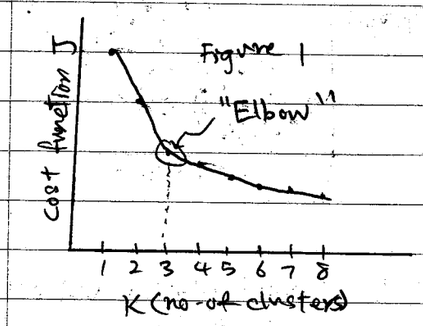
What is the right value of K?



Some people might say 4, and some might say 2, might say 3 as well. so it is quite ambiguous.

Choosing number of cluster

=) Elbow method.



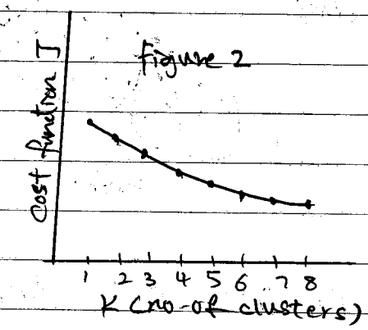


Figure 1:
- vary the value of K and run K-mean to see the value of cost function J
- go down rapidly from 1 → 2 → 3 and goes very slowly after that
- So maybe using 3 clusters is the right number of clusters, because it is the elbow of the curve.

Figure 2:
- Fairly often will end up with curve that much more ambiguous
- Might no clear elbow, looks like distortion continuously goes down.
- Harder to choose number of cluster

Note: Not really good method to choose the number of cluster

Quiz: Suppose you run k-means using K=3 and K=5. You find that cost function J is much higher for K=5 than K=3. What can you conclude?

- In the run with K=5, k-means got stuck in a bad local minimum. You should try re-running k-means with multiple random initializations.

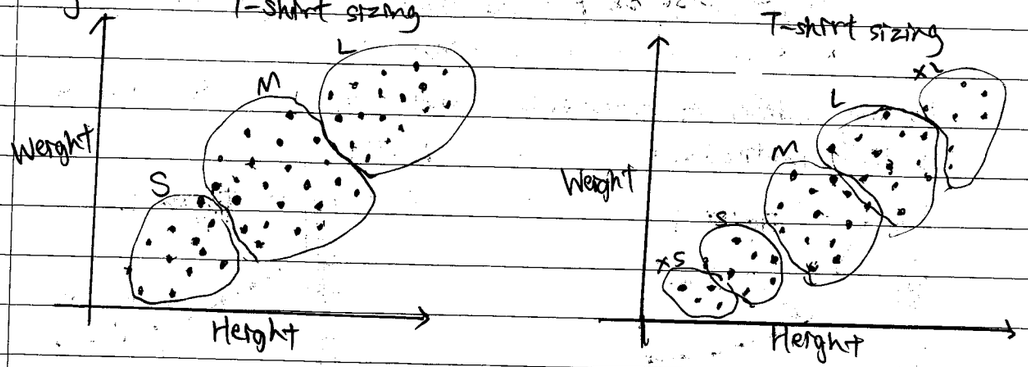
Choosing the value of K

Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate k-means based on a metric for how well it performs for that later purpose.

If choose $k=3 \Rightarrow$ S, M, L

If choose $k=5 \Rightarrow$ XS, S, M, L, XL (redundant)

E.g.



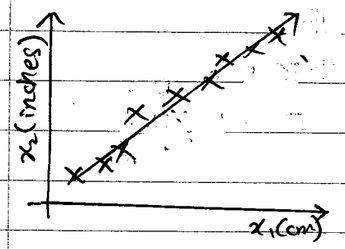
Can think of this from perspective of T-shirt business and ask
- If has 5 segments, how well will T-shirt fit the customers
- How many T-shirts can I sell?
- How happy will my customers be?
- Want have more/fewer T-shirt size to fit customers better?
- Or make fewer size of T-shirt to sell customers cheaply.

Better way to choose value of K is to ask yourself, what purpose are you running K-means? And think, what is the number of cluster K that serve that what is the later purpose that you actually run the K-means for.

Dimensionality Reduction

Motivation 1: Data Compression

Data Compression



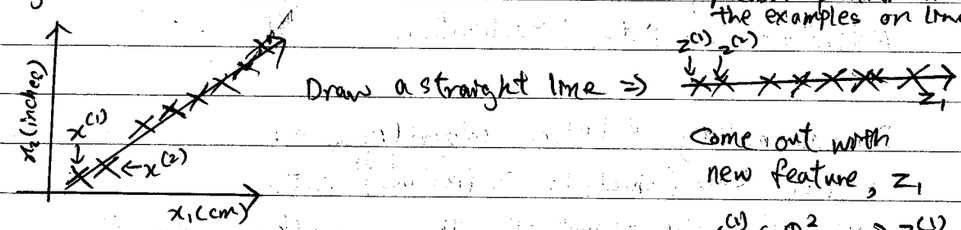
Reduce data from 2D to 1D (Reduce redundancy)

x_1 can also be pilot skill
x_2 can also be pilot enjoyment
} 2 features are highly correlated

=) What you need to care about is that this different feature really measures pilot aptitude

If the features are really highly correlated, need to reduce dimension.

Change to different colour



Draw a straight line =>

Come out with new feature, z_1

$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$

$x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$

$\vdots$

$x^{(m)} \rightarrow z^{(m)}$

=) This halve the memory/space to store data. Data compression can also allow us to run learning algorithm more quickly. This more interesting in Data compression.

No. _____

Date _____

Data Compression

1000D → 100D

Reduce Data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3$

$z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ $z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$

Project the 3D training set onto 2D, and use z_1 and z_2 as the axis.

Quiz: Suppose we apply dimensionality reduction to a dataset of m examples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$.
As a result of this, we will get out:
Answer: A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$ of m examples where $z^{(i)} \in \mathbb{R}^k$ for some value of k and $k \leq n$.

Dimensionality Reduction

Motivation II: Data Visualization

Dimensionality Reduction algorithms can reduce 50D (which is 50 features) $\mathbb{R}^{50}$ into 2D ($\mathbb{R}^2$). The output of this algorithm, doesn't ascribe a particular meaning to new features (z_1 and z_2). But it will up to us to figure out what does this features mean
⇒ will find that, in plotting graph, axes z_1 and z_2 can help us to most succinctly capture really what are the 2 main dimensions of the variations amongst different (example) countries/people/other things that being measured and plot.

Quiz: Suppose you have a dataset $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ where $x^{(i)} \in \mathbb{R}^n$. In order to visualize it, we apply dimensional reduction and get $\{z^{(1)}, z^{(2)}, \dots, z^{(m)}\}$ where $z^{(i)} \in \mathbb{R}^k$ is k-dimensional. In typical setting, which (would you expect) is true?
- $k \leq n$
- $k = 2$ or $k = 3$ (since we can plot 2D or 3D data but don't have ways to visualize higher dimensional data)

No. _____

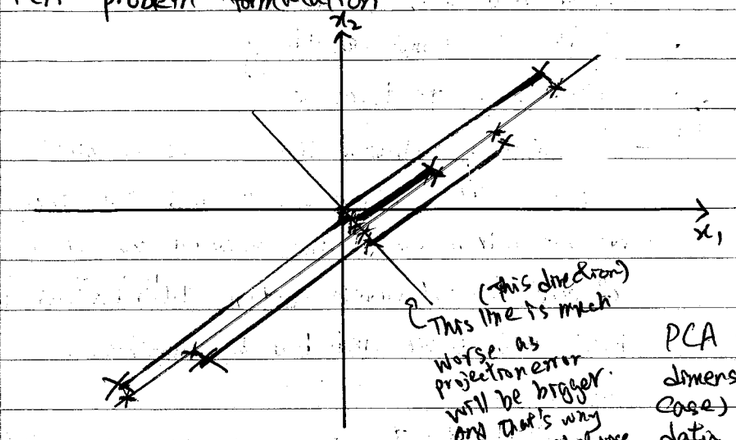
Date _____

Dimensionality Reduction

Principal Component Analysis problem formulation

PCA problem formulation

$x \in \mathbb{R}^2$

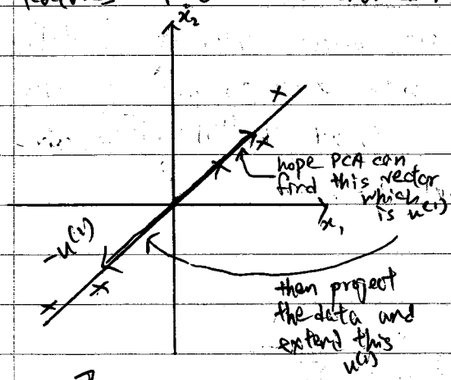


← Want reduce from 2D to 1D (to find a line onto which to project the data)

PCA tries to find a lower dimensional surface (a line in this case) onto which to project the data so that the sum of squares of these little yellow line segments is minimized

Note: length of yellow line segment Sometimes called projection error.

Before applying PCA, perform mean normalization at feature scaling so that features x_1 and x_2 should have zero mean and comparable ranges of value

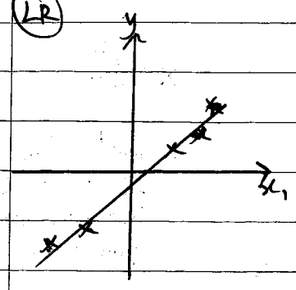


Reduce from 2-D to 1-D: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$ in this case is $\mathbb{R}^2$) onto which to project the data so as to minimize the projection error

In general: Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

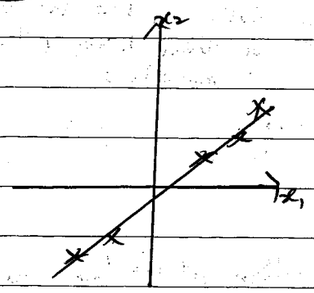
PCA is not linear regression (totally different algorithms)

LR



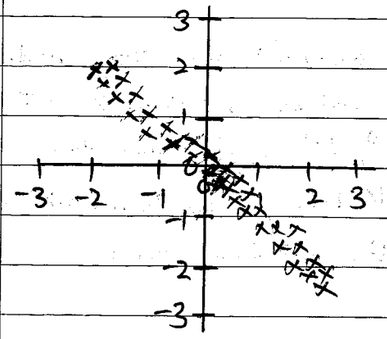
- linear regression is to predict value of some variable y given info features x
- we fit a straight line, so to minimize square error btwn point and the straight line.
- We are minimizing the squared magnitude of purple lines (which are vertical distance between point and value predicted by hypothesis)

PCA



- PCA tries to minimize the distance between points and the line, no matter in which angle, just want the shortest orthogonal distances.
  ↑ not necessary vertical

Suppose you run PCA on the dataset below. Which of the following would be a reasonable vector $u^{(1)}$ onto which to project the data? (By convention, we choose $u^{(1)}$ so that $\|u^{(1)}\| = \sqrt{(u_1^{(1)})^2 + (u_2^{(1)})^2}$, the length of the vector $u^{(1)}$, equals 1.)



$$u^{(1)} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Svd
↳ Singular value decomposition

Dimensionality Reduction

Principal Component Analysis algorithm

Before applying PCA, have a step to do first. ↘

Data Preprocessing

Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ ← unlabeled

Preprocessing (feature scaling/mean normalization):

Compute mean of each feature: $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$

... Replace each $x_j^{(i)}$ with $x_j - \mu_j$

make each features has exactly zero mean ⇒

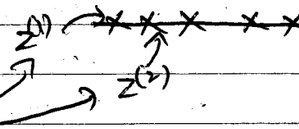
Scale: If different features on different scales (e.g. x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable range of values.

$$\frac{x_j^{(i)} - \mu_j}{s_j}$$

s_j ⇐ Measure of range of value of features j / max − min value / σ of j ← standard deviation

Reduce data from 2D to 1D

$x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$



Initially they were $x^{(1)}, x^{(2)}$ in 2D

Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

Procedure in Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix": ← This is summation symbol

Just a alphabet → $\Sigma = \frac{1}{m}\sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T$ → $n \times n$ matrix

($n \times 1$)($1 \times n$)

Compute "eigenvectors" of matrix Σ:

$[U, S, V] = \text{svd}(\text{Sigma});$ ← If not using octave/MATLAB to compute this, can download a numerical linear algebra library that can compute SVD

svd outputs 3 matrices

* Thing that svd really need is U matrix

$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(n)} \\ | & | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$

Take first k vectors → k

$u^{(1)}, \dots, u^{(k)}$ ← k direction onto which we want to project data

Principal Component Analysis (PCA) algorithm

From [U, S, V] = svd(Sigma); we get:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

($u^{(1)}, u^{(2)}, \ldots, u^{(k)}$: k)

Now to make $x \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^k$

$$\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}$$

($n \times k$)

Matrix U_{reduce} (as it is reduced version of U matrix)

Use this to reduce dimension of data

$$z^{(i)} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T \cdot x^{(i)} = \begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix} \cdot x^{(i)}$$

($k \times n$) · ($n \times 1$) → ($k \times 1$)

The x can be examples in training set / cv set / test set

$z \in \mathbb{R}^k$

PCA algorithm summary

- After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

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

```
[U, S, V] = svd(Sigma);
Ureduce = U(:, 1:k);
z = Ureduce' * x;
```

Vectorized implementation

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}$$

```
Sigma = (1/m) * X' * X;
```

You apply PCA with $x \in \mathbb{R}$

so ~~$x_0 = 1$~~

Quiz: In PCA, we obtain $z \in \mathbb{R}^k$ from $x \in \mathbb{R}^n$ as follows:

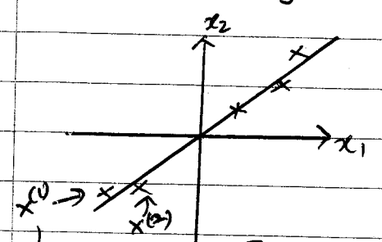
$$z = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x = \begin{bmatrix} — (u^{(1)})^T — \\ — (u^{(2)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix} x$$

$\therefore z_j = (u^{(j)})^T x$

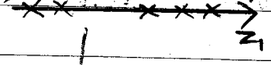
Dimensionality Reduction — Reconstruction from compressed representation

PCA is kind of compression algorithm

- So how do we get back to the form that are not compressed?



$z^{(1)}$ $z = U_{reduce}^T x$ ... $z \in \mathbb{R} \longrightarrow x \in \mathbb{R}^2$

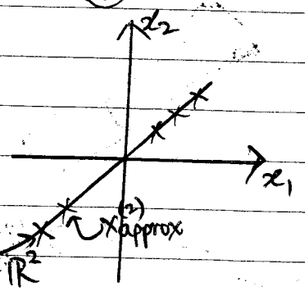


If squared projection error is not too big $\approx x$ (will close to original value of x that have used to derive z in the first place)

Equation: $x_{approx} = U_{reduce} * z$

($\mathbb{R}^n$ ← $n \times k$ · $k \times 1$ → $n \times 1$)

Back to this



$x_{approx}^{(1)} \in \mathbb{R}^2$

Quiz:

Suppose we run PCA with k=n, so that dimension of data is not reduced at all. (This is not useful in practice but is good thought exercise.) Recall that the percent/fraction of variance retained is given by: $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$ which ~~are~~ are true?

- U_{reduce} will be an $n \times n$ matrix
- $x_{approx} = x$ for every example x.
- The percentage of variance retained will be 100%

Dimensionality Reduction

Date

Choosing the number of principal components

PCA - we take n-dimensional features and reduce to k-dimensional feature representation.

number k is the parameter of PCA algorithm.
↳ also called number of principal components

— So, how people choose parameter k for PCA?

Choosing k (number of principal components)

PCA tries to minimize ⇒ Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$ ⇒ on average, how far are the examples from vector or just zero (origin)

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01 \quad (1\%)$$

(0.05) (5%)
(0.10) (10%)

Choose k so that, "99% of variance is retained"
(95%)
(90%)

Common range of value is 95–99%

Choosing k (number of principal components)

Algorithm:
Try PCA with $k=1$
Compute $U_{reduce}, z^{(1)}, z^{(2)}, \dots, z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$
Check if

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01 ?$$

If it is good, then use $k=1$,
If not good then next try $k=2$ and run through procedure and check again, maybe keep trying until a satisfied k value which 99% of data is retained

$[U, S, V] = \text{svd}(\text{Sigma})$

$S = \begin{bmatrix} S_{11} & & & \\ & S_{22} & & \\ & & S_{33} & \\ & & & \ddots \\ & & & & S_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n}$ (everything except diagonal is zero)

For given k:

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$$

Let's say $k=3$ → Sum of S_{11}, S_{22} and S_{33}

→ Sum of S_{11} to S_{nn}

1 minus this ratio can give you quantity of this

OR $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$

Slowly increase k, and see which k can ensures 99% of variance is retained.

smallest

Quiz:
PCA chooses a direction $u^{(1)}$ (or k direction $u^{(1)}, \dots, u^{(k)}$) onto which to project the data so as to minimize the (squared) projection error.
Another way is to say the same is the PCA tries to minimize:
$\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2$

Dimensionality Reduction — Advice for applying PCA

Most common use of PCA →

Supervised learning speedup
$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$
Extract inputs: (Put aside the y's)
Unlabeled dataset: $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^{10,000}$

Example: Doing computer vision
$x^{(i)} \in \mathbb{R}^{10,000}$ ← 100 images
100 images
will be slow to run learning algorithm

↓ PCA

$z^{(1)}, z^{(2)}, \dots, z^{(m)} \in \mathbb{R}^{1000}$

New training set:
$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$ ⇒ and feed to learning algorithm $h_\theta(z) = \frac{1}{1+e^{-\theta^T z}}$

If Get a new example x, take it and map it to the same mapping that found by PCA to get z. and fed to hypothesis

U_{reduce}, after computing all features on training set, just apply same mapping to cv and test set.

Note:
Mapping $x^{(i)} \to z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

Application of PCA

- Compression
  - Reduce memory / disk needed to store data
  - Speed up learning algorithm
  - Choose k by % of variance retain
- Visualization
  - plot out the data that is compressed
  - $k=2$ or $k=3$ as we can plot only 2D and 3D data sets.

Bad use of PCA: To prevent overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k<n$

Thus, fewer features, less likely to overfit.

↳ And this will work BAD!

Cause PCA only use x and don't use y, it will reduce and throw away some information without knowing value of y.

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\min_\theta \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

PCA is sometimes used where it shouldn't be

Design of ML system:

- Get training set $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\dots,(x^{(m)},y^{(m)})\}$
- ~~Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$~~
- Train logistic regression on $\{(z^{(1)},y^{(1)}),\dots,(z^{(m)},y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)},y_{test}^{(1)}),\dots,(z_{test}^{(m)},y_{test}^{(m)})\}$

Before run this, ask yourself → How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.

↑ Only if $x^{(i)}$ don't work, just consider using compressed representation ($z^{(i)}$).

Note: PCA

- Use to speed up running time of algorithm
- Compress data
- to reduce memory/disk space requirements
- to use it to visualize data

Anomaly Detection

Problem motivation (Density Estimation)

Anomaly detection example

Aircraft engine features:

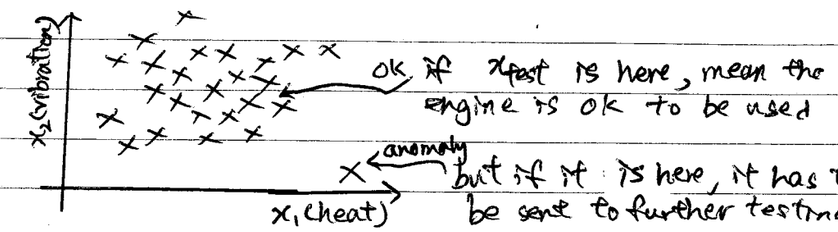
x_1 = heat generated

x_2 = vibration intensity

...

Dataset: $\{x^{(1)},x^{(2)},\dots,x^{(m)}\}$

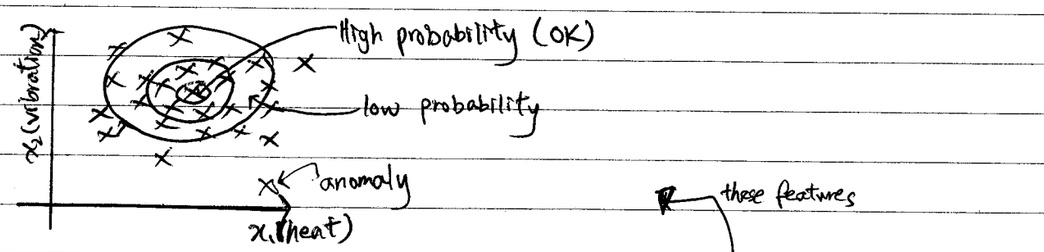
New engine: x_{test}



Density estimation

Dataset: $\{x^{(1)},x^{(2)},\dots,x^{(m)}\}$

Is x_{test} anomalous?



Given this training set (unlabeled), we build a model for $p(x)$

$p(x_{test}) < \varepsilon \rightarrow$ flag anomaly

$p(x_{test}) \geq \varepsilon \rightarrow$ OK

Anomaly detection example

(1) Fraud detection:

$x^{(i)}$ = features of user i's activities → maybe x_1 = how often does user log in, x_2 = number of web pages visited.

Model $p(x)$ from data

Identify unusual users by checking which have $p(x) < \varepsilon$

If it is unusual of user's behavior, send profile of users for further review / demand additional identification from users

(2) Manufacturing (aircraft engine)

(3) Monitoring computers in a data center.

$x^{(i)}$ = features of machine i

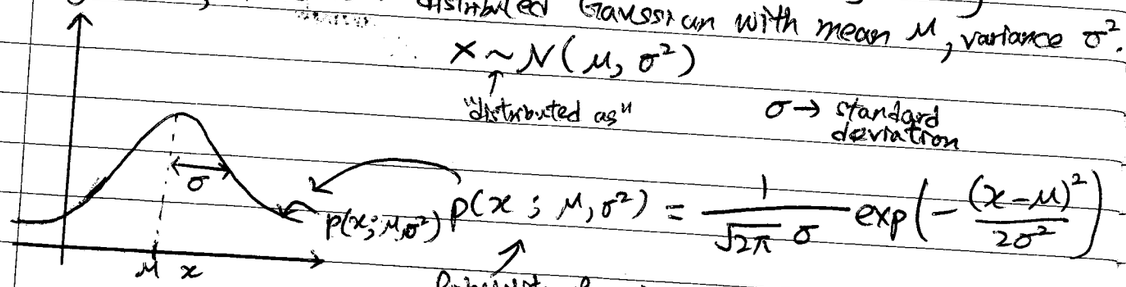
x_1 = memory use, x_2 = number of disk accesses/sec, x_3 = CPU load, x_4 = CPU load / network traffic...

Quiz:
Your anomaly detection system flags x as anomalous whenever $p(x) \leq \varepsilon$. Suppose your system is flagging too many things as anomalous that are not actually so (similar to supervised learning, these mistake are called false positives). What should you do?
—Try decreasing ε.

Anomaly detection — Gaussian distribution (normal distribution)
Gaussian (Normal) distribution / Gaussian probability density
Say $x \in \mathbb{R}$, if x is a distributed Gaussian with mean μ, variance σ^2.

$$x \sim N(\mu, \sigma^2)$$

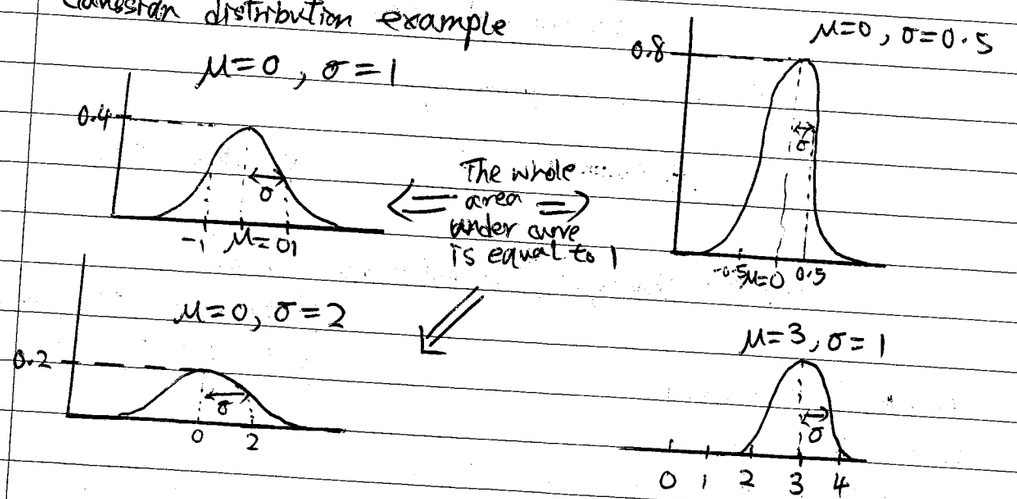
"distributed as" $\sigma \rightarrow$ standard deviation



$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Probability of x is parameterized by 2 parameters — μ, σ^2

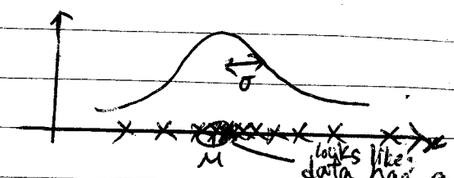
Gaussian distribution example



centered at 3 because it shifts over the entire Gaussian distribution

---

Parameter estimation
Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}$



$$x^{(i)} \sim N(\mu, \sigma^2)$$

Problem of parameter estimation
Is given data set, try to figure out, estimate what are the values of μ and σ^2

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)^2$$