

Artificial Intelligence and Machine Learning: A Guided Tour

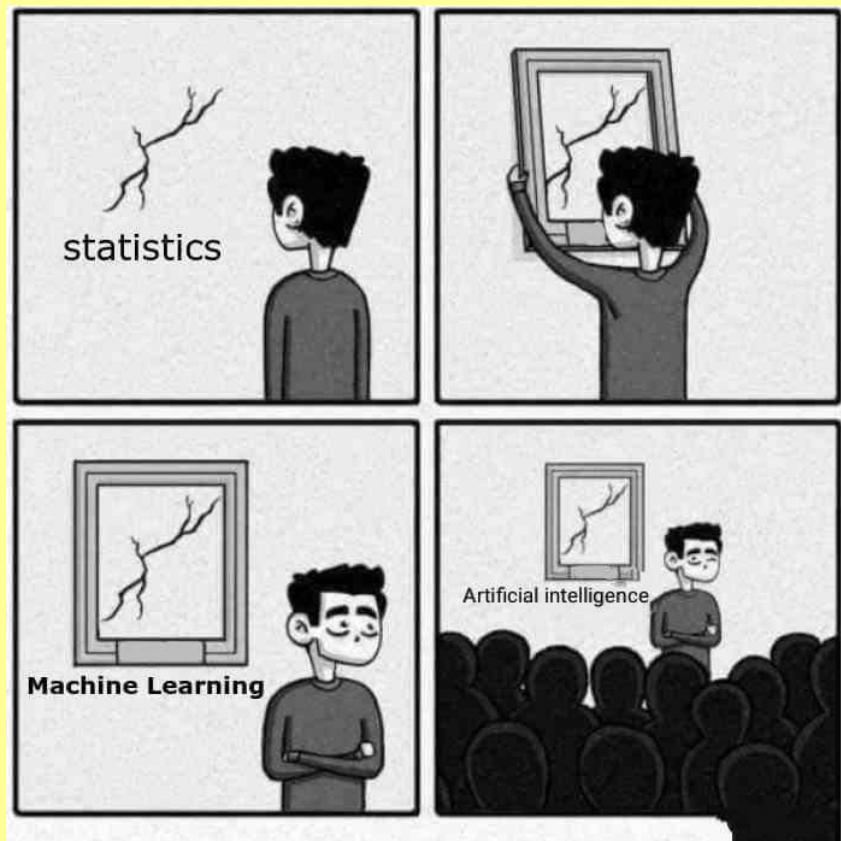
By

Swagatam Das

E-mail: swagatam.das@isical.ac.in

Electronics and Communication Sciences Unit,
Indian Statistical Institute, Kolkata – 700 108, India.

We start a little light....



No, Artificial
Intelligence is not
just glorified
Statistics

What is artificial intelligence?

- There is no clear consensus on the definition of AI
- John McCarthy coined the phrase AI in 1956

<http://www.formal.stanford.edu/jmc/whatisai/whatisai.html>

Q. What is artificial intelligence?

- A. It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human or *other* intelligence, but AI does not have to confine itself to methods that are biologically observable.

Q. Yes, but what is intelligence?

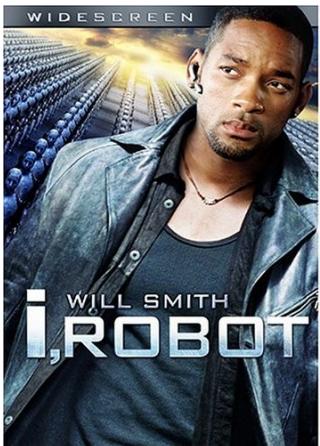
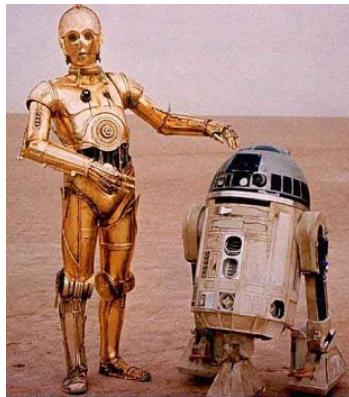
- A. Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals and some machines.

One Working Definition of AI

Artificial intelligence is the study of how to make computers do things that people are better at or would be better at if:

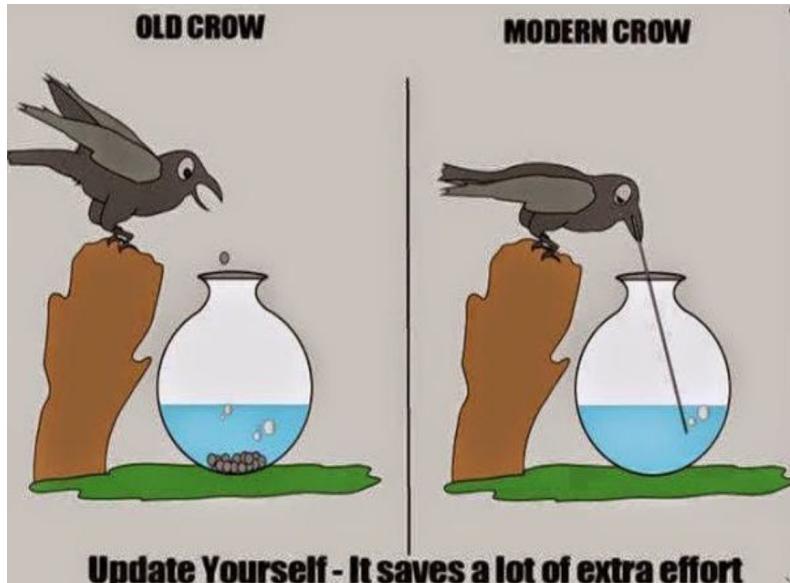
- they could extend what they do to a World Wide Web-sized amount of data and
- not make mistakes.

Artificial Intelligence in the Movies



Expecting more than the reality?

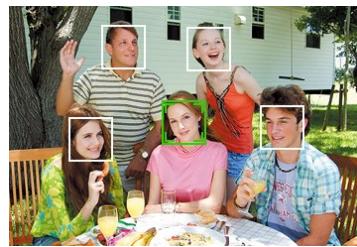
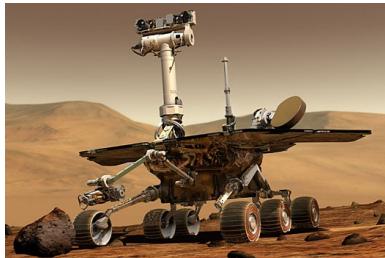
- We've all seen science fiction movies, and it can be tempting to think of machine learning as something that gives machines human-level intelligence. In reality, while machine learning can help you in many ways, it's better thought of as a specialized tool to analyze data than as a silver bullet to solve any problem.



Artificial Intelligence in Real Life

A young science (\approx 50 years old)

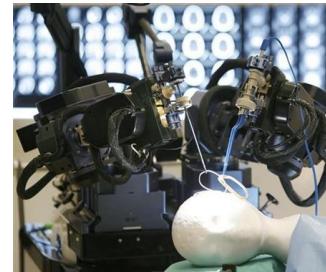
- Exciting and dynamic field, lots of uncharted territory left
- Impressive success stories
- “Intelligent” in specialized domains
- Many application areas



Face detection



Formal verification



Why the interest in AI?



Labor



Science

Google™
YAHOO!

Search engines



Medicine/
Diagnosis



Appliances

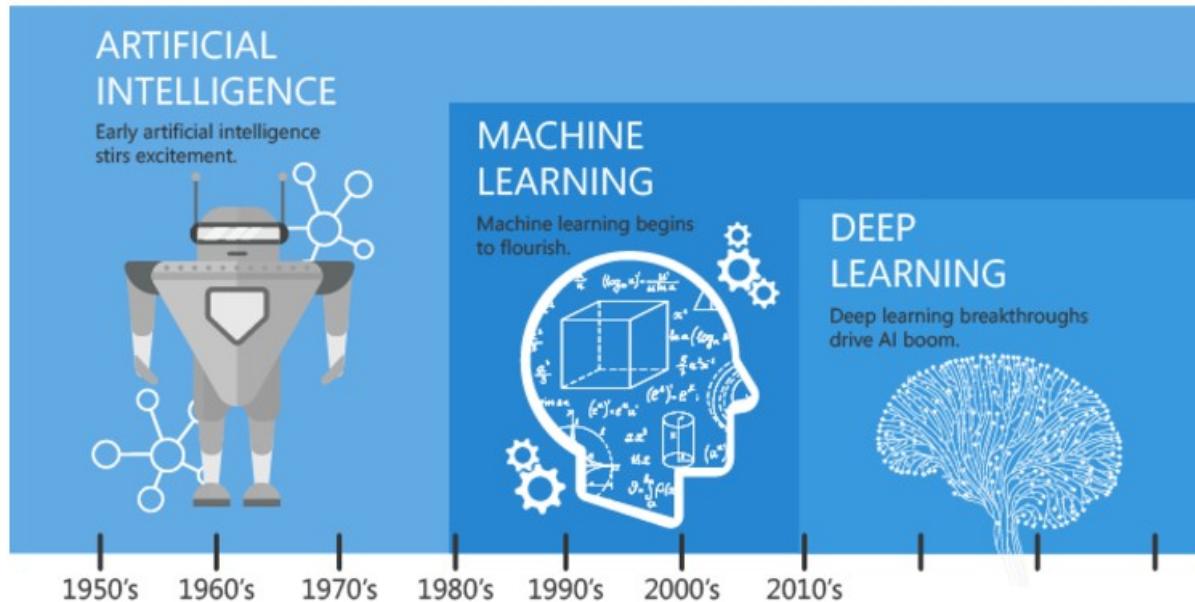
What else?

Weak and Strong AI Claims

- Weak AI:
 - Machines can be made to act as if they were intelligent.
- Strong AI:
 - Machines that act intelligently have real, conscious minds.

AI 2.0: Extract and Use Knowledge from New Data Driven Knowledge Sources

- Paradigm Shift in Science
 - First 3 Paradigms: Experiment, Theory, Simulation
 - Rutherford, Bohr, Oppenheimer
 - 4th Paradigm: Data Driven Science
- Create Next Generation AI systems
 - Data Driven AI systems
 - To Solve Previously Unsolved Problems
- Previously Unavailable Sources of Data
 - Knowledge from Big Data:
 - **Data Driven Learning of Models and Algorithms**
 - Knowledge from Multiple (Cross) Media:
 - **Social Media Intelligence Gathering**
 - **From All Language Sources**
 - **From All the Media: Text, Speech, Image and Video**
 - Knowledge from Crowd Intelligence:
 - **Global Brain: from Individual Intelligence to Collective Intelligence**
 - Knowledge from Augmented Intelligence:
 - **Human-Machine Hybrid Intelligence for Collaborative Problem Solving**
 - Knowledge from Unmanned Autonomous Vehicles:
 - **Intelligence from Collaborating Teams of Robots**
- Automatic Discovery of New Knowledge
 - Machine Learning using Big Data
 - **Deep Learning**



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

Machine Learning

- **Herbert Alexander Simon:**
“Learning is any process by which a system improves performance from experience.”
- “Machine Learning is concerned with computer programs that automatically improve their performance through experience.”



Herbert Simon
Turing Award 1975
Nobel Prize in Economics 1978

What is Machine Learning?

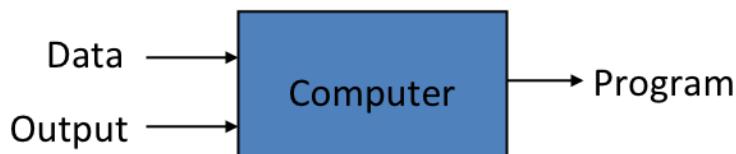
Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with **data**, without being explicitly programmed.

The name machine learning was coined in 1959 by Arthur Samuel. - [Wikipedia](#)

Traditional Programming

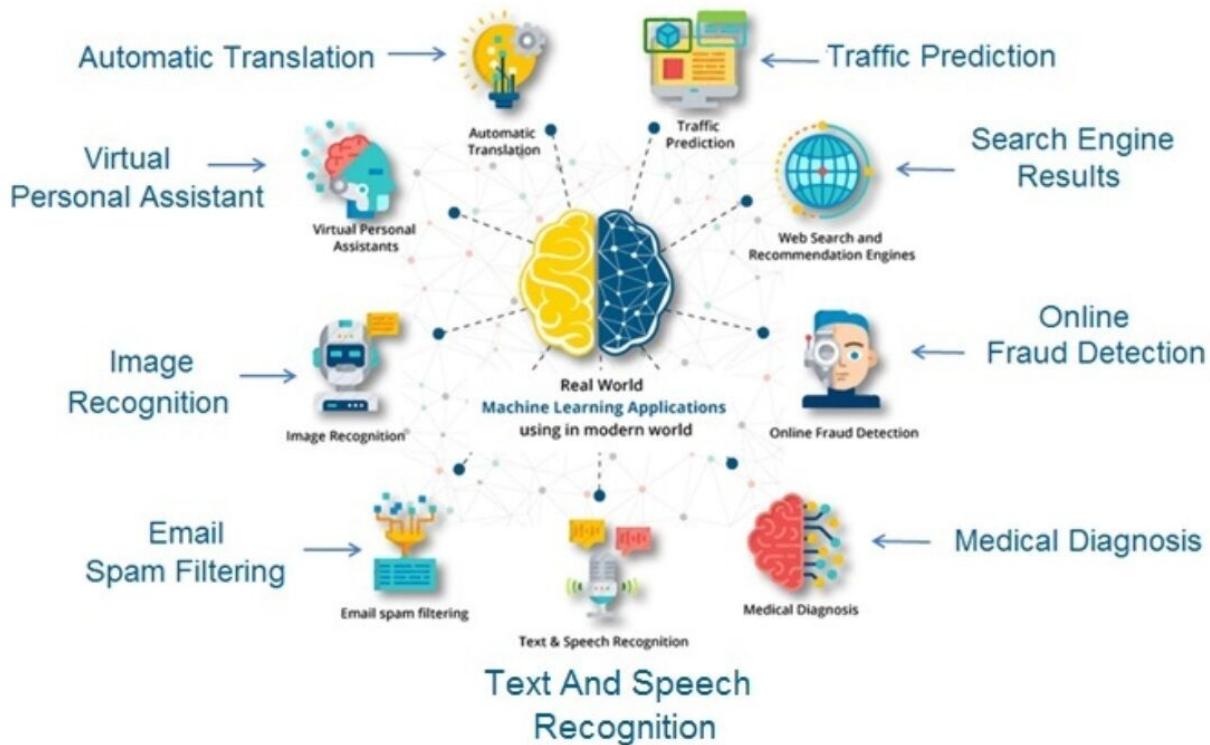


Machine Learning



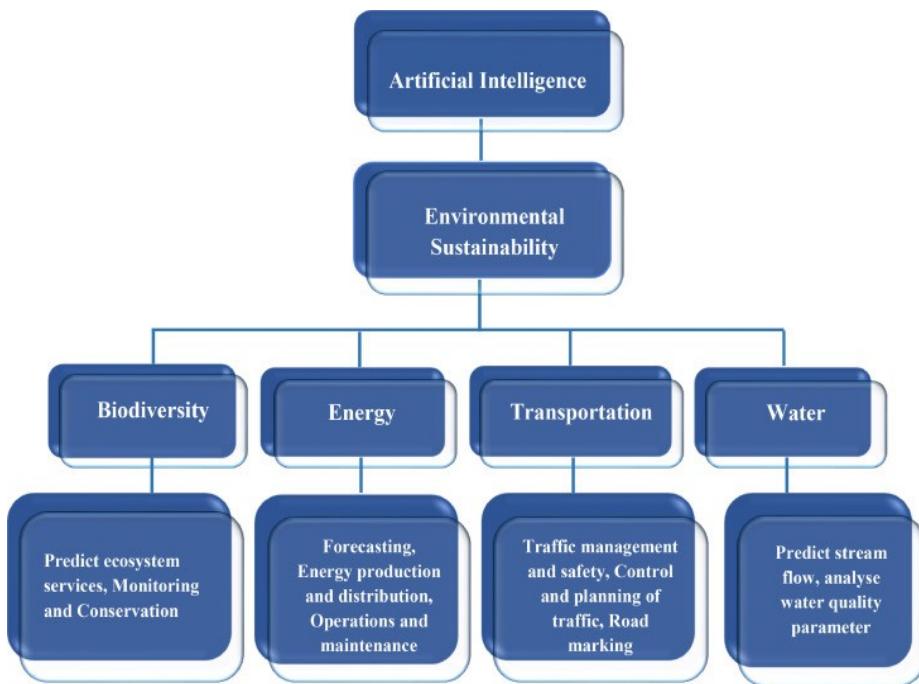
13

Top Real-World Examples of Machine Learning



14

Artificial Intelligence and Sustainability



You can start to understand what machine learning is by looking at where and how it's used. It's actually being used increasingly in a wide range of technology products.

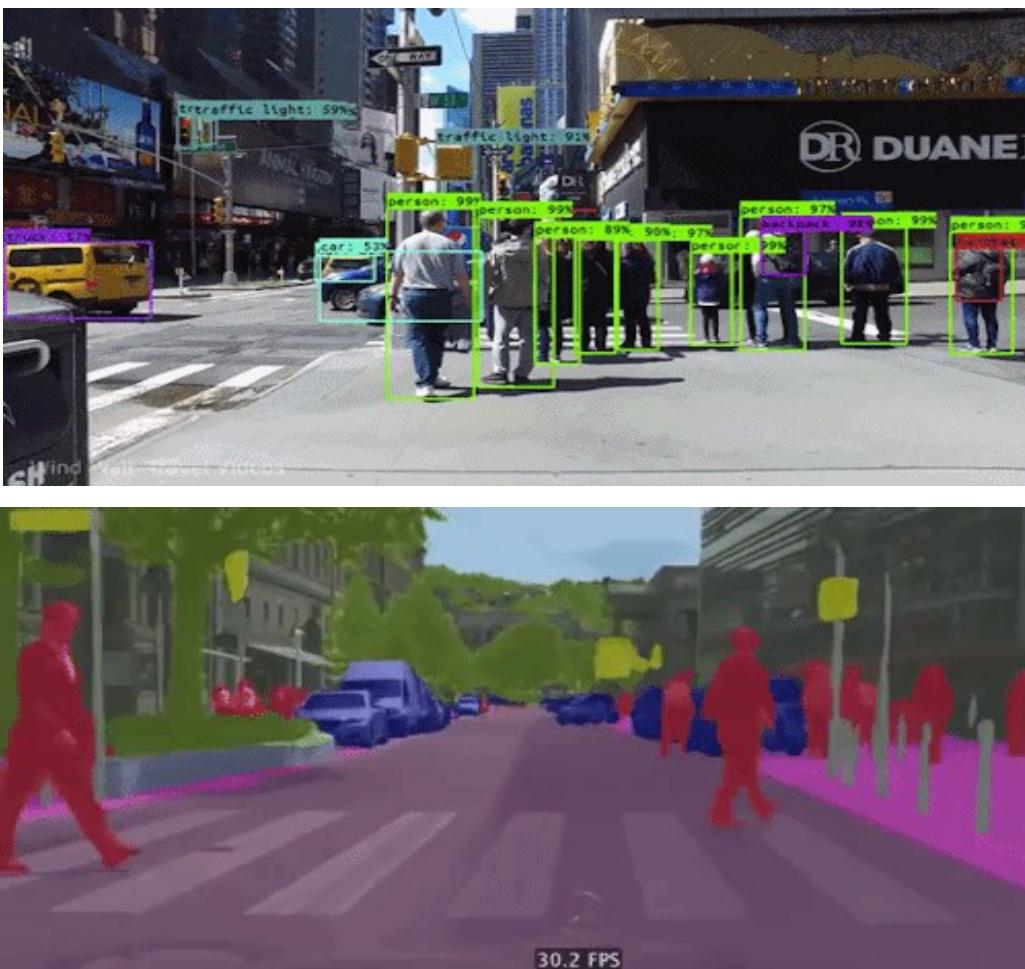
Here are a few examples of machine learning in our everyday lives:

- recommendation systems (on sites like Amazon and Netflix)
- Vehicle detection, classification, and self driving cars
- spam classification

Books you may like

Page 1 of 7

Mathematics for Machine Learning Marc Peter Deisenroth 4.5 stars, 226 reviews Paperback \$46.99	Deep Learning (Adaptive Computation and Machine Learning series) Ian Goodfellow 4.5 stars, 1,320 reviews Hardcover \$39.00	An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) Gareth James 4.5 stars, 1,077 reviews Hardcover \$43.99	Storytelling with Data: A Data Visualization Guide for Business... Cole Nussbaumer Knaflic 4.5 stars, 1,884 reviews Paperback #1 Best Seller in Information Management \$26.99
--	---	---	---



The data and the goal

- **Data:** A set of data records (also called examples, instances or cases) described by
 - **k attributes/variables/features:** A_1, A_2, \dots, A_k .
 - **a class:** Each example is labelled with a pre-defined class.
- **Goal:** To learn a classification model f from the data that can be used to predict the classes of new (future, or test) cases/instances.

An example: data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

19

An example: the learning task

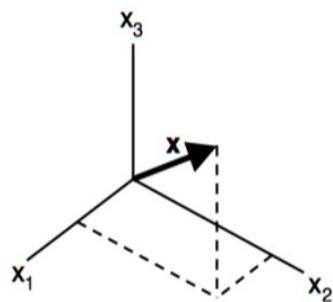
- Learn a classification model from the data
- Use the model to classify future loan applications into
 - Yes (approved) and
 - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

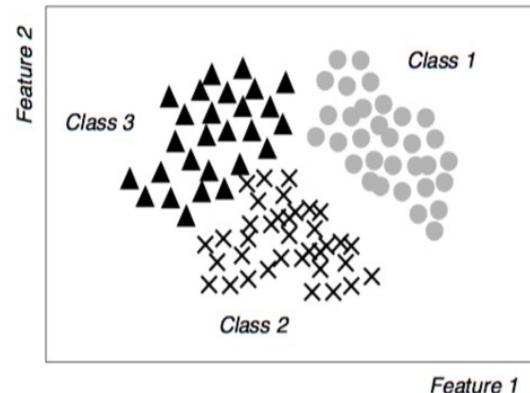
20

How to represent objects to a ‘Machine’?

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_d \end{bmatrix}$$



Feature vector



Feature space (3D)

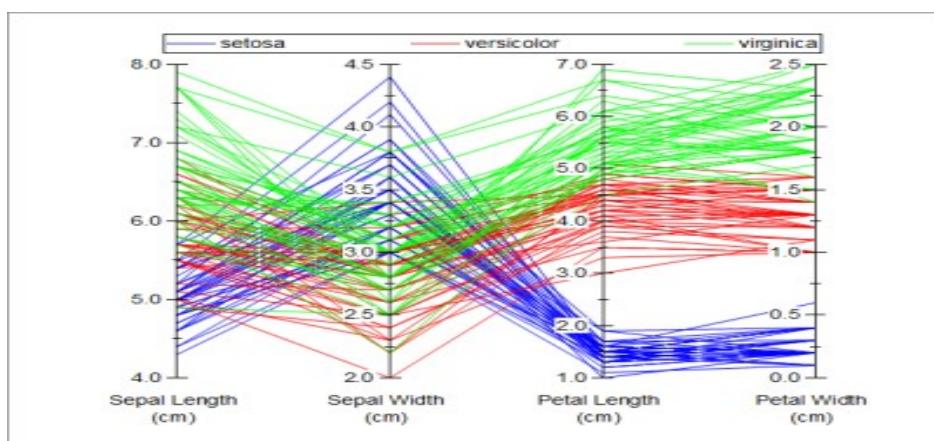
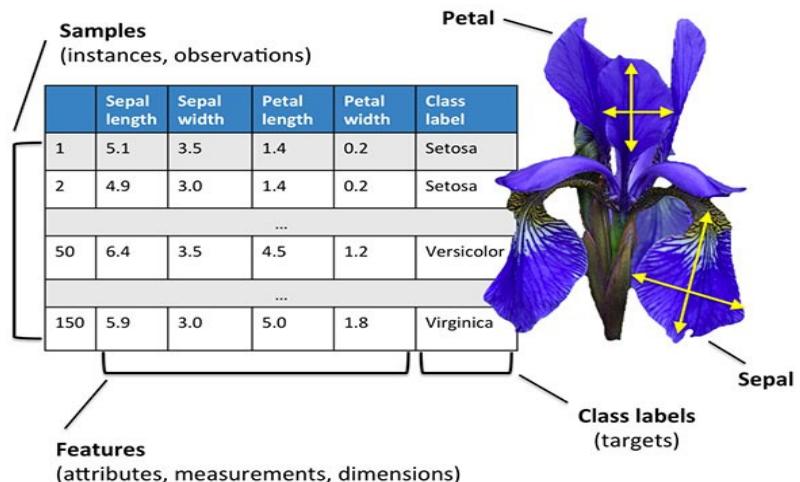
Scatter plot (2D)

21

Ways to visualize Multivariate Data:



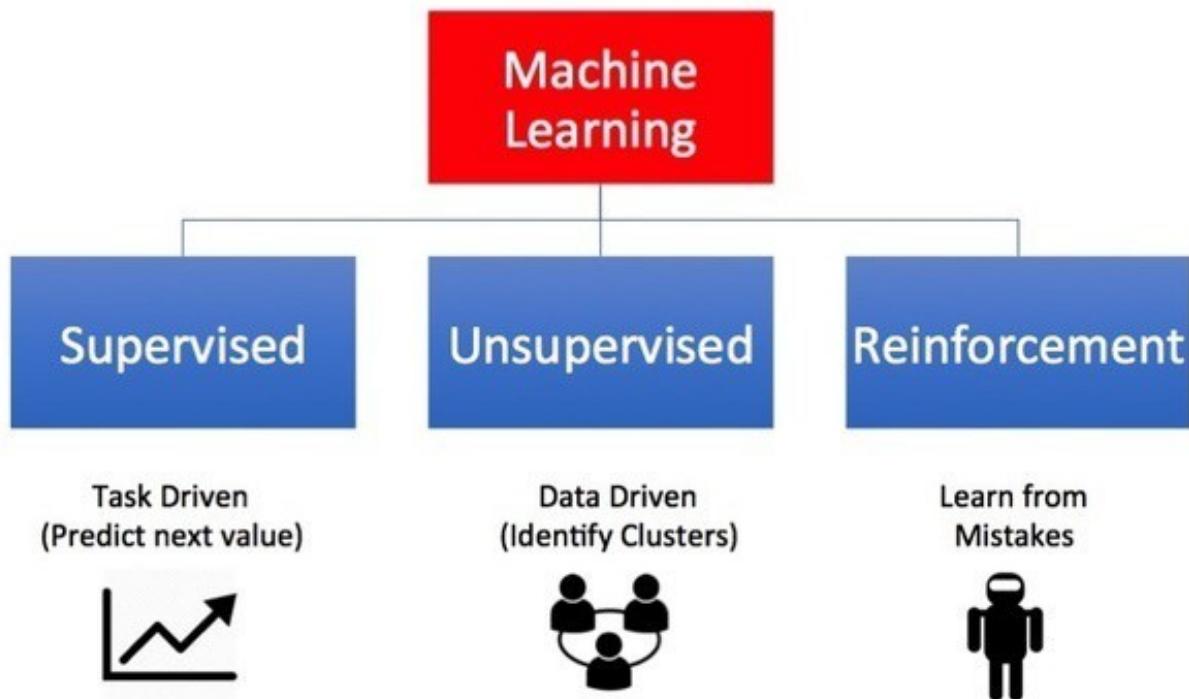
IRIS DATASET



The parallel
coordinate plots

22

Types of Machine Learning



<https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f>

Conventionally....

Machine Learning Problems

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

The machine learning framework

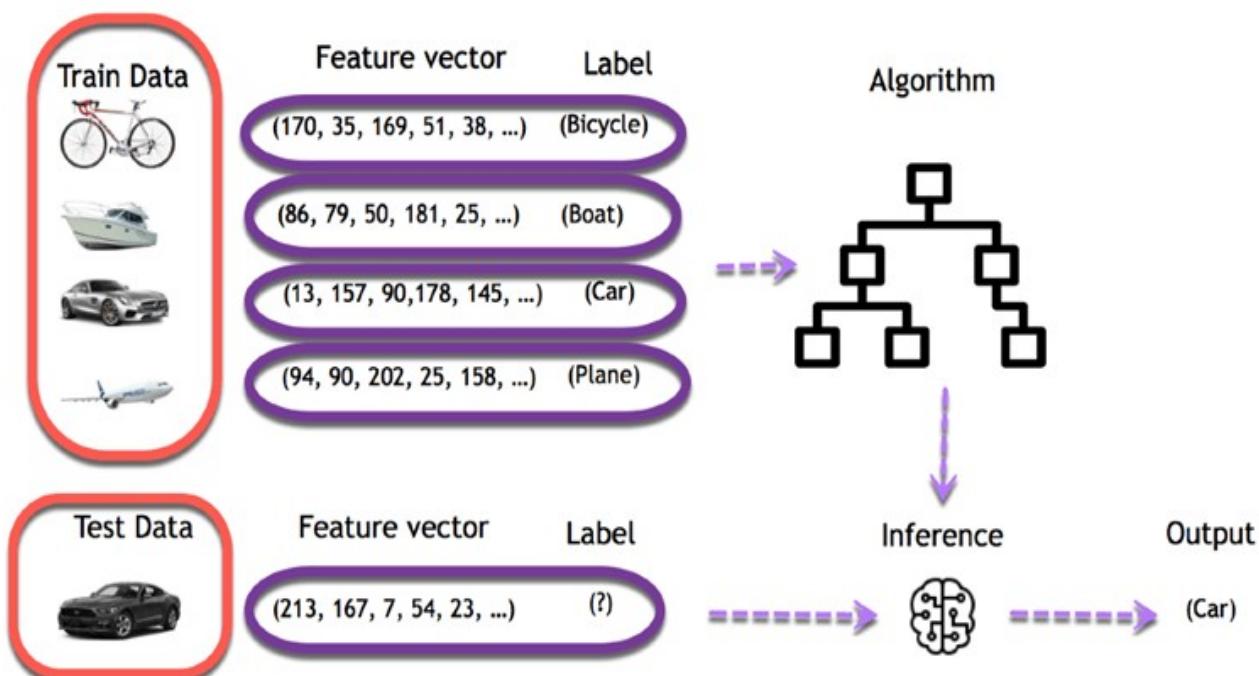
$$y = f(x)$$

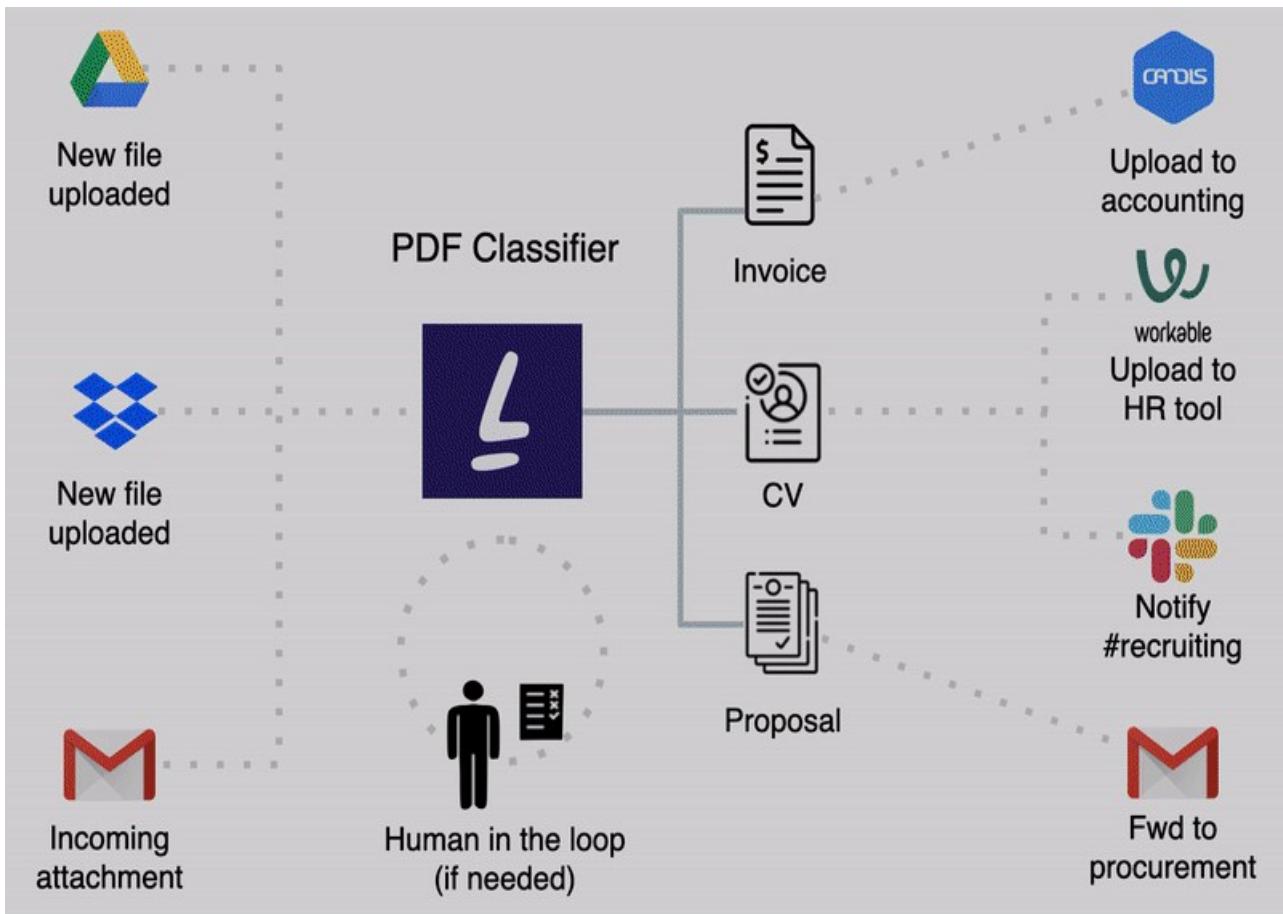
↑ ↑ ↗
output prediction function Image feature

- **Training:** given a *training set* of labeled examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* x and output the predicted value $y = f(x)$

■ Slide credit: L. Lazebnik

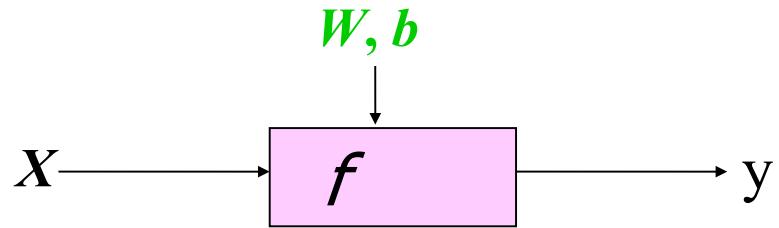
The Supervised Learning Business: Core of Pattern Recognition



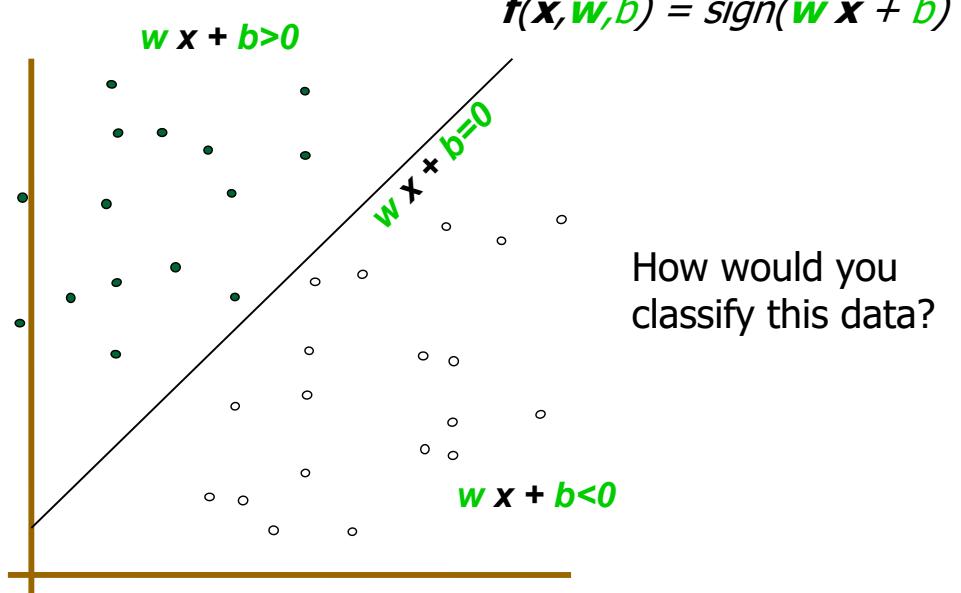


27

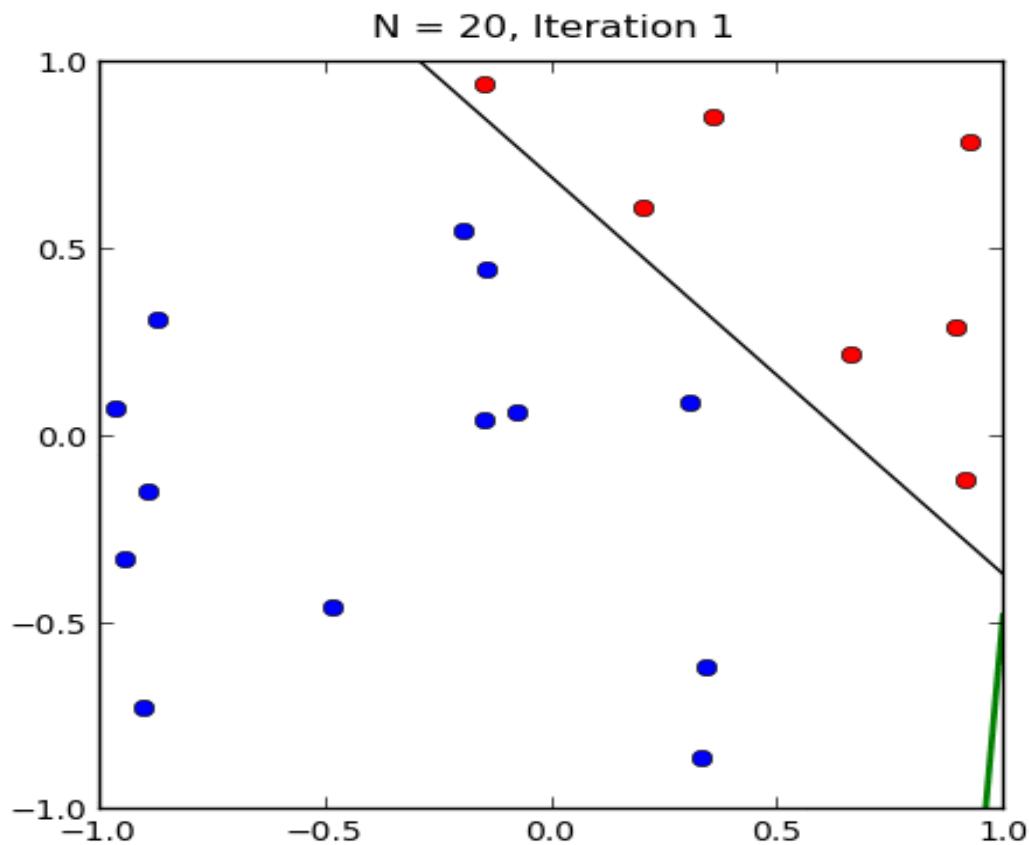
Models!



- denotes +1
- denotes -1



A Linear Classifier in Action.....



Is ML really so hard?

Input: X

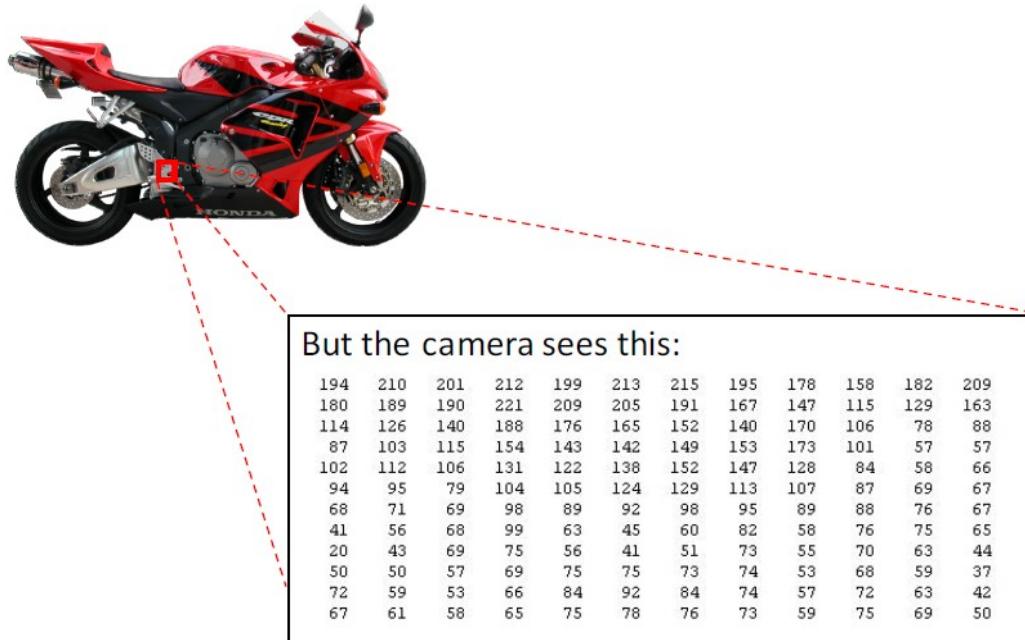
Output: Y



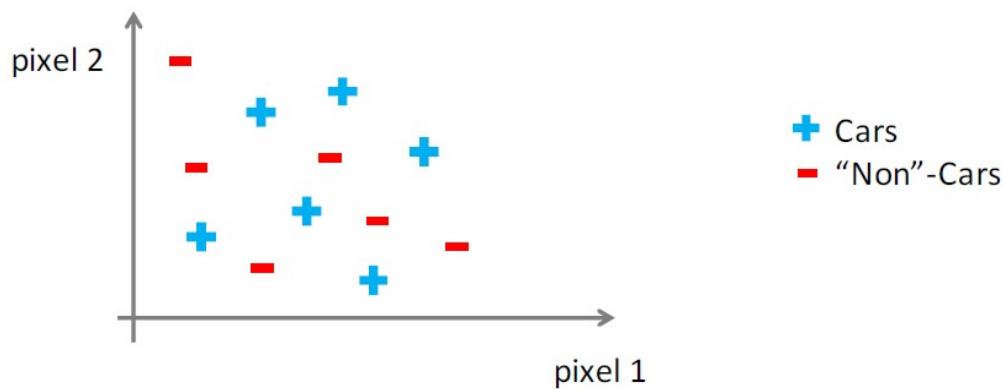
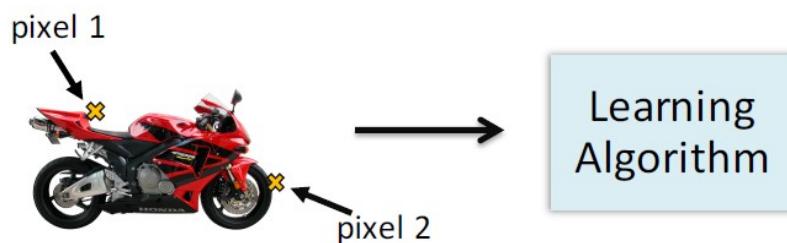
Label "motorcycle"

Why is it hard?

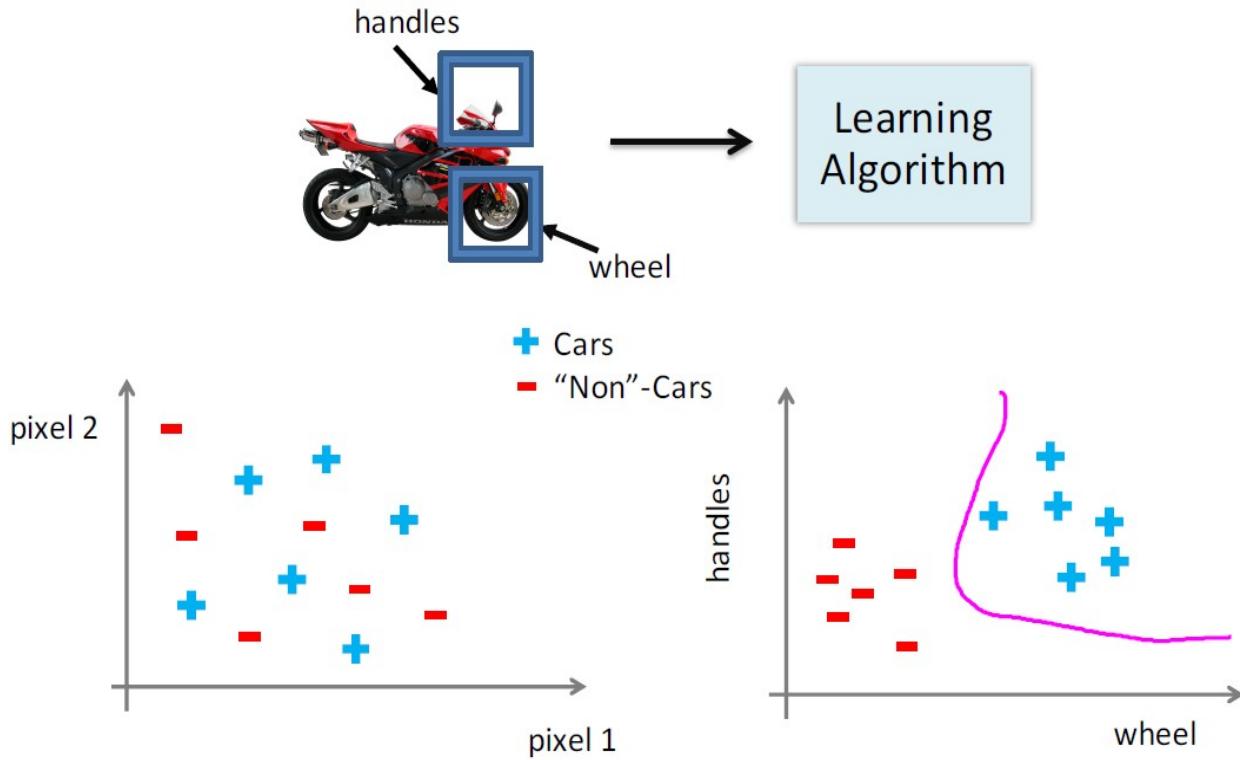
You see this



Raw Image Representation



Better Feature Representation?



But still classification is not that easy....especially nowadays...

Labradoodle or fried chicken



Barn owl or apple



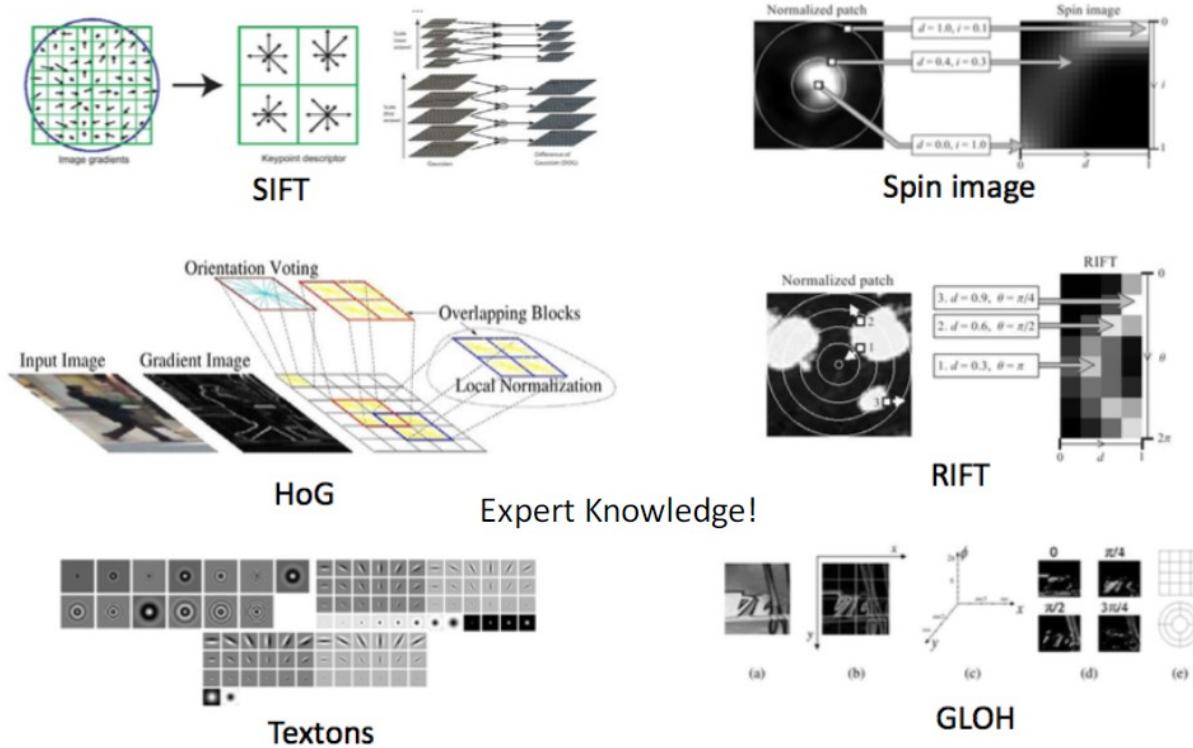
@teenybiscuit

Chihuahua or muffin



@teenybiscuit

Perhaps a lot of it depends on proper feature representations: **Feature Engineering!**



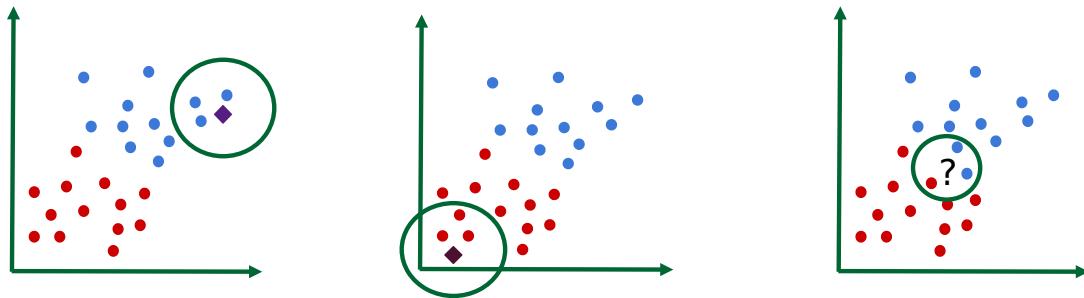
So many classifiers over the years....

- k-nearest neighbor
- SVM
- Decision Trees
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- **The Deep Learning Systems**
- And so on.....

And then comes the **No Free Lunch Theorem of ML**.....

The *k*-Nearest Neighbor Classifier

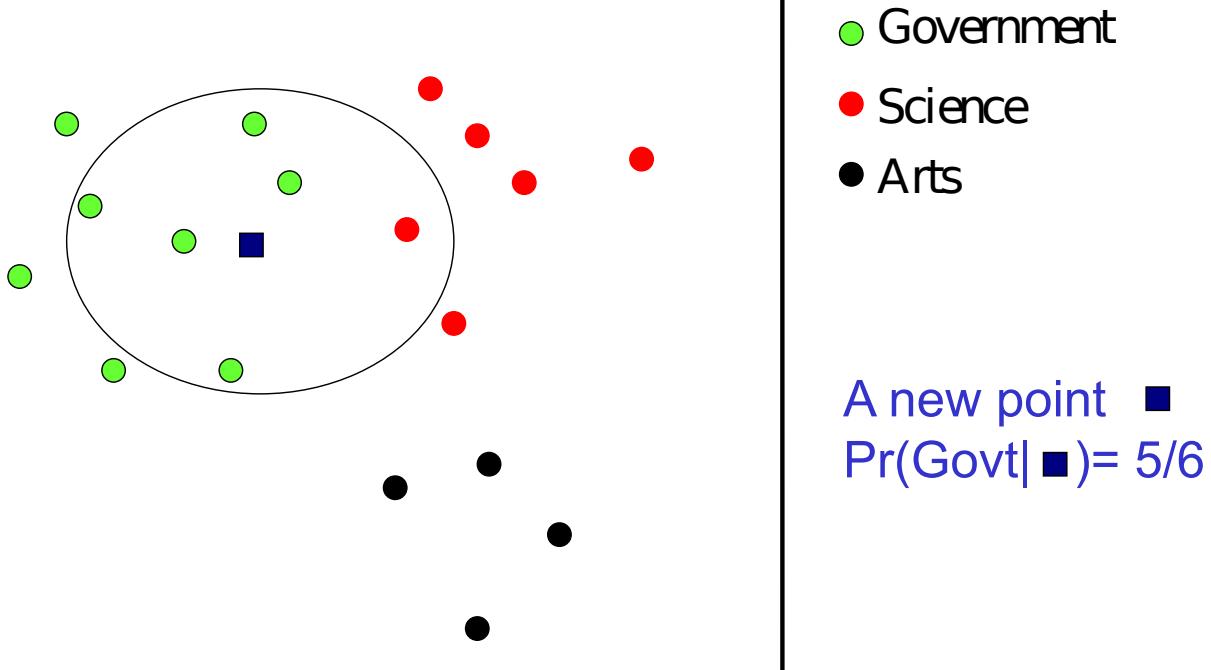
The **k**-Nearest Neighbor (**kNN**) classifier (**Fix and Hodges 1958, Cover and Hart 1967**) labels a test point y , by that class which has the majority number of representatives among the training set neighbors of y .



Reference:

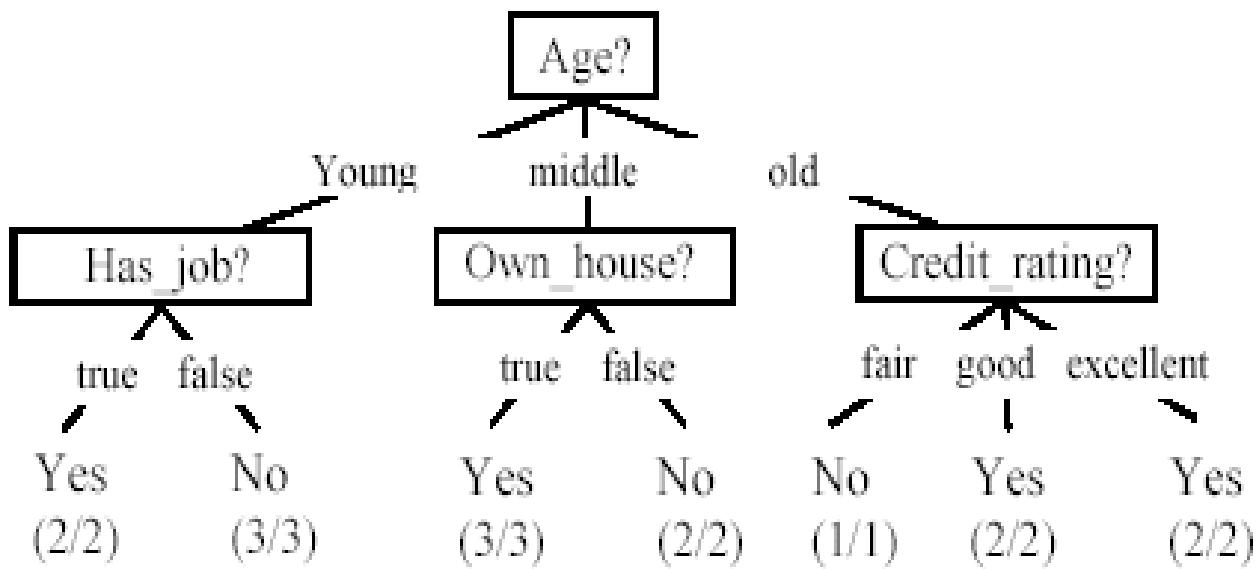
- E. Fix and J.L. Hodges, Discriminatory analysis-nonparametric discrimination: consistency properties, Technical Report, California Univ Berkeley (1951)
- T. M. Cover and P. E. Hart, "Nearest neighbour pattern classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, 1967.

Example: $k=6$ (6NN)



The Decision Tree Classifier

Decision nodes and leaf nodes (classes)



41

Decision Tree Classifiers....

The Loan Data Reproduced...

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Use the decision tree

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

No

```

graph TD
    Root[Age?] --> Young[Young]
    Root --> Middle[middle]
    Root --> Old[old]
    Young --> HasJobTrue[Has job? true]
    Young --> HasJobFalse[Has job? false]
    Middle --> OwnHouseTrue[Own house? true]
    Middle --> OwnHouseFalse[Own house? false]
    Old --> CreditRatingFair[fair]
    Old --> CreditRatingGood[good]
    Old --> CreditRatingExcellent[excellent]
    
```

Yes (2/2)	No (3/3)	Yes (3/3)	No (2/2)	No (1/1)	Yes (2/2)	Yes (2/2)
--------------	-------------	--------------	-------------	-------------	--------------	--------------

43

Is the decision tree unique?

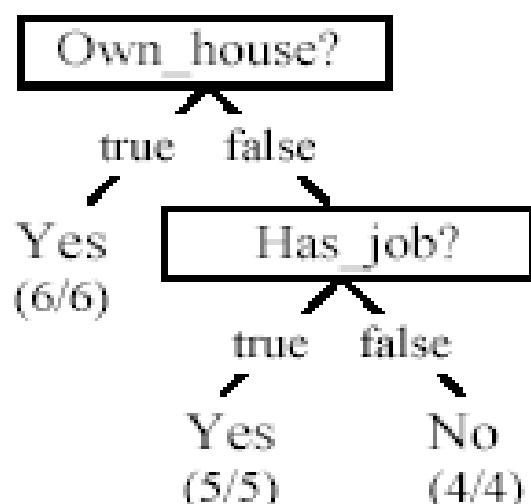
No. Here is a simpler tree.

We want **smaller tree** and **accurate tree**.

Easy to understand and perform better.

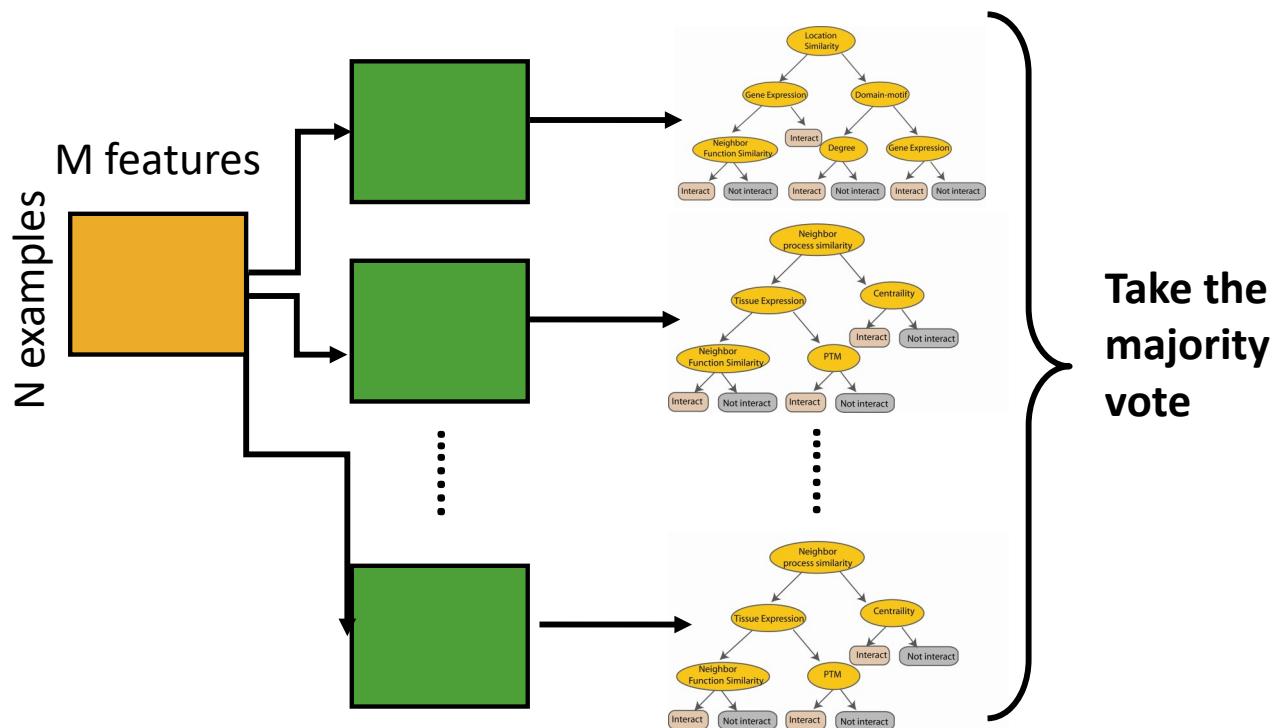
Finding the best tree is
NP-hard.

All current tree building
algorithms are heuristic
algorithms

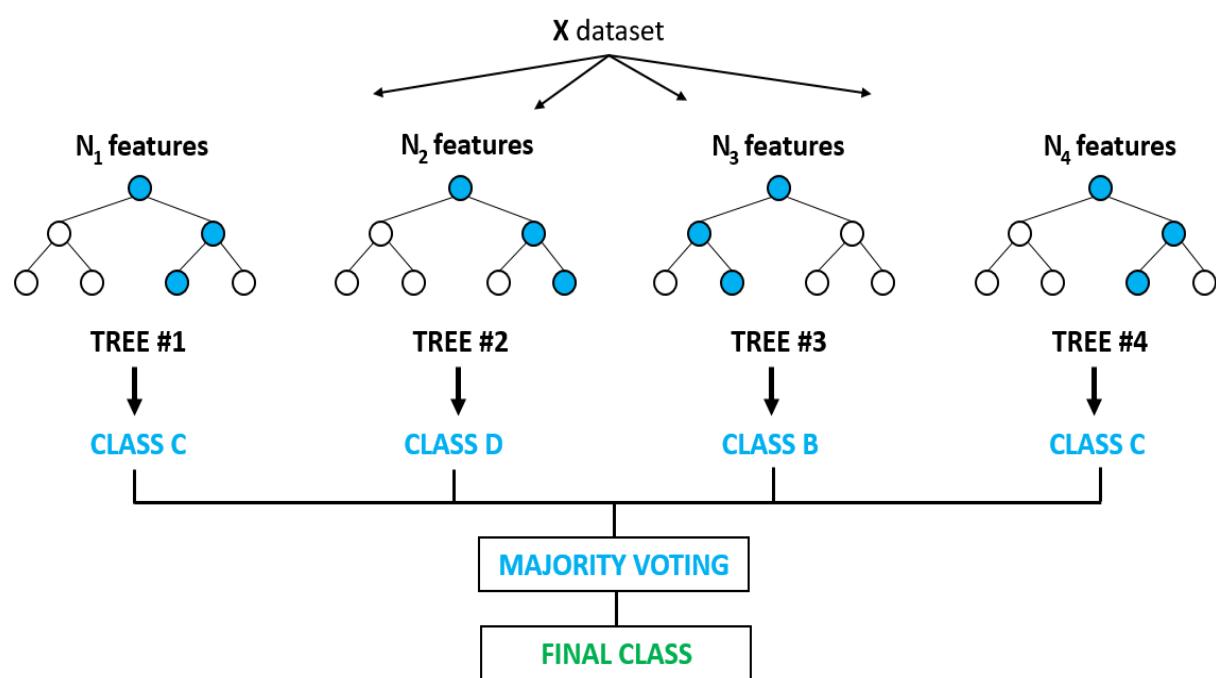


44

Random Forest Classifier



Thus,.....

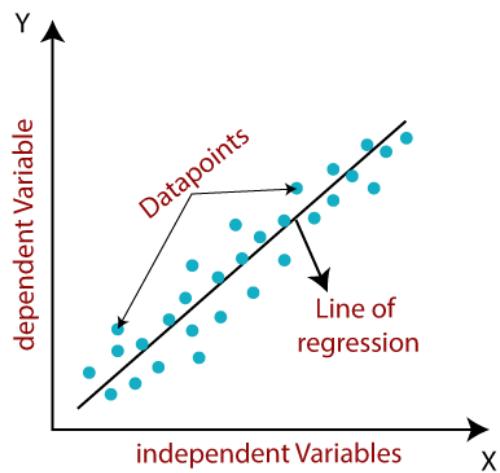
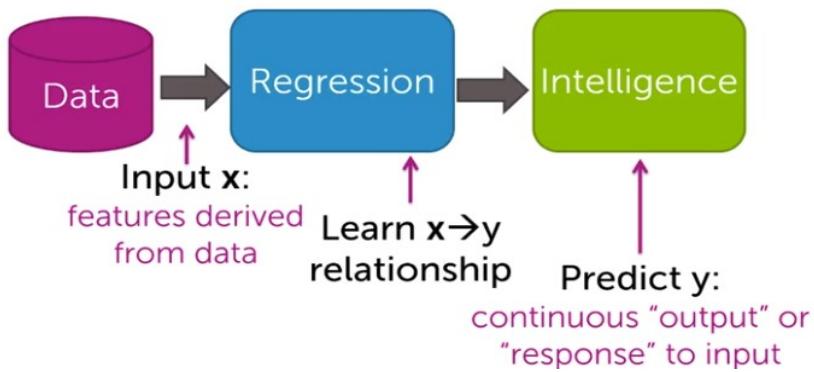


Machine Learning Problems

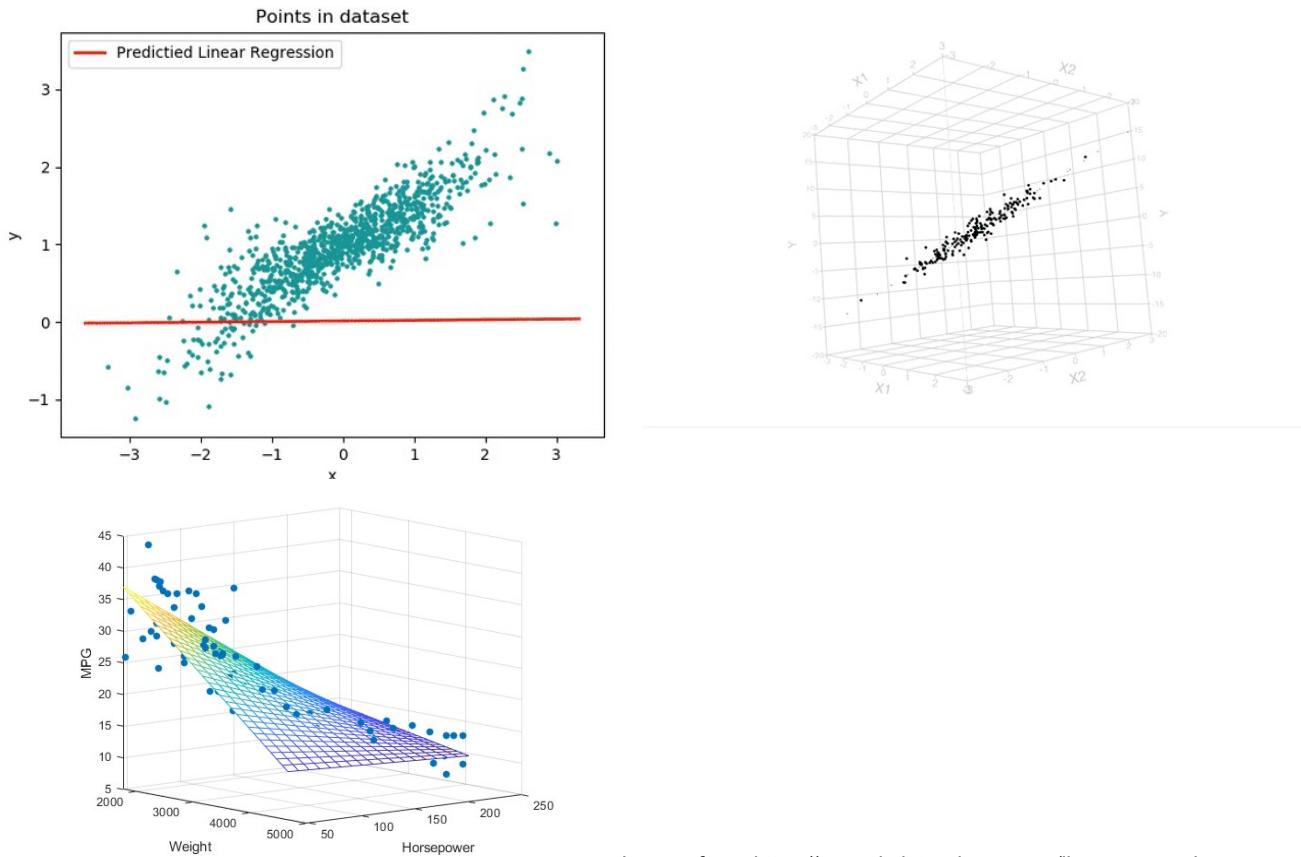
	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

What is regression?

From features to predictions

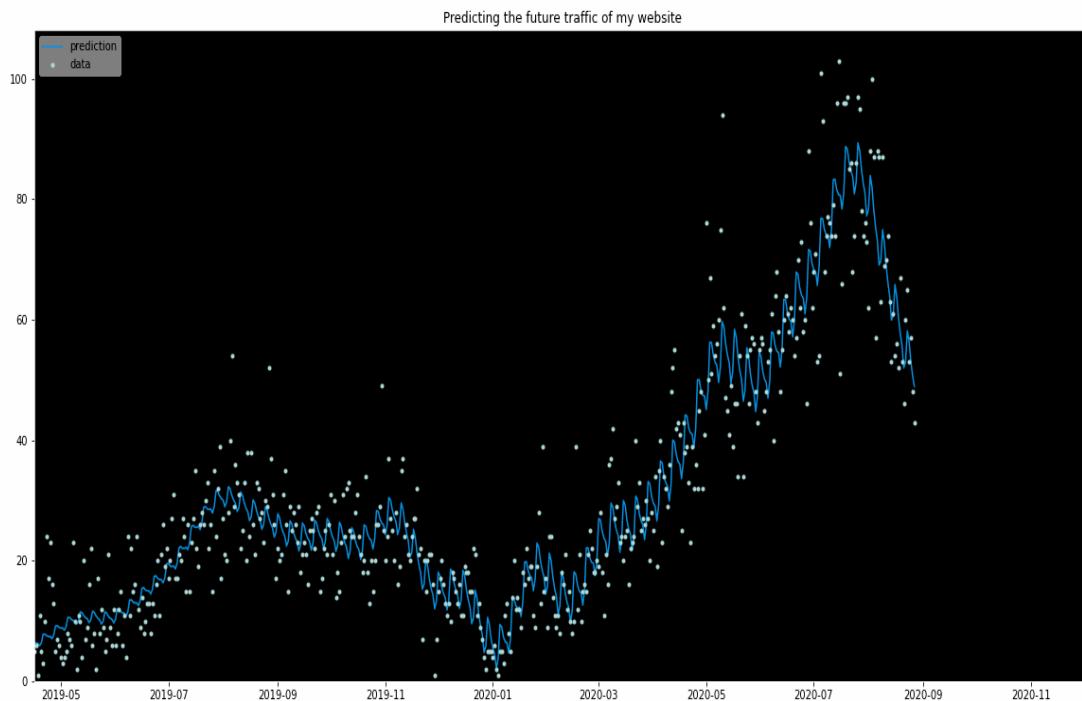


Visualizing Linear Regression....



Images from: <https://towardsdatascience.com/linear-regression-5100fe32993a>

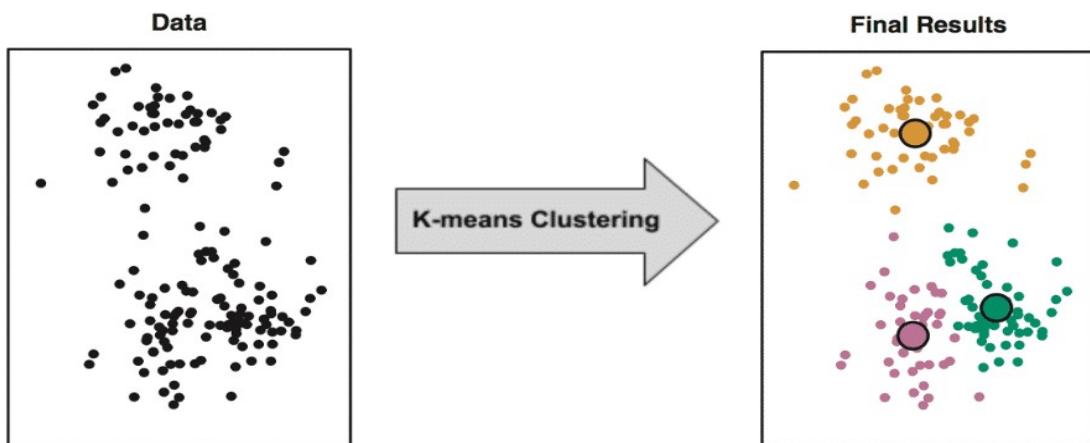
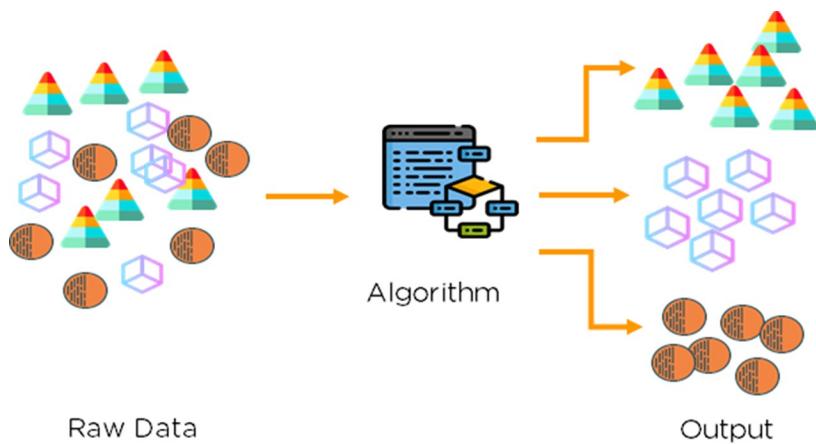
Forecasting: Often a Regression Problem



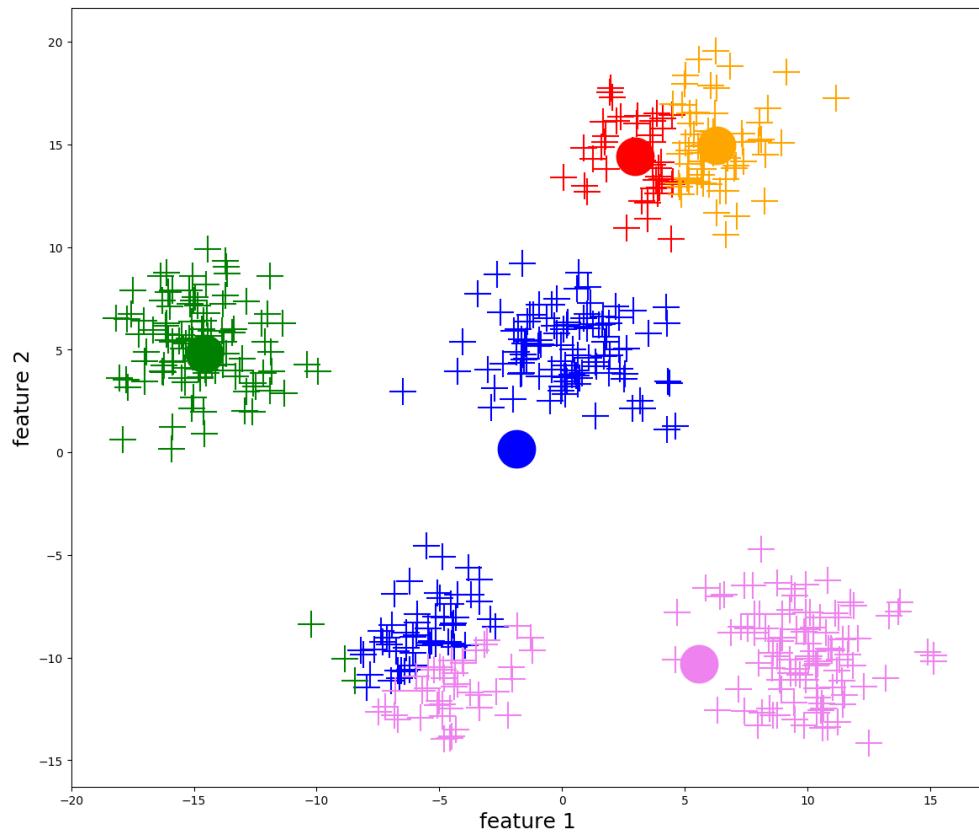
Machine Learning Problems

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

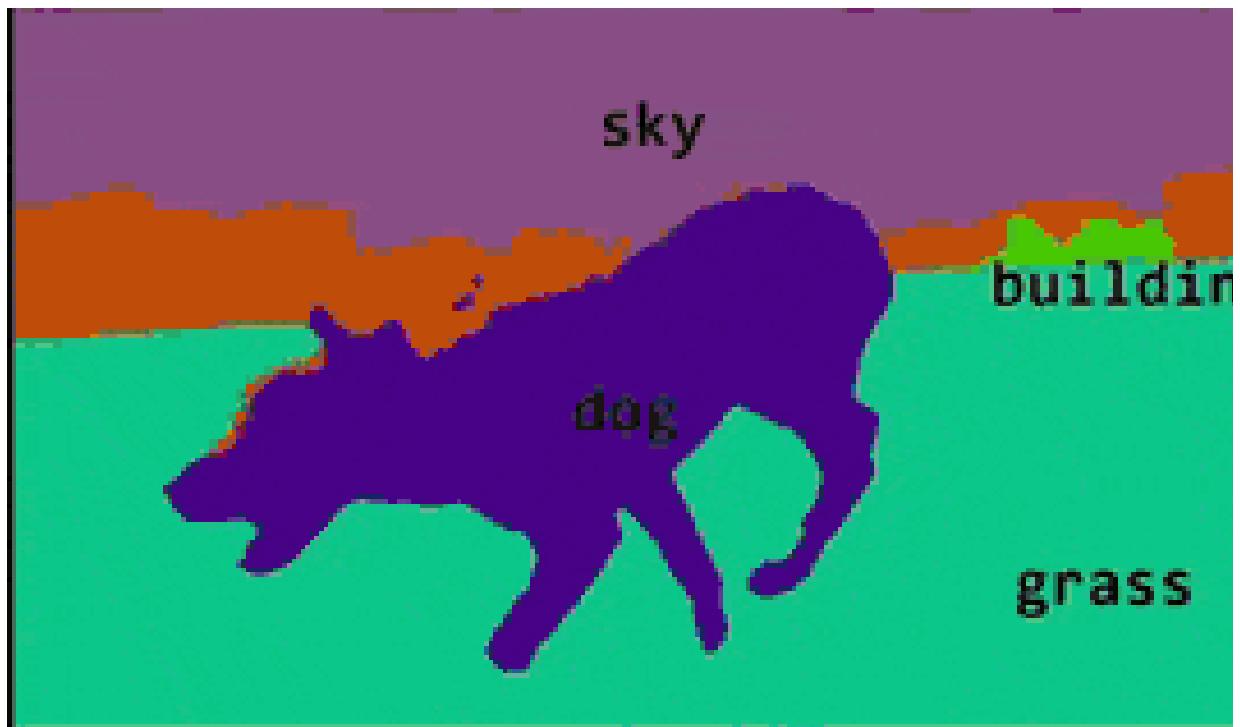
Clustering:



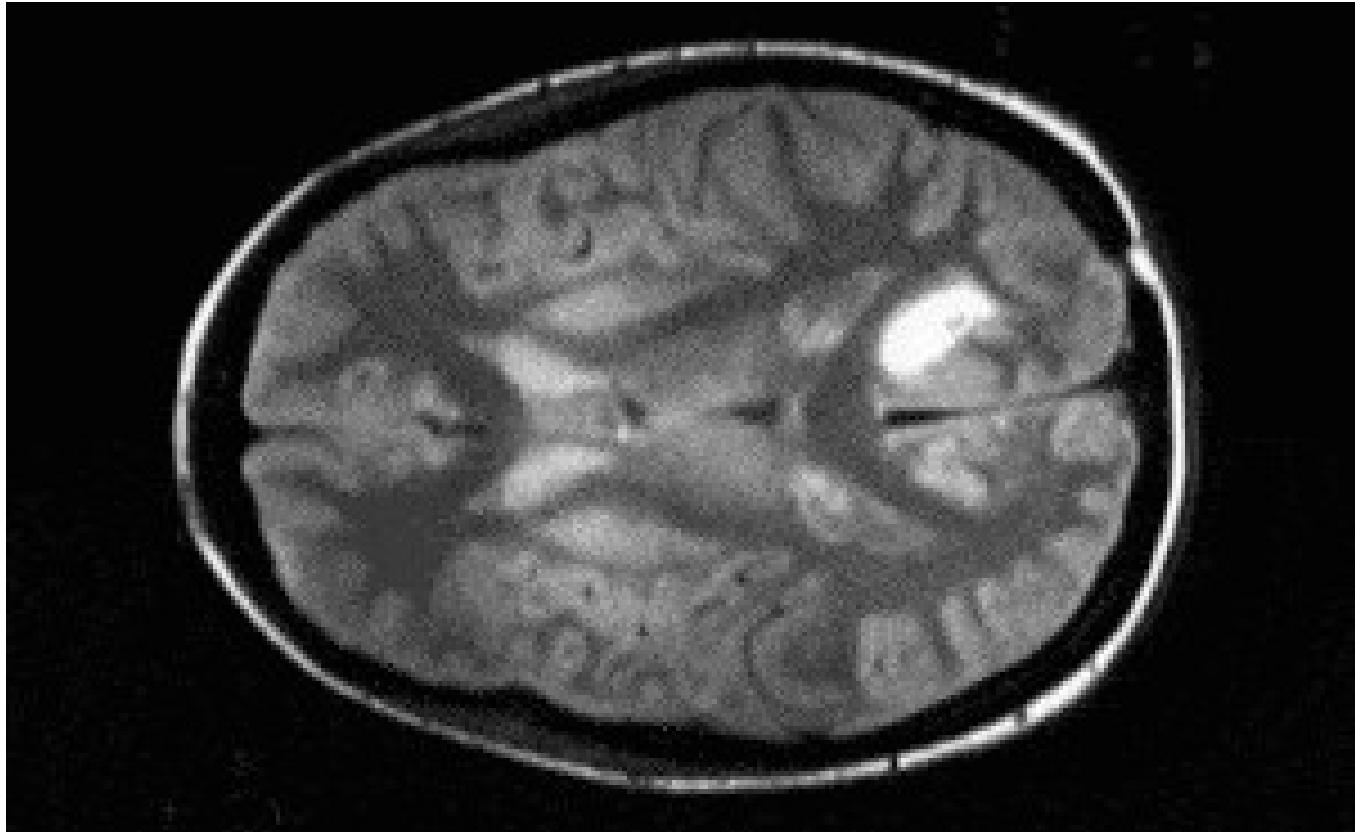
K-means clustering in action....



Semantic Segmentation of Images



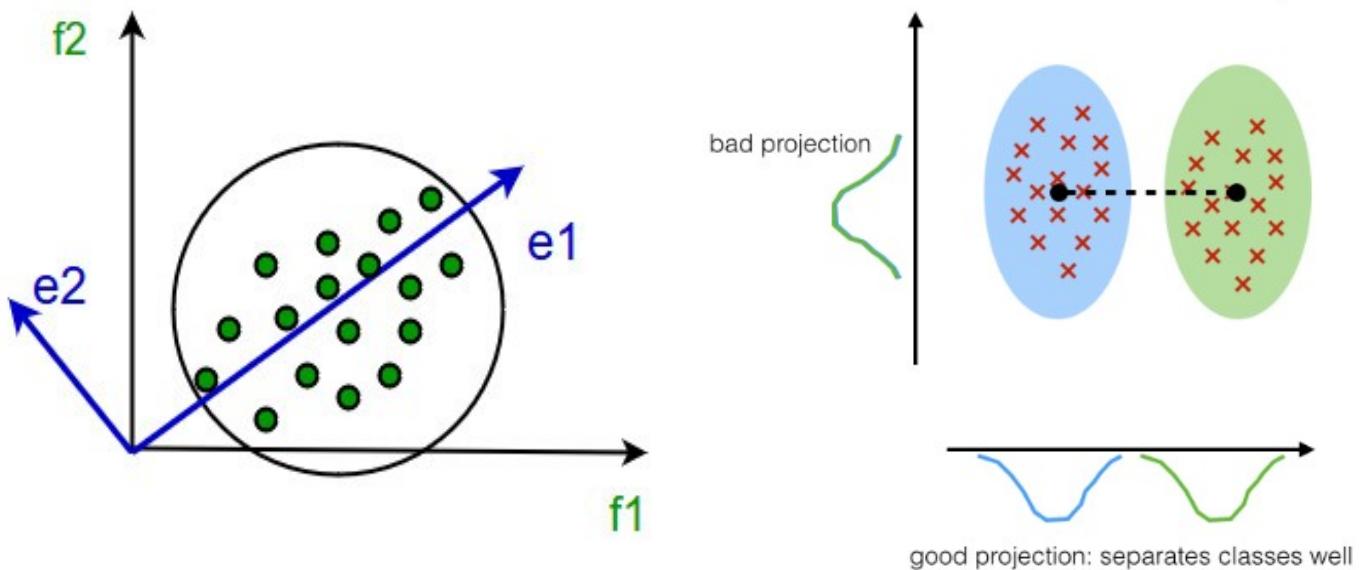
Tumor Segmentation from Medical Images



Machine Learning Problems

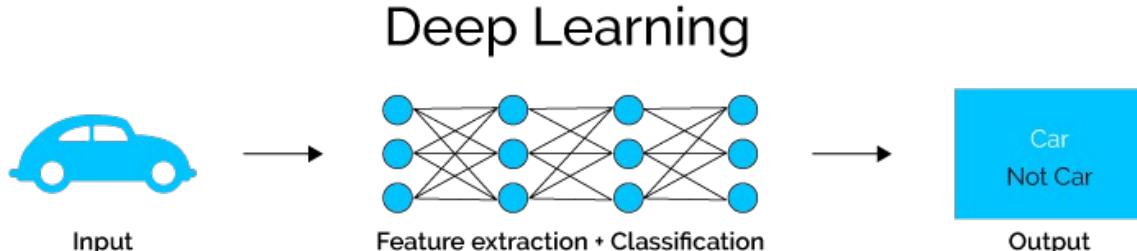
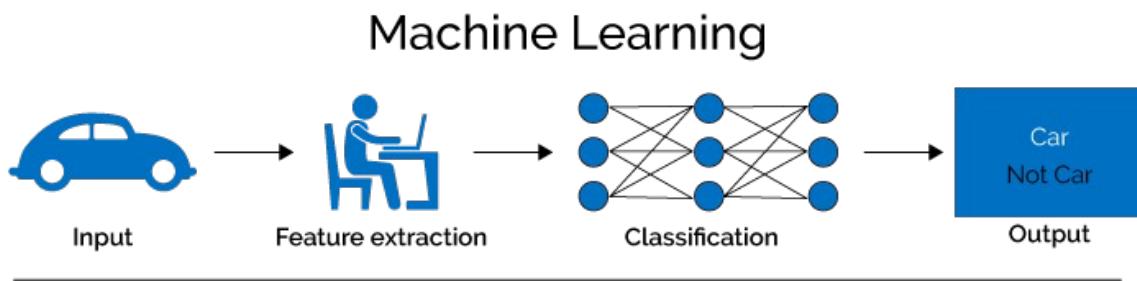
	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

Dimensionality Reduction....

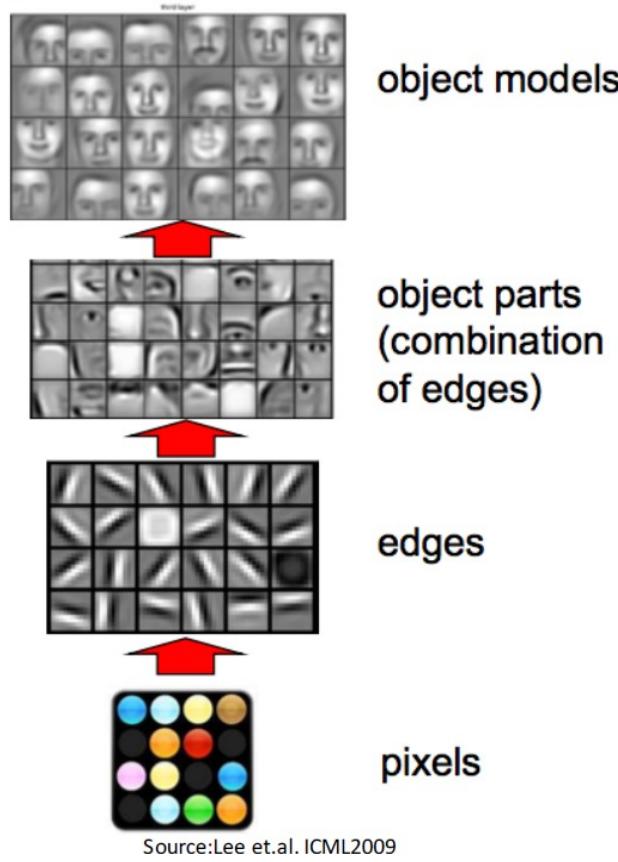


What is Deep Learning (DL) ?

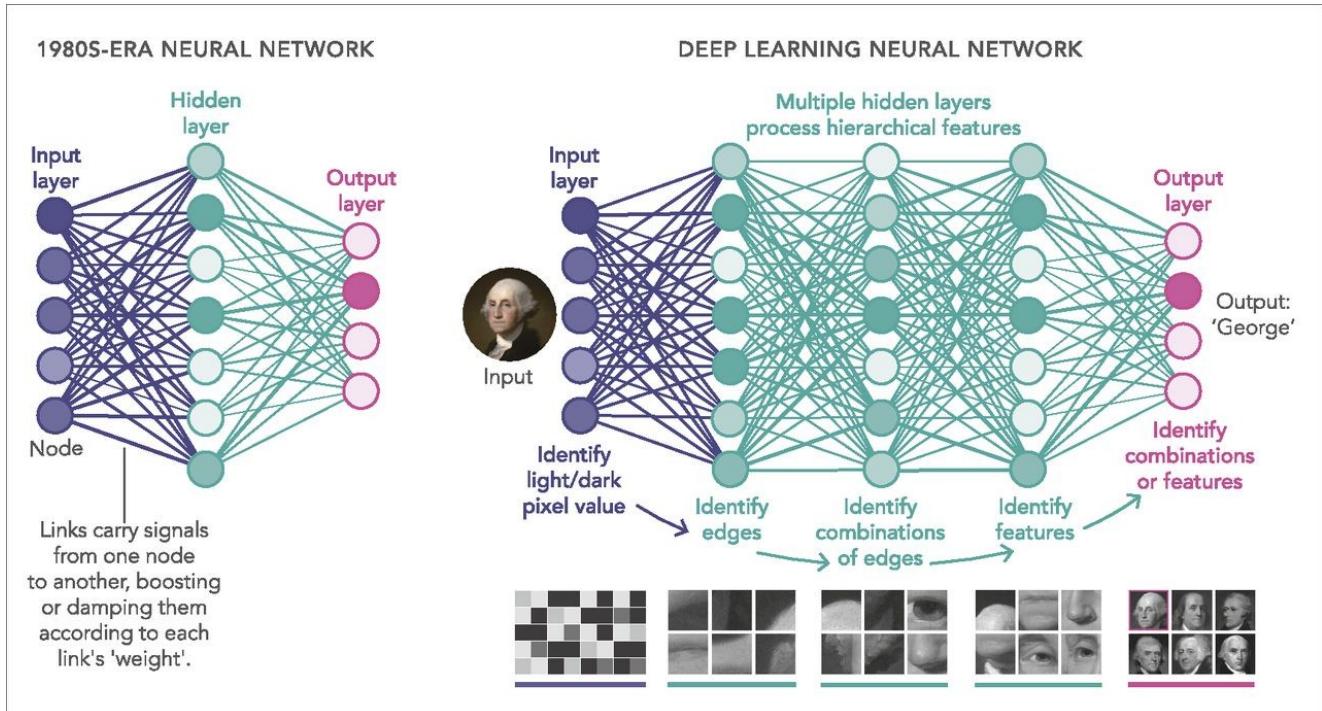
A machine learning subfield is the learning of **representations** in data.
Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**.
If you provide the system **tons of information**, it begins to understand it and respond in useful ways.



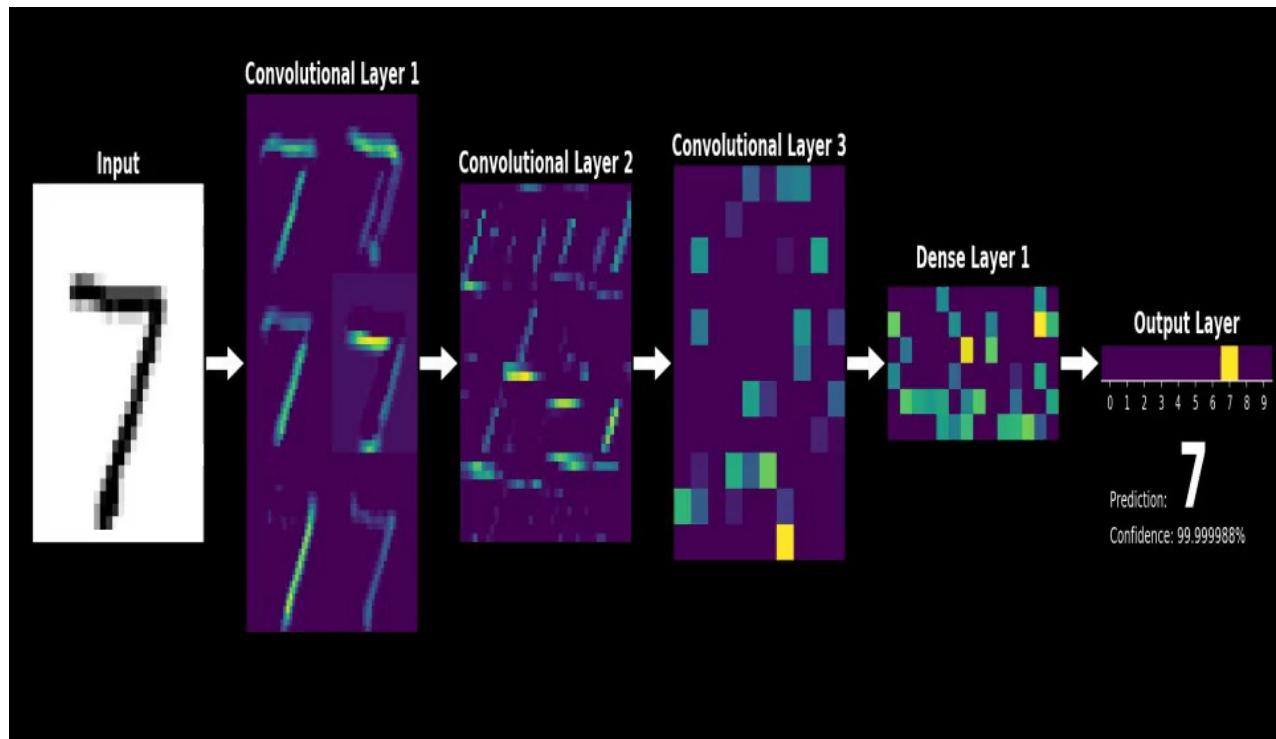
Deep Learning: learn representations!



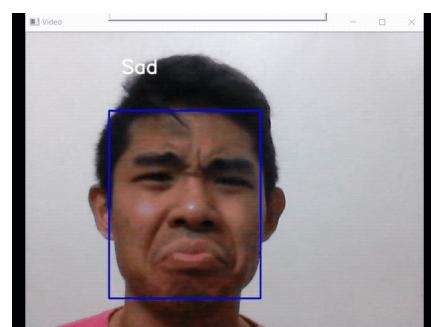
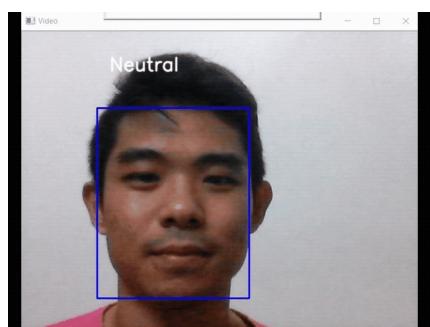
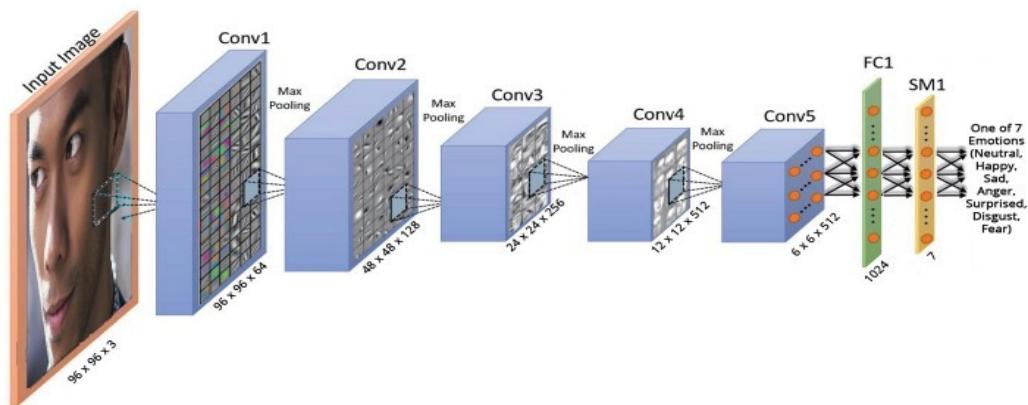
So what are these Deep Nets in a nutshell?.....

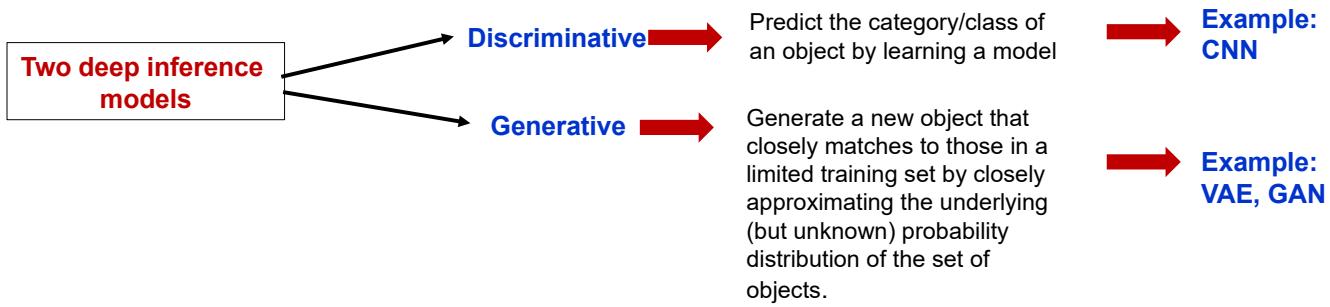


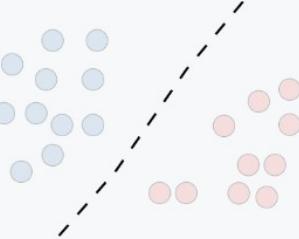
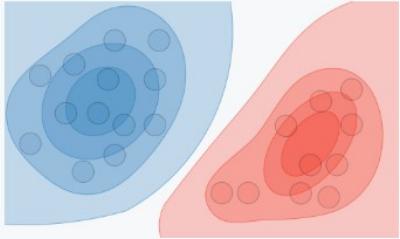
Visualizing a Deep Learning Work flow....



Emotion Recognition with Convolutional Nets....





	Discriminative model	Generative model
Goal	Directly estimate $P(y x)$	Estimate $P(x y)$ to then deduce $P(y x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

63

Generative Adversarial Networks (GANs)

The 2014 breakthrough:

- Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua. "Generative Adversarial Networks". [arXiv:1406.2661](https://arxiv.org/abs/1406.2661), NIPS 2014

40,000+ citations till date!

"the most interesting idea in the last 10 years in machine learning."

- Yann LeCun, Director of AI research at Facebook

"the future of Artificial Intelligence" - Yoshua Bengio, 2018 Turing Awardee for pioneering Deep Learning

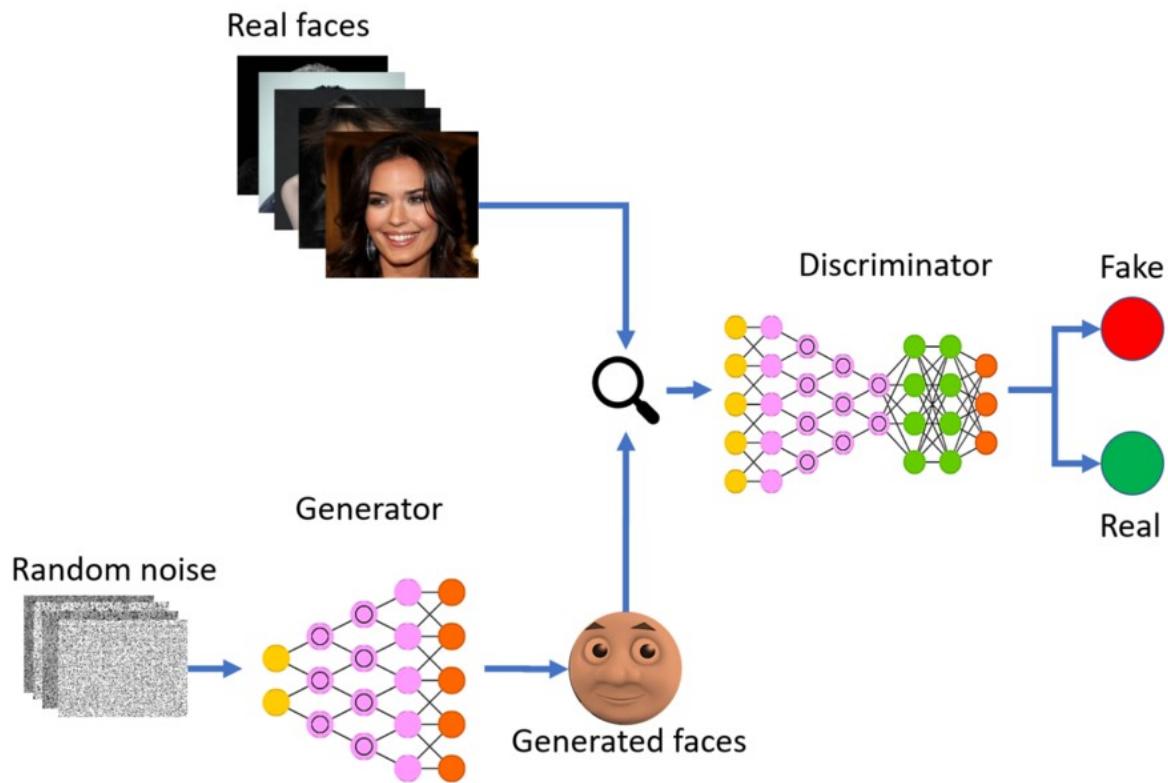
GANs were used to create the 2018 painting *Edmond de Belamy* which sold for \$432,500



"What I cannot create, I do not understand"

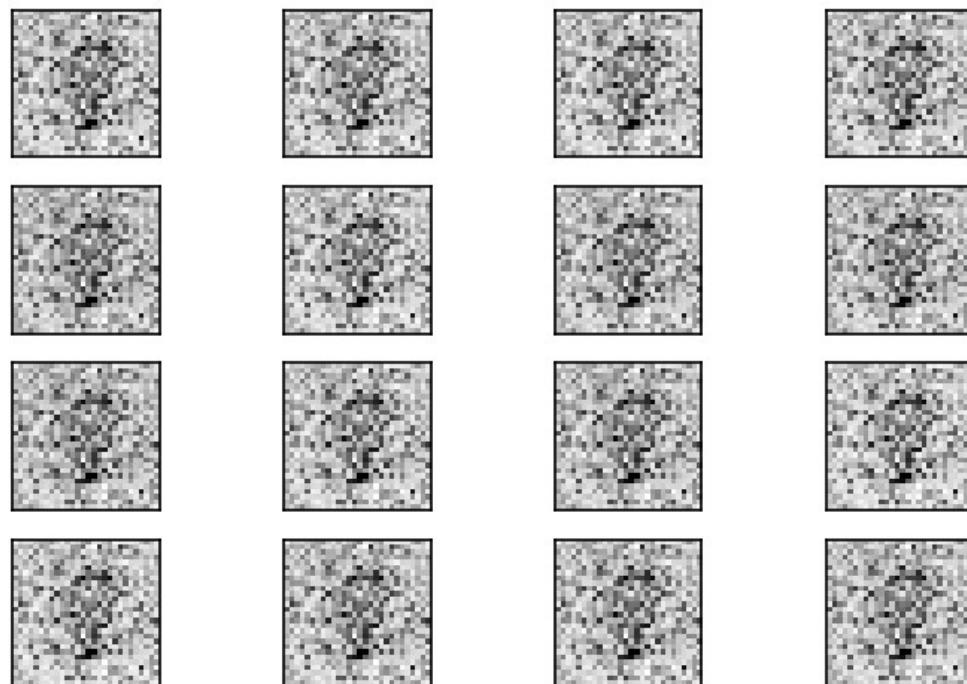
Richard Feynman

Can we use an indirect training then? Yes! There comes the GAN



GAN in action: Learning to generate handwritten digits

After 1 epoch(s)



GAN Samples

Objects:



Anyway,
GAN is meant for a lot more than that....

How GAN images evolved with time...!

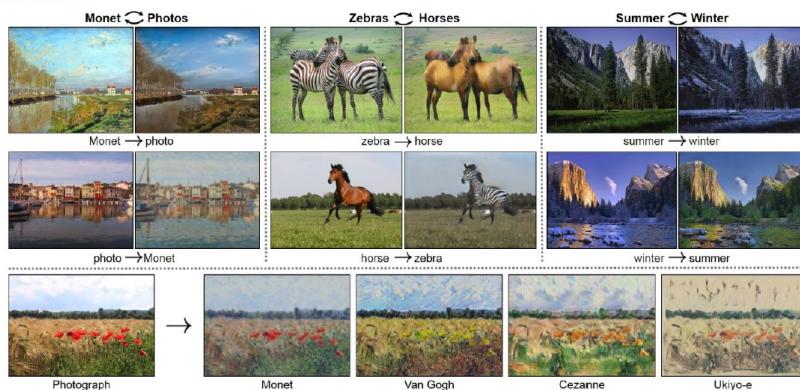


GANs can generate not only human faces. A recent [BigGAN](#) (by DeepMind) generates images from the classes of ImageNet (1000 classes) with better than before quality:



CycleGAN

Style transfer problem: change the style of an image while preserving the content.

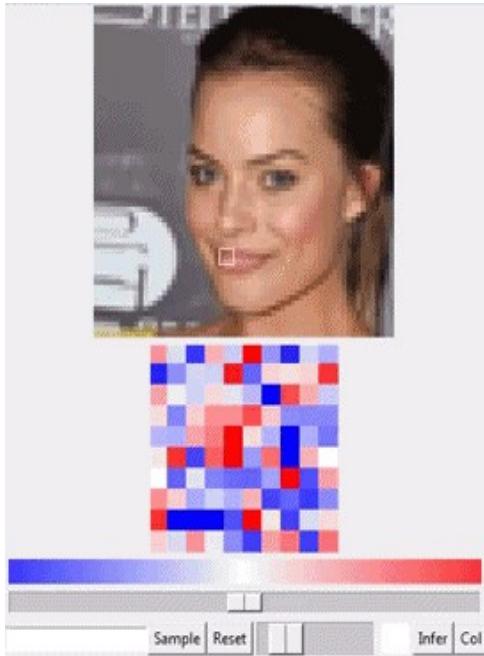


Data: Two unrelated collections of images, one for each style

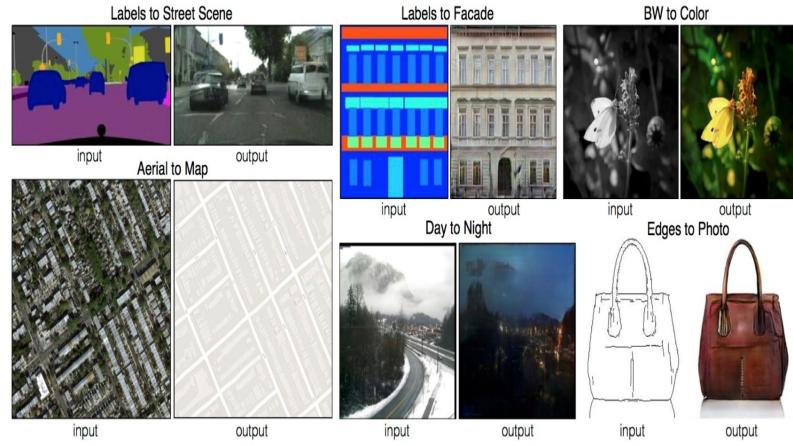


Future: More Cool Applications of Improved GAN Architectures...

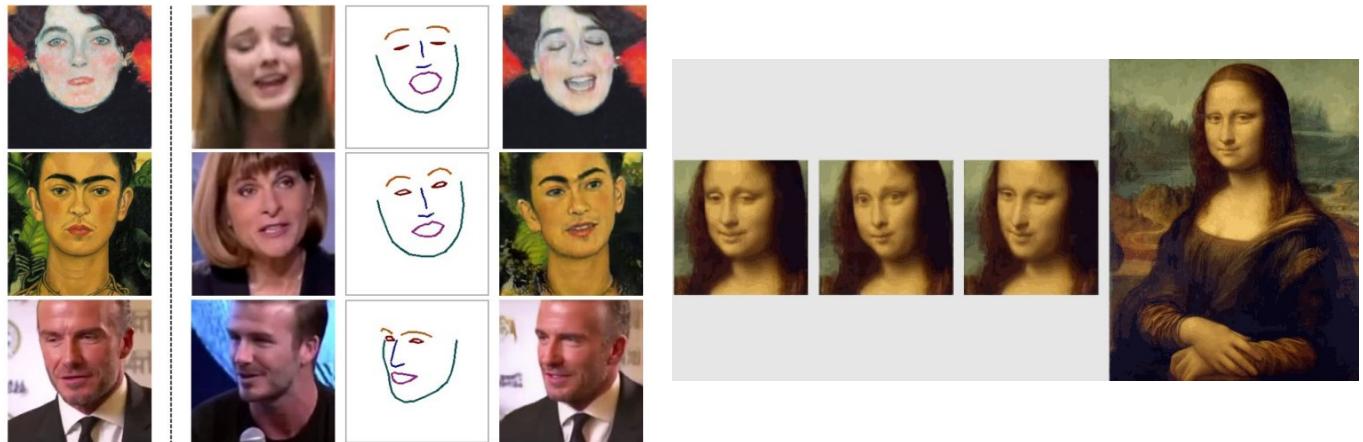
Neural Photo Editor: Content based image editing: for example, extend the hairband.



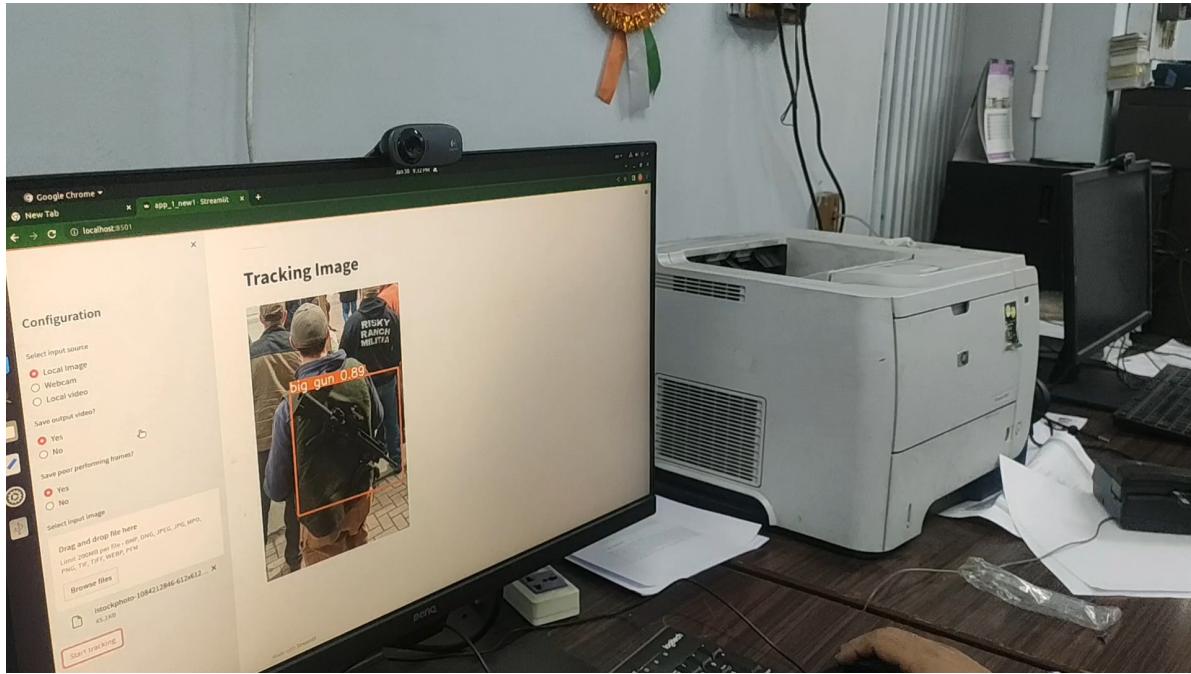
Pix2Pix is an image-to-image translation that get quoted in cross-domain GAN's paper frequently. For example, it converts a satellite image into a map (the bottom left).



Bringing Older Photos to Life:

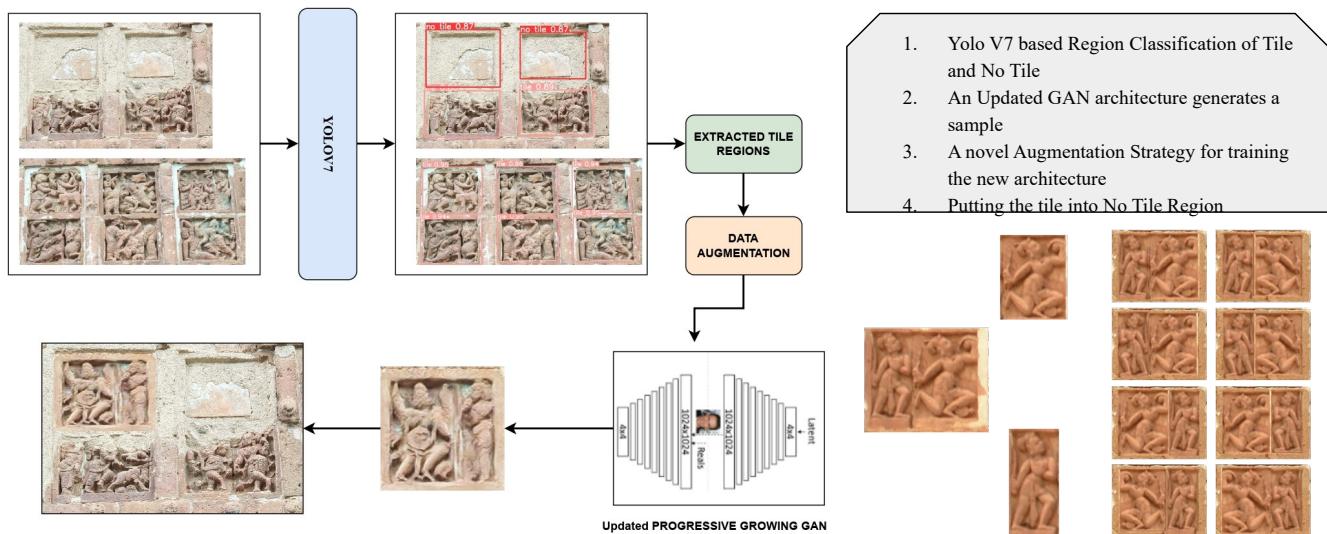


Weapon Detection System @ MLR Lab, ECSU, ISI



Self Supervised Tile Generation Approach (SSTG) to Digital Heritage Project, DST-SERB, ISI Kolkata

An automatic approach to **Detect** and **Fill** empty tile regions from a temple wall with **meaningful details**



SSTG: Results



Many other facets of machine learning are gaining momentum....

- Few Shot Learning
(ICLR 2019 Open Review: <https://openreview.net/forum?id=HkxLXnAcFQ>)
- Extreme Class Classification Problems
(NIPS 2017 Workshop: <http://manikvarma.org/events/XC17/index.html>)
- Privacy preserving learning
(NeurIPS 2018 Workshop on PPML: <https://ppml-workshop.github.io/ppml/>)
- Multi-label learning
(A nice survey in IEEE TKDE: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6471714>)
- Meta-learning and learning to learn
(see Sebastian Thrun's homepage: <http://robots.stanford.edu/papers/thrun.book3.html>)

[...and that's it!](#)

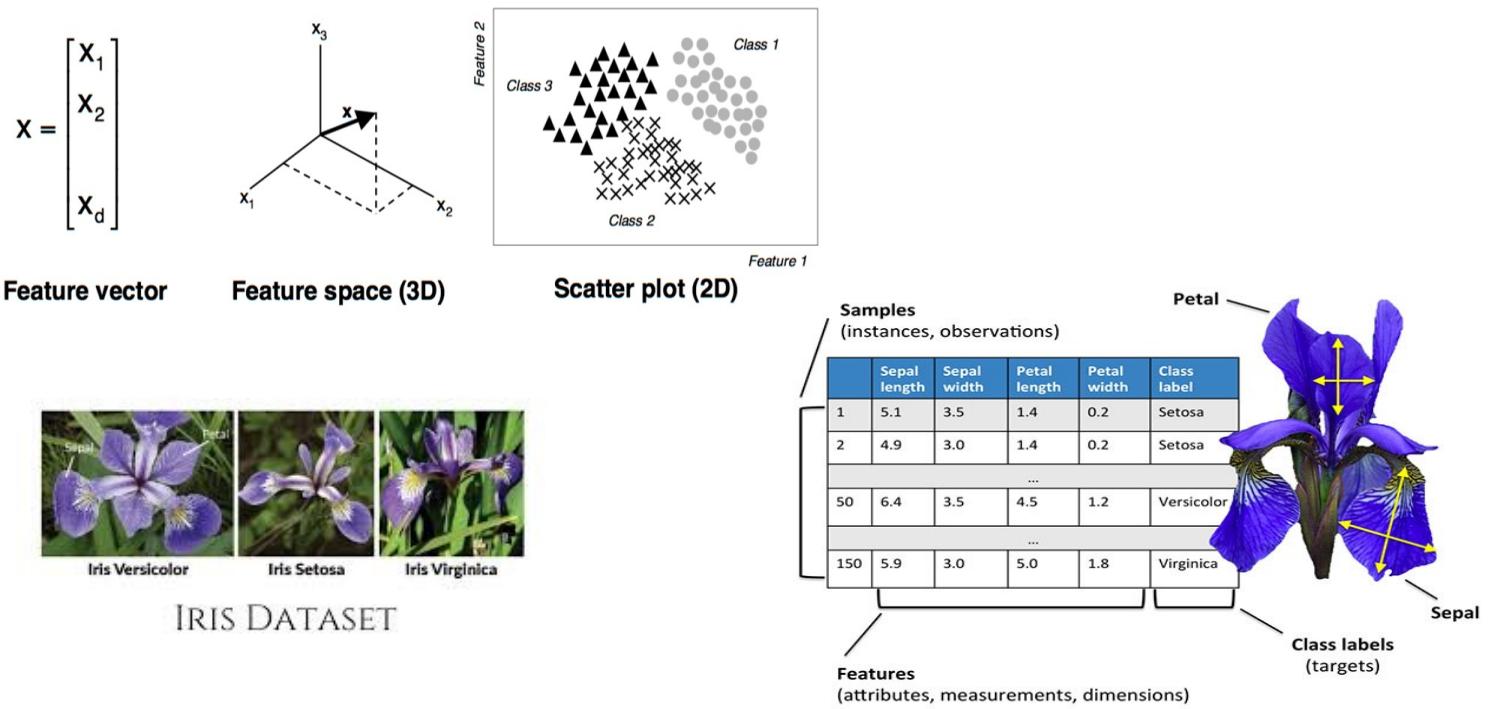
Thank You



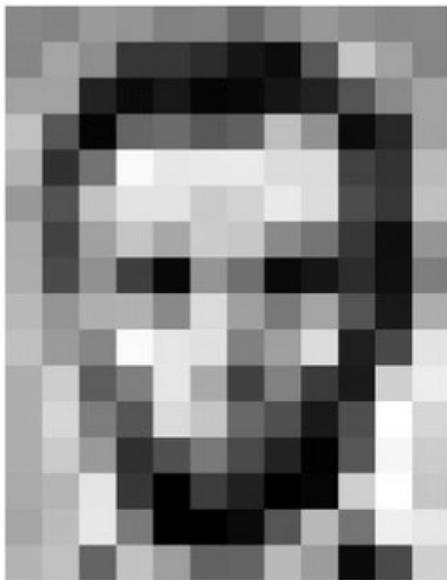
Linear Algebra: Review Lecture 1

M. Tech(CS) 2024- 25

“Data” as scalar, vector, matrix, and tensor?



“Data” as scalar, vector, matrix, and tensor?

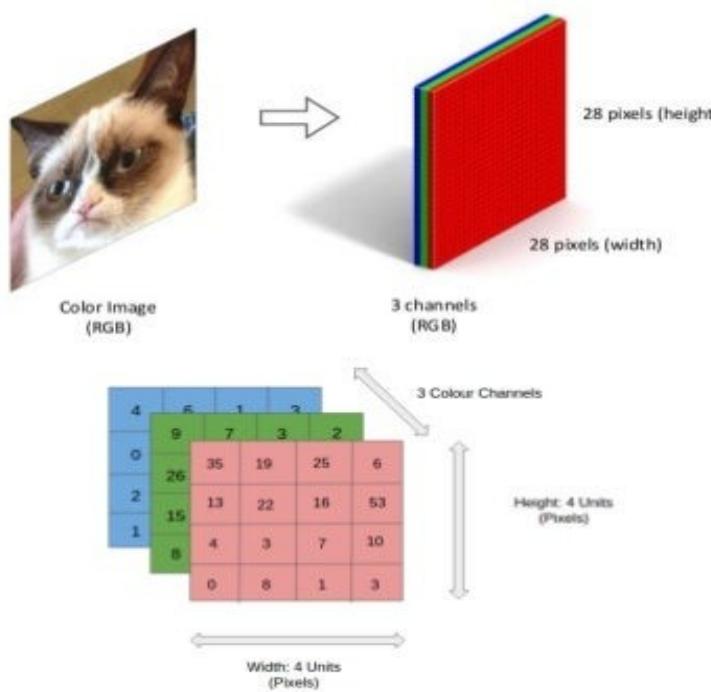


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	199	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	98	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	216

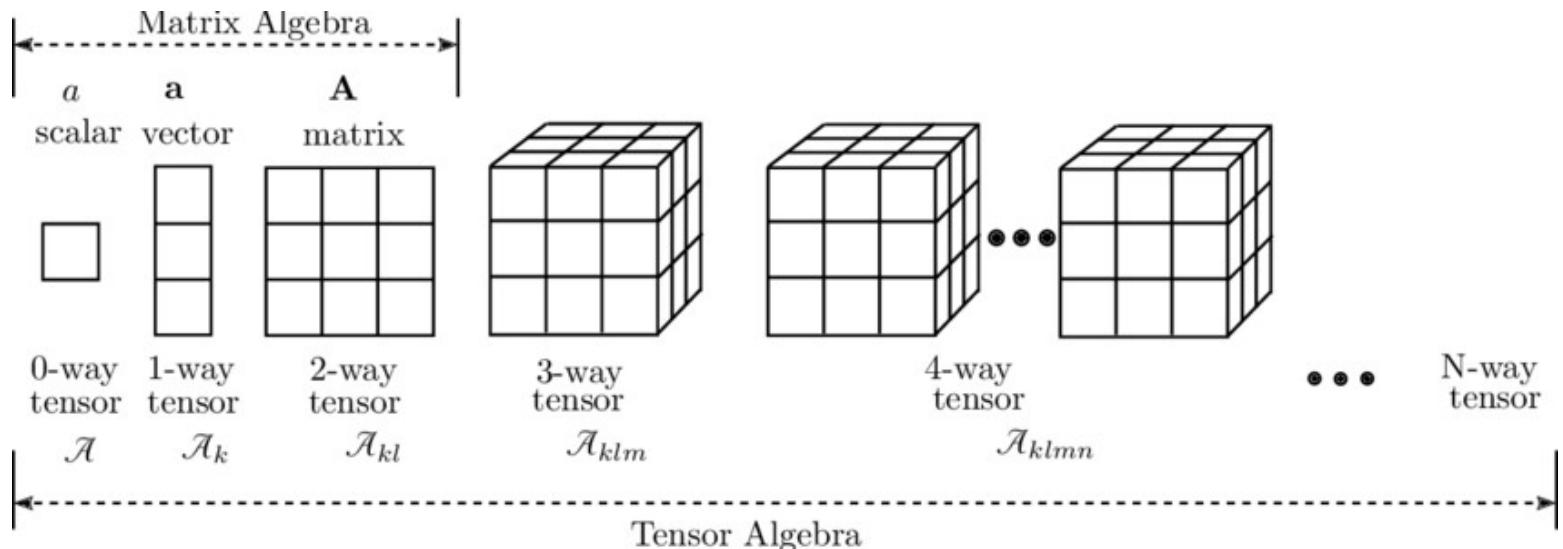
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	199	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	143	182	106
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	98	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	216

Source: Melvin Wevers, Thomas Smits, The visual digital turn: Using neural networks to study historical images, *Digital Scholarship in the Humanities*, Volume 35, Issue 1, April 2020

color image is 3rd-order tensor



“Data” as scalar, vector, matrix, and tensor?



Reference

- Matrix calculus, *Wiki* http://en.wikipedia.org/wiki/Matrix_calculus
- The Matrix Cookbook
http://www.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf
- Video lectures on Linear Algebra by **Prof. Gilbert Strang**, MIT
<https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/>
- Linear Algebra course by **Dr. Hung-yi Lee**, Nat. Taiwan Univ.

What are we going to learn?

System

- A system has input and output (function, transformation, operator)

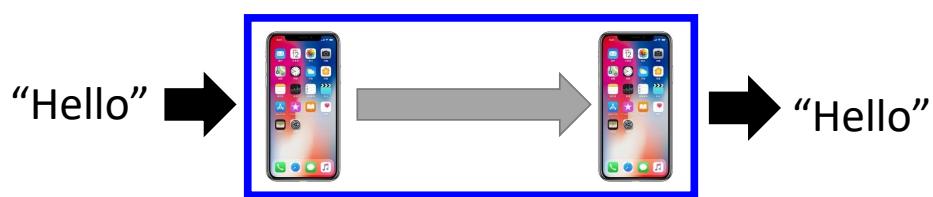
Speech Recognition System



Dialogue System (e.g. Siri, Alexa)

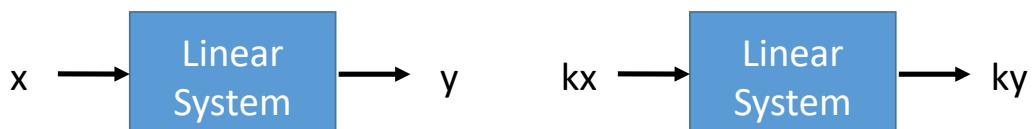


Communication System

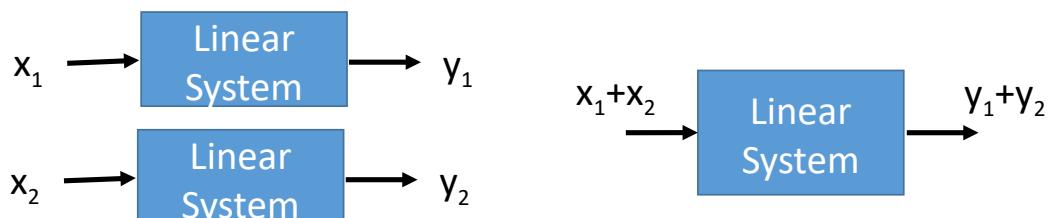


Linear System

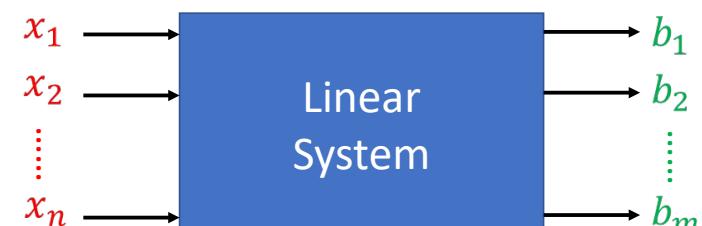
- Linear system have two properties
 - 1. Persevering Multiplication



- 2. Persevering Addition



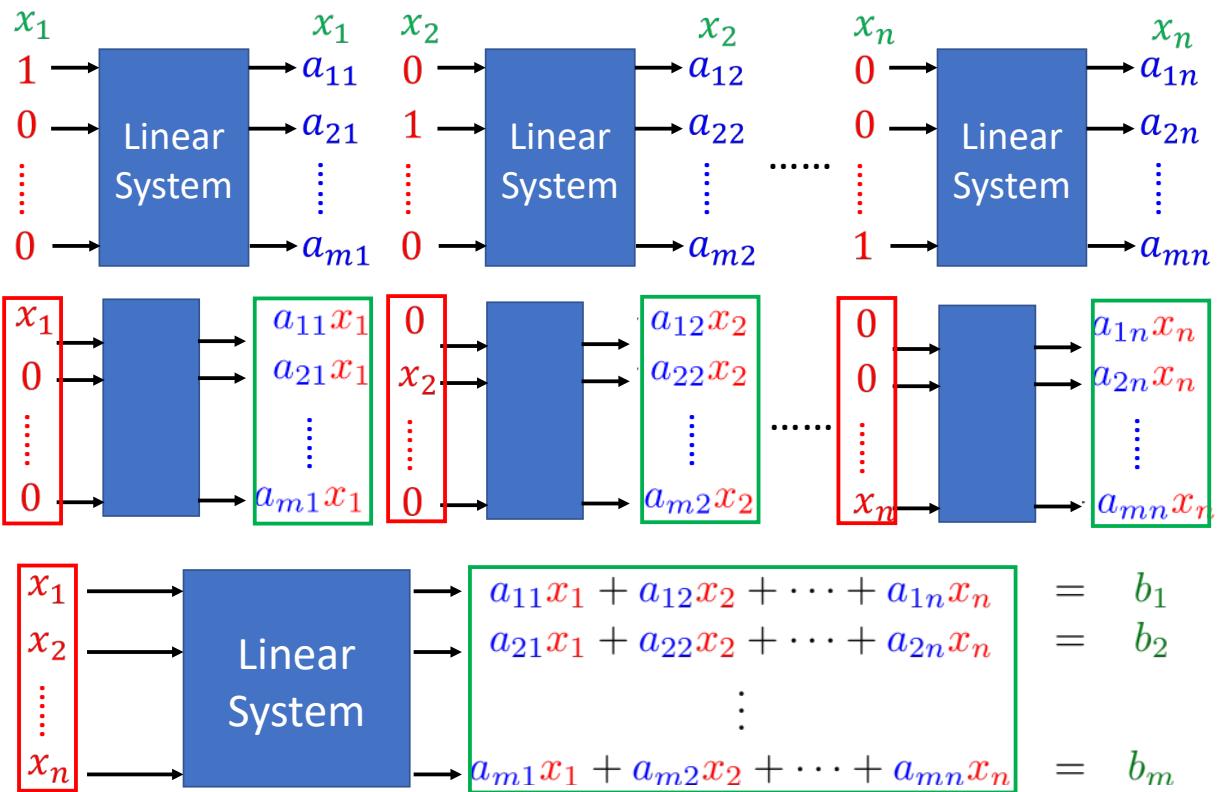
Linear System v.s. System of Linear Equations



? trivial

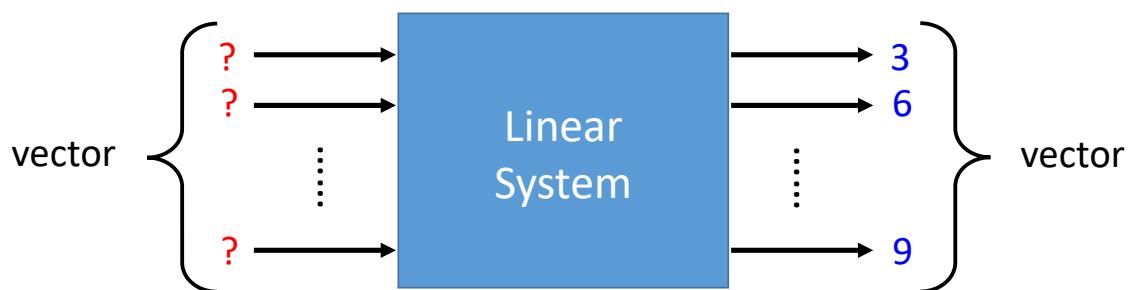
$$\begin{array}{lcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & = & b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & = & b_2 \end{array}$$

$$\begin{array}{c} \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = k b_m \end{array}$$



A linear system is described by a system of linear equations

What are we going to learn?



Does it have
solution?

Does it have
unique
solution?

How to find
the solution?

Determinants

Beyond 3 X 3

Different view from high school

Basic concepts

- **Vector** in \mathbb{R}^n is an ordered set of n real numbers.
 - e.g. $v = (1, 6, 3, 4)$ is in \mathbb{R}^4
 - A column vector: $\begin{pmatrix} 1 \\ 6 \\ 3 \\ 4 \end{pmatrix}$
 - A row vector: $(1 \ 6 \ 3 \ 4)$

- **m-by-n matrix** is an object in $\mathbb{R}^{m \times n}$ with m rows and n columns, each entry filled with a (typically) real number:

$$\begin{pmatrix} 1 & 2 & 8 \\ 4 & 78 & 6 \\ 9 & 3 & 2 \end{pmatrix}$$

Notation

- Matrix: $\mathbf{A}, \mathbf{X}, \mathbf{Y}$
 - bold capital letter
- Vector: $\mathbf{a}, \mathbf{x}, \mathbf{y}$ (**column**)
 - boldface lowercase letter
- Scalar: a, x, y
 - lowercase italicc typeface
- Transpose: $\mathbf{A}^T, \mathbf{a}^T$
- Trace: $\text{tr}(\mathbf{A})$
 - $\text{tr}(\mathbf{A}) = A_{11} + A_{22} + \dots + A_{nn} = \sum_{i=1}^n A_{ii}$
- Determinant: $\det(\mathbf{A})$

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \ddots & A_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}$$

m rows n columns
 $m \times n$ matrix
 m × 1 vector 1 × 1 scalar

Properties of Transpose

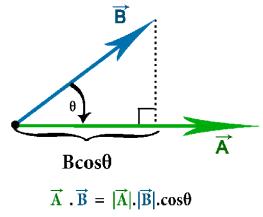
- $(\mathbf{A}^T)^T = \mathbf{A}$
- $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$
- $(\mathbf{A} + \mathbf{B} + \mathbf{C})^T = \mathbf{A}^T + \mathbf{B}^T + \mathbf{C}^T$
- $(r\mathbf{A})^T = r\mathbf{A}^T$
- $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$
- $(\mathbf{ABC})^T = \mathbf{C}^T \mathbf{B}^T \mathbf{A}^T$

Basic concepts

We will use lower case letters for vectors. The elements are referred by x_i .

- **Vector dot (inner) product:**

$$\mathbf{x}^T \mathbf{y} \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^n x_i y_i.$$



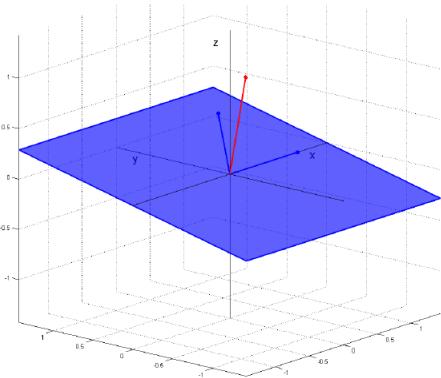
If $\mathbf{u} \cdot \mathbf{v} = 0$, $\|\mathbf{u}\|_2 \neq 0$, $\|\mathbf{v}\|_2 \neq 0 \rightarrow u \text{ and } v \text{ are orthogonal}$

If $\mathbf{u} \cdot \mathbf{v} = 0$, $\|\mathbf{u}\|_2 = 1$, $\|\mathbf{v}\|_2 = 1 \rightarrow u \text{ and } v \text{ are orthonormal}$

- **Vector outer product:**

$$\mathbf{x}\mathbf{y}^T \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}$$

Linear Subspaces



- Subspace $\mathcal{V} \subset \mathbb{R}^n$ satisfies

1. $0 \in \mathcal{V}$
2. If $x, y \in \mathcal{V}$ and $c \in \mathbb{R}$, then $c(x + y) \in \mathcal{V}$

- Vectors $\mathbf{x}_1, \dots, \mathbf{x}_m$ span \mathcal{V} if

$$\mathcal{V} = \left\{ \sum_{i=1}^m \alpha_i \mathbf{x}_i \mid \alpha \in \mathbb{R}^m \right\}$$

Linear Independence and Dimension

- Vectors $\mathbf{x}_1, \dots, \mathbf{x}_m$ are *linearly independent* if

$$\sum_{i=1}^m \alpha_i \mathbf{x}_i = 0 \Leftrightarrow \alpha = 0$$

– Every linear combination of the \mathbf{x}_i is unique

- $\text{Dim}(\mathcal{V}) = m$ if $\mathbf{x}_1, \dots, \mathbf{x}_m$ span \mathcal{V} and are linearly independent

– If $\mathbf{y}_1, \dots, \mathbf{y}_k$ span \mathcal{V} then

- $k \geq m$

- If $k > m$ then \mathbf{y}_i are NOT linearly independent

Matrix-Vector Products

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x \in \mathbb{R}^n$, their product is a vector $y = Ax \in \mathbb{R}^m$.

$$y = Ax = \begin{bmatrix} & & & \\ a_1 & a_2 & \cdots & a_n \\ & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} x_1 + \begin{bmatrix} a_2 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} x_2 + \dots + \begin{bmatrix} a_n \\ a_n \\ \vdots \\ a_n \end{bmatrix} x_n$$

If we write A by rows, then we can express Ax as,

$$y = Ax = \begin{bmatrix} \quad a_1^T \quad \quad \\ \quad a_2^T \quad \quad \\ \vdots \\ \quad a_m^T \quad \quad \end{bmatrix} x = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}.$$

Now can we **visualize** the column space of a matrix?

$$\text{eq.(a)} \quad \begin{bmatrix} 2 & 1 \\ 0 & 2 \\ 9 & 5 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$$

\mathbf{x} $\boldsymbol{\theta}$ \mathbf{y}

$$\text{eq.(b)} \quad \theta_1 \begin{bmatrix} 2 \\ 0 \\ 9 \end{bmatrix} + \theta_2 \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$$

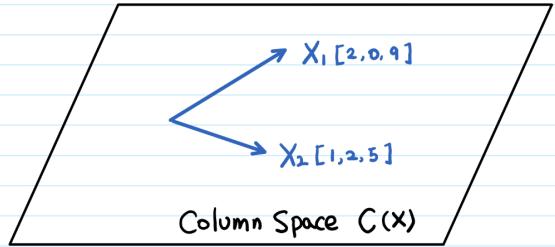
x_1 x_2 y

Vector $[2, 0, 9]$

→ pointing in some direction

Vector $[1, 2, 5]$

→ pointing in some other direction



Matrix-Matrix Products

$A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, $a_i \in \mathbb{R}^n$ and $b_j \in \mathbb{R}^n$

$$C = AB = \left[\begin{array}{ccc|c} & a_1^T & & \\ & a_2^T & & \\ \vdots & & & \\ & a_m^T & & \end{array} \right] \left[\begin{array}{cccc|c} & b_1 & b_2 & \cdots & b_p & \\ | & | & | & & | & \\ \end{array} \right] = \left[\begin{array}{cccc} a_1^T b_1 & a_1^T b_2 & \cdots & a_1^T b_p \\ a_2^T b_1 & a_2^T b_2 & \cdots & a_2^T b_p \\ \vdots & \vdots & \ddots & \vdots \\ a_m^T b_1 & a_m^T b_2 & \cdots & a_m^T b_p \end{array} \right]$$

A much trickier representation:

$$C = AB = \left[\begin{array}{cccc|c} & b_1^T & & & & \\ | & | & & & & \\ a_1 & a_2 & \cdots & a_n & & \\ | & | & & | & & \\ \end{array} \right] \left[\begin{array}{ccc|c} & b_1^T & & \\ & b_2^T & & \\ \vdots & & & \\ & b_n^T & & \end{array} \right] = \sum_{i=1}^n a_i b_i^T$$

Viewing Matrix-Matrix Product in terms of Vector Matrix Product:

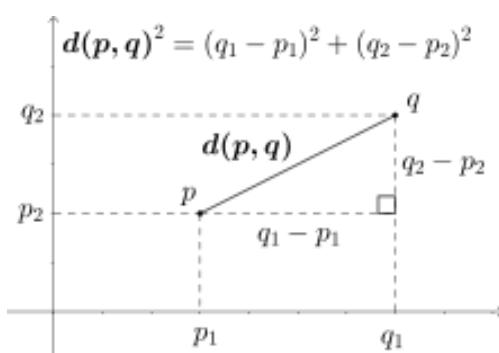
$$C = AB = A \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ab_1 & Ab_2 & \cdots & Ab_p \\ | & | & & | \end{bmatrix}$$

And

$$C = AB = \begin{bmatrix} \cdots & a_1^T & \cdots \\ \cdots & a_2^T & \cdots \\ \vdots & & \vdots \\ \cdots & a_m^T & \cdots \end{bmatrix} B = \begin{bmatrix} \cdots & a_1^T B & \cdots \\ \cdots & a_2^T B & \cdots \\ \vdots & & \vdots \\ \cdots & a_m^T B & \cdots \end{bmatrix}$$

Basic concepts

Remember the good, old Euclidean distance?



- Quantify “size” of a vector
- Given $x \in \mathbb{R}^n$, a norm satisfies
 1. $\|cx\| = |c|\|x\|$
 2. $\|x\| = 0 \Leftrightarrow x = 0$
 3. $\|x + y\| \leq \|x\| + \|y\|$
- Common norms:
 1. Euclidean L_2 -norm: $\|x\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$
 2. L_1 -norm: $\|x\|_1 = |x_1| + \cdots + |x_n|$
 3. L_∞ -norm: $\|x\|_\infty = \max_i |x_i|$

Norms can also be defined for matrices, such as the Frobenius norm,

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2} = \sqrt{\text{tr}(A^T A)}.$$

Special matrices

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \text{ diagonal} \quad \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \text{ upper-triangular}$$

$$\begin{pmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{pmatrix} \text{ tri-diagonal} \quad \begin{pmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{pmatrix} \text{ lower-triangular}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ I (identity matrix)}$$

Useful Matrices

- Identity matrix $I \in \mathbb{R}^{m \times m}$

$$- AI = A, IA = A$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad I_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

- Diagonal matrix $A \in \mathbb{R}^{m \times m}$

$$A = \text{diag}(a_1, \dots, a_m) = \begin{bmatrix} a_1 & \cdots & 0 \\ \vdots & a_i & \vdots \\ 0 & \cdots & a_m \end{bmatrix}$$

Useful Matrices

- Symmetric $A \in \mathbb{R}^{m \times m}$: $A = A^T$

- Orthogonal $U \in \mathbb{R}^{m \times m}$:

$$U^T U = U U^T = I$$

– Columns/ rows are orthonormal

- Positive semidefinite $A \in \mathbb{R}^{m \times m}$.

$$x^T A x \geq 0 \quad \text{for all } x \in \mathbb{R}^m$$

– Equivalently, there exists $L \in \mathbb{R}^{m \times m}$

$$A = L L^T$$

Matrix Subspaces

- Matrix $M \in \mathbb{R}^{m \times n}$ defines two subspaces
 - Column space $\text{col}(M) = \{M\alpha | \alpha \in \mathbb{R}^n\} \subset \mathbb{R}^m$
 - Row space $\text{row}(M) = \{M^T \beta | \beta \in \mathbb{R}^m\} \subset \mathbb{R}^n$
- Nullspace of M : $\text{null}(M) = \{x \in \mathbb{R}^n | Mx = 0\}$
 - $\text{null}(M) \perp \text{row}(M)$
 - $\dim(\text{null}(M)) + \dim(\text{row}(M)) = n$
 - Analog for column space

Matrix Inverse

- $M \in \mathbb{R}^{m \times m}$ is invertible iff $\text{rank}(M) = m$
- Inverse is unique and satisfies
 1. $M^{-1}M = MM^{-1} = I$
 2. $(M^{-1})^{-1} = M$
 3. $(M^T)^{-1} = (M^{-1})^T$
 4. If A is invertible then MA is invertible and $(MA)^{-1} = A^{-1}M^{-1}$

Systems of Equations

- Given $M \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^m$ wish to solve $Mx = y$
 - Exists only if $y \in \text{col}(M)$
 - Possibly infinite number of solutions
 - If M is invertible then $x = M^{-1}y$
 - Notational device, do not actually invert matrices
 - Computationally, use solving routines like Gaussian elimination

Determinant of a Matrix

- Used for inversion
- If $\det(A) = 0$, then A has no inverse

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \det(A) = ad - bc$$

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

<http://www.euclideanspace.com/maths/algebra/matrix/functions/inverse/threeD/index.htm>

Can we find a matrix to multiply the first matrix by to get the identity?

$$\begin{bmatrix} -3 & -1 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} -1 & -\frac{1}{2} \\ 2 & \frac{3}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -\frac{1}{2} \\ 2 & \frac{3}{2} \end{bmatrix} \begin{bmatrix} -3 & -1 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Let A be an $n \times n$ matrix. If there exists a matrix B such that $AB = BA = I$ then we call this matrix the **inverse** of A and denote it A^{-1} .

If A has an inverse we say that A is **nonsingular**.
If A^{-1} does not exist we say A is **singular**.

To find the inverse of a matrix we put the matrix A , a line and then the **identity matrix**. We then perform row operations on matrix A to turn it into the identity. We carry the row operations across and the **right hand side will turn into the inverse**.

$$\left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ -2 & -7 & 0 & 1 \end{array} \right]$$

$$A = \begin{bmatrix} 1 & 3 \\ -2 & -7 \end{bmatrix}$$

$$-r_2 \quad \left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 0 & 1 & -2 & -1 \end{array} \right]$$

$$2r_1 + r_2 \quad \left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 0 & -1 & 2 & 1 \end{array} \right]$$

$$r_1 - r_2 \quad \left[\begin{array}{cc|cc} 1 & 0 & 7 & 3 \\ 0 & 1 & -2 & -1 \end{array} \right]$$

$$A = \begin{bmatrix} 1 & 3 \\ -2 & -7 \end{bmatrix} \quad A^{-1} = \left[\begin{array}{c|c} 7 & 3 \\ -2 & -1 \end{array} \right]$$

Check this answer by multiplying. We should get the identity matrix if we've found the inverse.

$$AA^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

We can use A^{-1} to solve a system of equations

$$x + 3y = 1$$

$$2x + 5y = 3$$

To see how, we can re-write a system of equations as matrices.

$$AX = b$$

coefficient matrix

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix}$$

variable matrix

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

constant matrix

$$= \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$$A\mathbf{x} = \mathbf{b}$$

left multiply both sides by the inverse of A

$$A^{-1}A\mathbf{x} = A^{-1}\mathbf{b}$$

This is just the identity

$$I\mathbf{x} = A^{-1}\mathbf{b}$$

This then gives us a formula for finding the variable matrix: Multiply A inverse by the constants.

but the identity times a matrix just gives us back the matrix so we have:

$$\mathbf{x} = A^{-1}\mathbf{b}$$

$$x + 3y = 1$$

$$2x + 5y = 3$$

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \quad \text{find the inverse}$$

$$\left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 2 & 5 & 0 & 1 \end{array} \right]$$

$$\begin{array}{l} -2r_1+r_2 \\ \hline \left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 0 & -1 & -2 & 1 \end{array} \right] \end{array}$$

$$\begin{array}{l} -r_2 \\ \hline \left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 0 & 1 & 2 & -1 \end{array} \right] \end{array}$$

$$\begin{array}{l} r_1-3r_2 \\ \hline \left[\begin{array}{cc|cc} 1 & 0 & -5 & 3 \\ 0 & 1 & 2 & -1 \end{array} \right] \end{array}$$

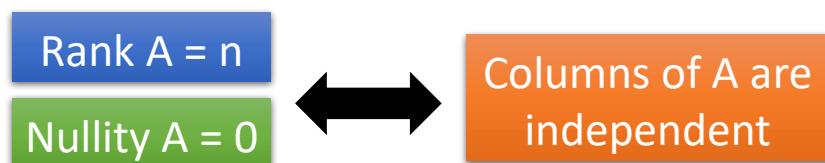
$$A^{-1}\mathbf{b} = \begin{bmatrix} -5 & 3 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}$$

This is the answer to the system

Rank and Nullity

- The **rank** of a matrix is defined as the maximum number of *linearly independent columns* in the matrix.
-
- **Nullity** = Number of columns - **rank**

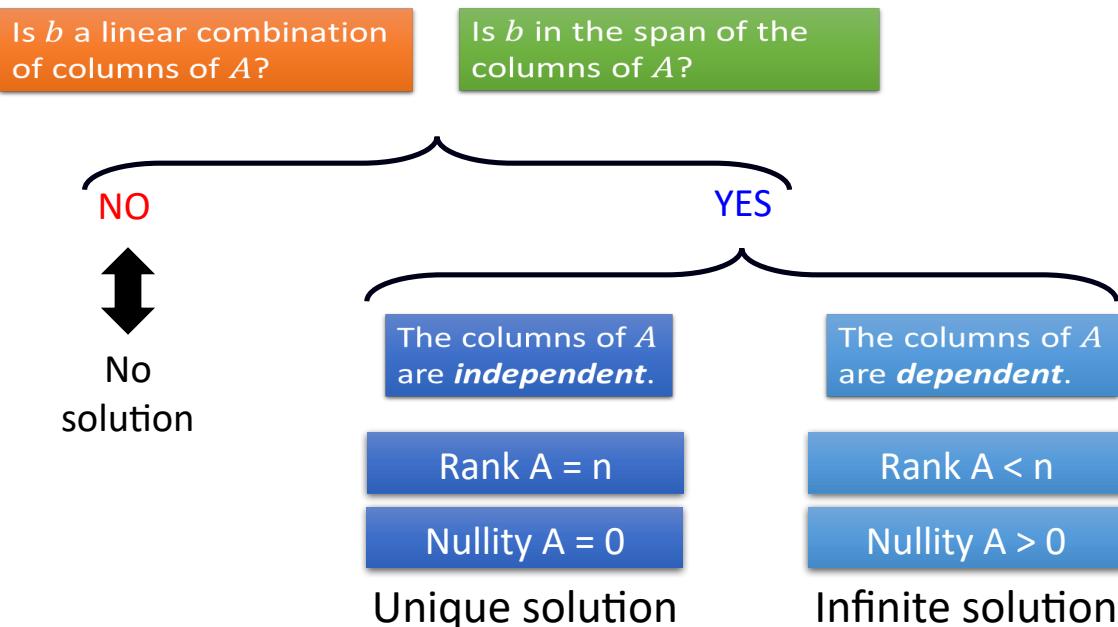
If A is a $m \times n$ matrix:



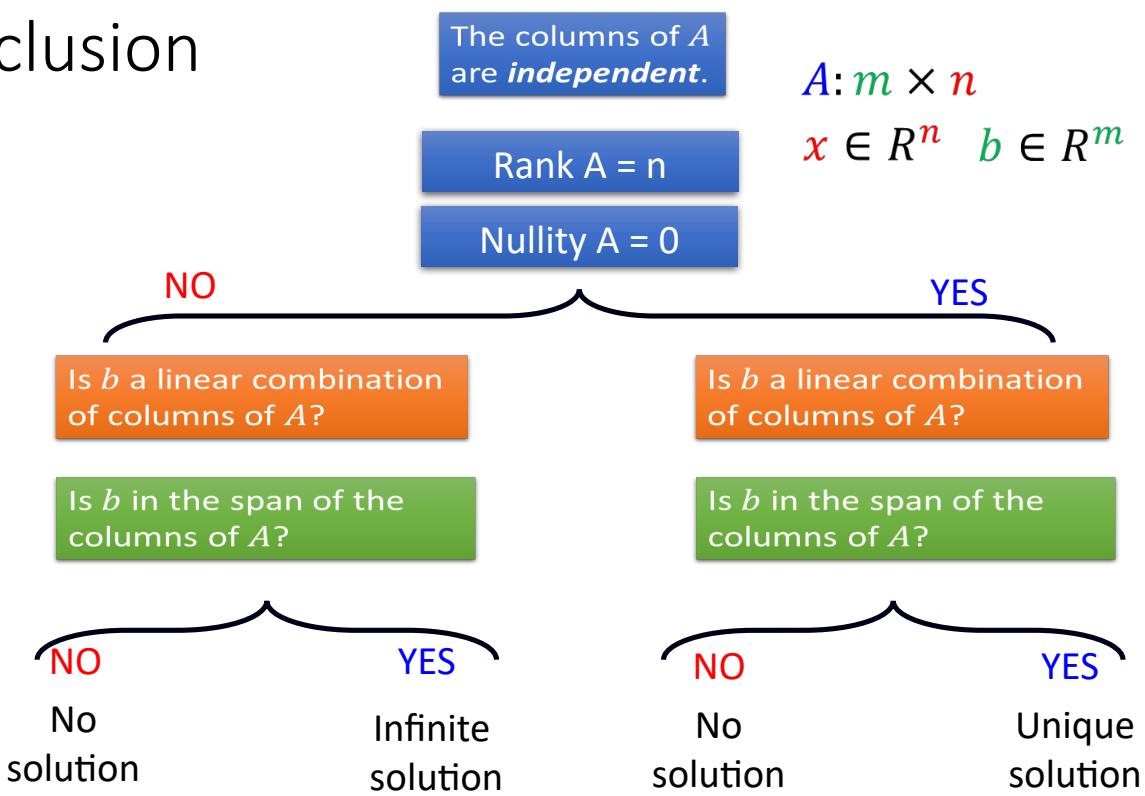
Conclusion

$$A\mathbf{x} = \mathbf{b}$$

$$A: m \times n \quad \mathbf{x} \in R^n \quad \mathbf{b} \in R^m$$



Conclusion



Visualizing System of Linear Equations...(every eqn. is a line, plane, or hyperplane)

Figure 1.9
All Three Equations

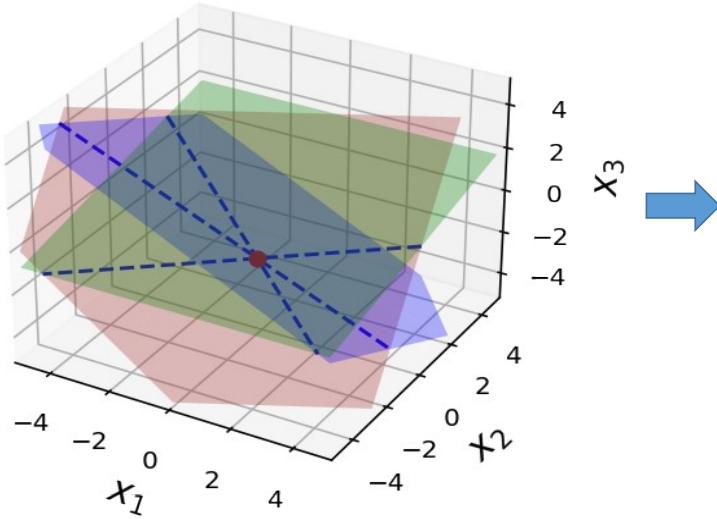
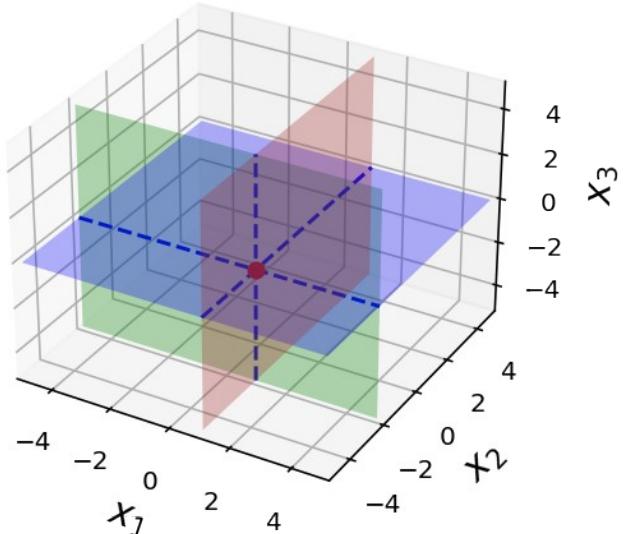


Figure 1.10
The Equivalent System



When you will obtain no unique solution....

Figure 1.2
Example 2

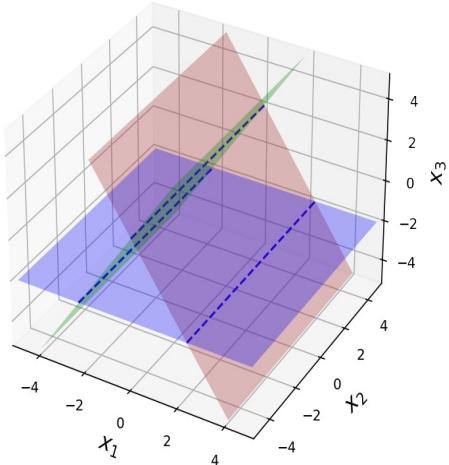


Figure 1.3
Example 3

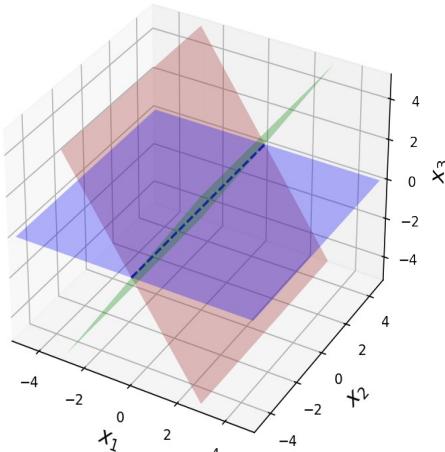
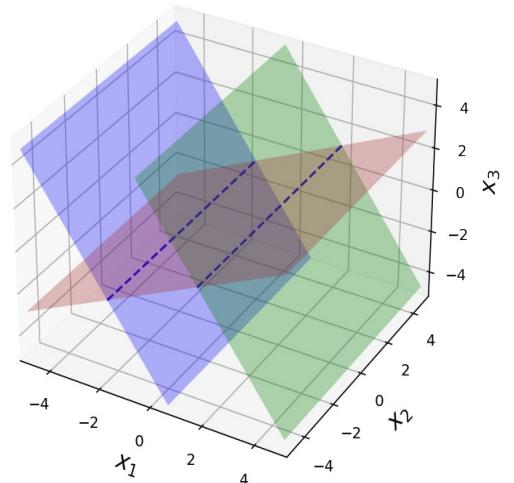


Figure 1.5
Example 5



Source: *Linear Algebra, Geometry, and Computation*, Mark Crovella

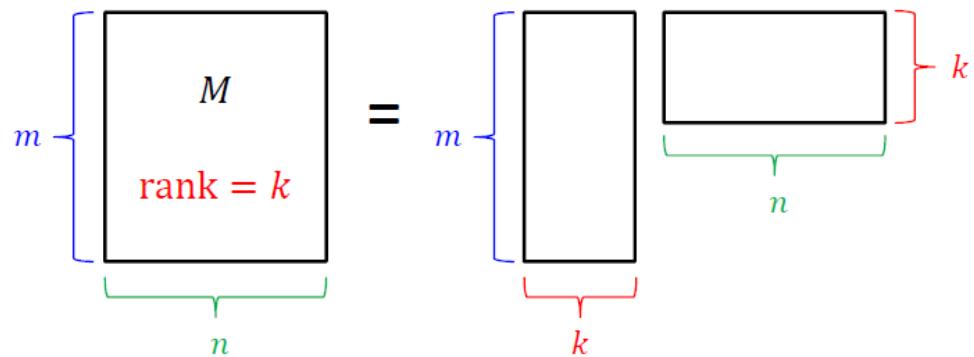
Systems of Equations

- What if $y \notin \text{col}(M)$?
- Find x that gives $\hat{y} = Mx$ closest to y
 - \hat{y} is projection of y onto $\text{col}(M)$
 - Also known as regression
- Assume $\text{rank}(M) = n < m$

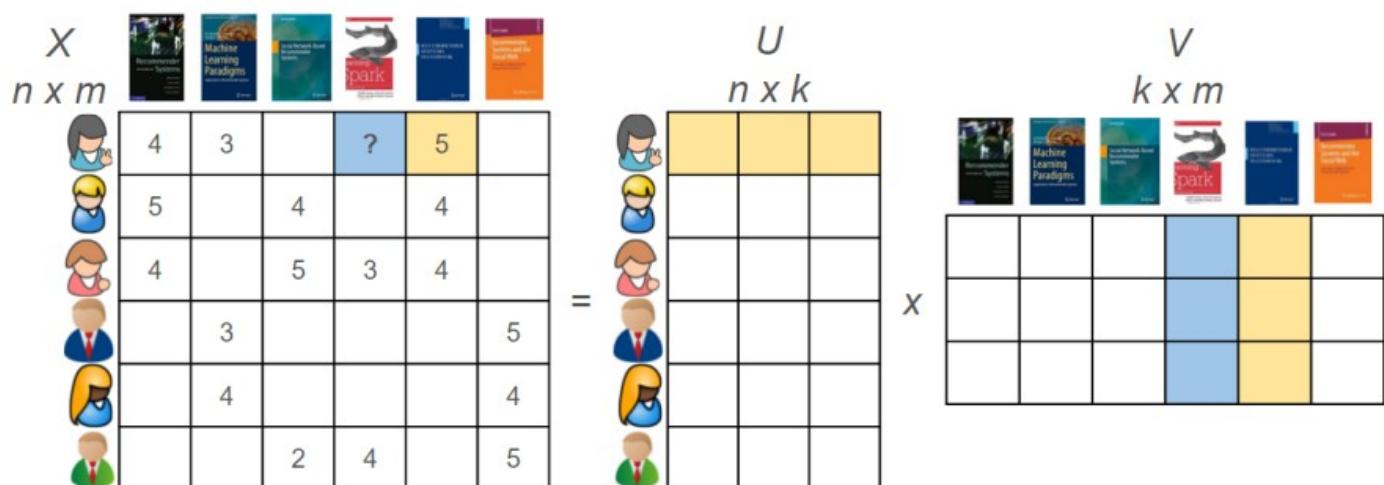
$$x = \underbrace{(M^T M)^{-1} M^T y}_{\text{Invertible}} \quad \hat{y} = \underbrace{M(M^T M)^{-1} M^T y}_{\text{Projection matrix}}$$

Matrix Rank

- $\text{rank}(M)$ gives dimensionality of row and column spaces
- If $M \in \mathbb{R}^{m \times n}$ has rank k , can decompose into product of $m \times k$ and $k \times n$ matrices



Matrix factorization: a key method for designing recommender systems



Eigenvalues and Eigenvectors

- Eigenvalue problem (one of the most important problems in the linear algebra):

If A is an $n \times n$ matrix, do there exist nonzero vectors \mathbf{x} in R^n such that $A\mathbf{x}$ is a scalar multiple of \mathbf{x} ?

(The term eigenvalue is from the German word *Eigenwert*, meaning “proper value”)

- Eigenvalue and Eigenvector :

A : an $n \times n$ matrix

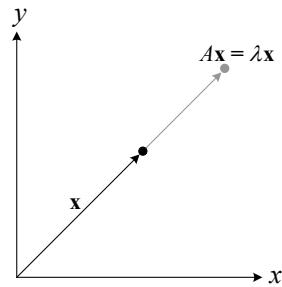
λ : a scalar (could be zero)

\mathbf{x} : a nonzero vector in R^n

$$A\mathbf{x} = \lambda\mathbf{x}$$

Eigenvalue
↓
↑
Eigenvector

※ Geometric Interpretation



Eigenvalues and Eigenvectors

- If $A\mathbf{v} = \lambda\mathbf{v}$ (\mathbf{v} is a vector, λ is a scalar)
 - \mathbf{v} is an eigenvector of A **excluding zero vector**
 - λ is an eigenvalue of A that corresponds to \mathbf{v}

A must be square

$$\begin{bmatrix} 5 & 2 & 1 \\ -2 & 1 & -1 \\ 2 & 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ -4 \\ 4 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Eigen value

Eigen vector

Looking for Eigenvalues

- Example 1: Find the eigenvalues of

$$A = \begin{bmatrix} -4 & -3 \\ 3 & 6 \end{bmatrix}$$

A scalar t is an eigenvalue of $A \iff \det(A - tI_n) = 0$

$$A - tI_2 = \begin{bmatrix} -4 - t & -3 \\ 3 & 6 - t \end{bmatrix}$$

$$\det(A - tI_2) = 0$$

$$\rightarrow t = -3 \text{ or } 5$$

The eigenvalues of A are -3 or 5.

Looking for Eigenvalues

- Example 1: Find the eigenvalues of

$$A = \begin{bmatrix} -4 & -3 \\ 3 & 6 \end{bmatrix}$$

The eigenvalues of A are -3 or 5.

Eigenspace of -3

$$Ax = -3x \rightarrow (A + 3I)x = 0$$

find the solution

Eigenspace of 5

$$Ax = 5x \rightarrow (A - 5I)x = 0$$

find the solution

Trace & Determinant

- If A is a square n -by- n matrix and if $\lambda_1, \dots, \lambda_n$ are the eigenvalues of A , then
 - $\text{tr}(A) = A_{11} + A_{22} + \dots + A_{nn} = \sum_{i=1}^n A_{ii} = \sum_{i=1}^n \lambda_i$
 - $\det(A) = \sum_{i=1}^n (-1)^{i+j} a_{i,j} M_{i,j} = \prod_{i=1}^n \lambda_i$
- Minor $M_{i,j}$:** the determinant of the $(n - 1) \times (n - 1)$ -matrix that results from A by removing the i th row and the j th column.
- Cofactor $C_{i,j}$:** $(-1)^{i+j} M_{i,j}$
- Cofactor matrix:** $C = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & & \ddots & \vdots \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix}$

$$A^{-1} = \frac{(C)^T}{\det(A)}$$

The Eigen-decomposition

$$A = \underbrace{\begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}}_{\text{square matrix}} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}^{-1}$$

the inverse exists only if eigenvectors are linearly independent

- Eigenvalue decomposition of symmetric $M \in \mathbb{R}^{m \times m}$ is

$$M = Q\Sigma Q^T = \sum_{i=1}^m \lambda_i \mathbf{q}_i \mathbf{q}_i^T$$

- $\Sigma = \text{diag}(\lambda_1, \dots, \lambda_n)$ contains eigenvalues of M
- Q is orthogonal and contains eigenvectors \mathbf{q}_i of M
- If M is not symmetric but *diagonalizable*

$$M = Q\Sigma Q^{-1}$$

- Σ is diagonal by possibly complex
- Q not necessarily orthogonal

~~Matrix calculus is a specialized notation for doing multivariable calculus, especially over spaces of matrices.~~

Let \mathbf{x} and \mathbf{y} be vectors of orders n and m respectively:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix},$$

where each component y_i may be a function of all the x_j , a fact represented by saying that \mathbf{y} is a function of \mathbf{x} , or

$$\mathbf{y} = \mathbf{y}(\mathbf{x}).$$

If $n = 1$, \mathbf{x} reduces to a scalar, which we call x . If $m = 1$, \mathbf{y} reduces to a scalar, which we call y . Various applications are studied in the following subsections.

Slides from: Fin500J Mathematical Foundations in Finance, Philip H. Dybvig

49

1.1 Derivative of Vector with Respect to Vector

The derivative of the vector \mathbf{y} with respect to vector \mathbf{x} is the $n \times m$ matrix

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

This is the tangent matrix or often referred as the Jacobian matrix.

1.2 Derivative of a Scalar with Respect to Vector

If y is a scalar

$$\frac{\partial y}{\partial \mathbf{x}} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}.$$

It is also called the gradient of y with respect to a vector variable x , denoted by

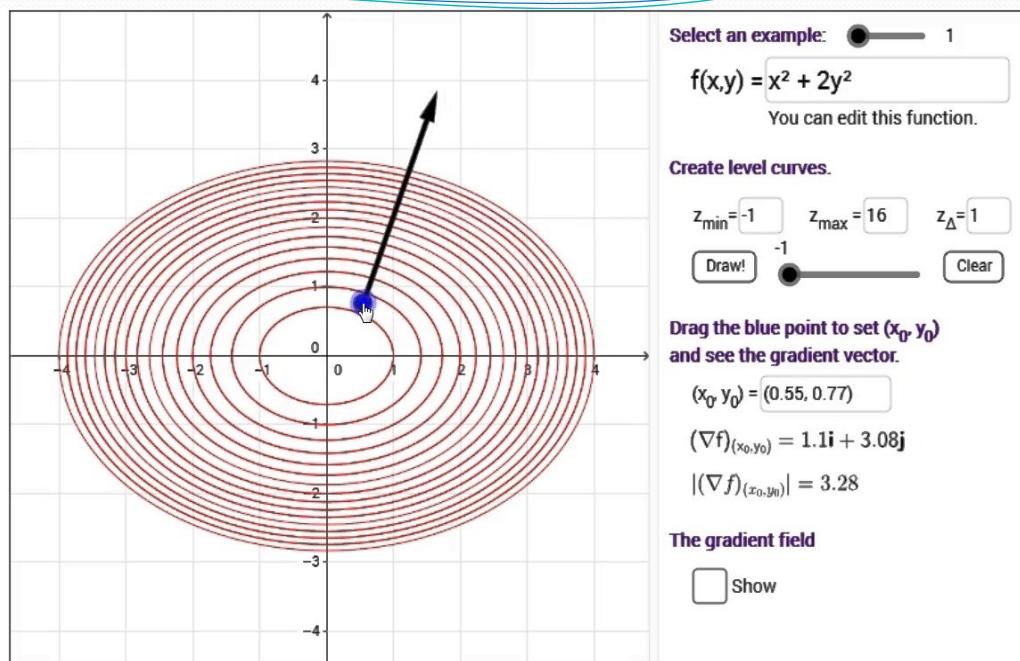
$$\nabla y$$

1.3 Derivative of Vector with Respect to Scalar

$$\frac{\partial \mathbf{y}}{\partial x} \stackrel{\text{def}}{=} \left[\frac{\partial y_1}{\partial x} \quad \frac{\partial y_2}{\partial x} \quad \cdots \quad \frac{\partial y_m}{\partial x} \right]$$

51

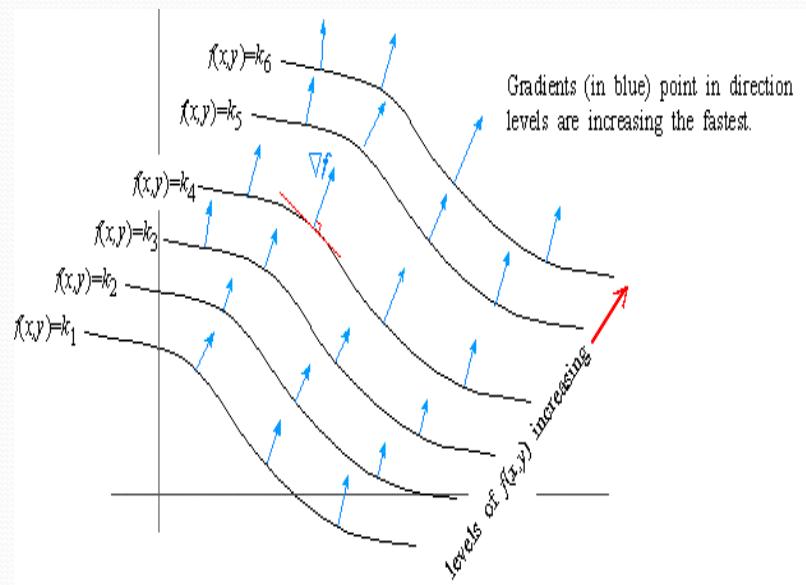
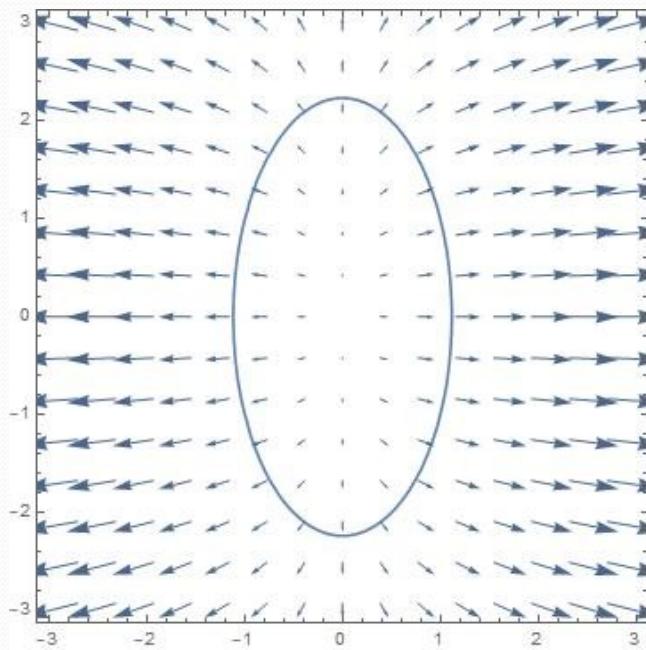
Can we “see” the gradients?



Nice Youtube video is here:

<https://www.youtube.com/watch?v=W6aDzrrLAzQ>

52



53

Example 1

Given $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

and $y_1 = x_1^2 - x_2$
 $y_2 = x_3^2 + 3x_2$

the partial derivative matrix $\partial \mathbf{y} / \partial \mathbf{x}$ is computed as follows:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 & 0 \\ -1 & 3 \\ 0 & 2x_3 \end{bmatrix}$$

54

Some useful vector derivative formulas

$$\frac{\partial \mathbf{C}\mathbf{x}}{\partial \mathbf{x}} = \mathbf{C}^T$$

$$\begin{cases} \frac{\partial \mathbf{x}^T \mathbf{C}}{\partial \mathbf{x}} = \mathbf{C} \\ \frac{\partial \mathbf{x}^T \mathbf{x}}{\partial \mathbf{x}} = 2\mathbf{x} \end{cases}$$

Homework

$$\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{t=1}^n x_t C_{1t} \\ \sum_{t=1}^n x_t C_{2t} \\ \vdots \\ \sum_{t=1}^n x_t C_{nt} \end{bmatrix}$$

$$\frac{\partial \mathbf{C}\mathbf{x}}{\partial \mathbf{x}} = \begin{pmatrix} c_{11} & c_{21} & \cdots & c_{n1} \\ c_{12} & c_{22} & \cdots & c_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{pmatrix} = \mathbf{C}^T$$

55

Important Property of Quadratic Form $\mathbf{x}^T \mathbf{C} \mathbf{x}$

$$\frac{\partial (\mathbf{x}^T \mathbf{C} \mathbf{x})}{\partial \mathbf{x}} = (\mathbf{C} + \mathbf{C}^T) \mathbf{x}$$

Proof:

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = \sum_{i=1}^n \left[x_i \sum_{j=1}^n (x_j C_{ij}) \right]$$

$$\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{t=1}^n x_t C_{1t} \\ \sum_{t=1}^n x_t C_{2t} \\ \vdots \\ \sum_{t=1}^n x_t C_{nt} \end{bmatrix}$$

$$\begin{aligned} \Rightarrow \frac{\partial (\mathbf{x}^T \mathbf{C} \mathbf{x})}{\partial x_k} &= \frac{\partial \left\{ \sum_{i=1}^n \left[x_i \sum_{j=1}^n (x_j C_{ij}) \right] \right\}}{\partial x_k} = \frac{\partial \left\{ x_k \sum_{j=1}^n (x_j C_{kj}) \right\}}{\partial x_k} + \frac{\partial \left\{ \sum_{i=1}^n [x_i x_k C_{ik}] \right\}}{\partial x_k} \\ &= \sum_{j=1}^n x_j C_{kj} + \sum_{i=1}^n x_i C_{ik} \end{aligned}$$

$$\Rightarrow \frac{\partial (\mathbf{x}^T \mathbf{C} \mathbf{x})}{\partial \mathbf{x}} = \mathbf{C} \mathbf{x} + \mathbf{C}^T \mathbf{x} = (\mathbf{C} + \mathbf{C}^T) \mathbf{x}$$

If \mathbf{C} is symmetric, $\frac{\partial (\mathbf{x}^T \mathbf{C} \mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{C} \mathbf{x}$

56

2 The Chain Rule for Vector Functions

Let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} \quad \text{and} \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

where \mathbf{z} is a function of \mathbf{y} , which is in turn a function of \mathbf{x} , we can write

$$\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right)^T = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \cdots & \frac{\partial z_1}{\partial x_n} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial z_m}{\partial x_1} & \frac{\partial z_m}{\partial x_2} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix}$$

Each entry of this matrix may be expanded as

$$\frac{\partial z_i}{\partial x_j} = \sum_{q=1}^r \frac{\partial z_i}{\partial y_q} \frac{\partial y_q}{\partial x_j} \quad \begin{cases} i = 1, 2, \dots, m \\ j = 1, 2, \dots, n. \end{cases}$$

The Chain Rule for Vector Functions (Cont.)

Then

$$\begin{aligned} \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right)^T &= \begin{bmatrix} \sum \frac{\partial z_1}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_1}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \cdots & \sum \frac{\partial z_1}{\partial y_q} \frac{\partial y_q}{\partial x_n} \\ \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \cdots & \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_n} \\ \vdots & & & \\ \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \cdots & \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_n} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} & \cdots & \frac{\partial z_1}{\partial y_r} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} & \cdots & \frac{\partial z_2}{\partial y_r} \\ \vdots & & & \\ \frac{\partial z_m}{\partial y_1} & \frac{\partial z_m}{\partial y_2} & \cdots & \frac{\partial z_m}{\partial y_r} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial y_r}{\partial x_1} & \frac{\partial y_r}{\partial x_2} & \cdots & \frac{\partial y_r}{\partial x_n} \end{bmatrix} \\ &= \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}} \right)^T \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)^T = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \right)^T. \end{aligned}$$

On transposing both sides, we finally obtain $\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}}$,

This is the chain rule for vectors (different from the conventional chain rule of calculus, the chain of matrices builds toward the left)

Example 2

~~x, y are as in Example 1 and z is a function of y defined as~~

$$z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix}, \text{ and } \begin{cases} z_1 = y_1^2 - 2y_2 \\ z_2 = y_2^2 - y_1 \\ z_3 = y_1^2 + y_2^2 \\ z_4 = 2y_1 + y_2 \end{cases}, \text{ we have}$$

$$\frac{\partial \mathbf{z}}{\partial y} = \begin{pmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_2}{\partial y_1} & \frac{\partial z_3}{\partial y_1} & \frac{\partial z_4}{\partial y_1} \\ \frac{\partial z_1}{\partial y_2} & \frac{\partial z_2}{\partial y_2} & \frac{\partial z_3}{\partial y_2} & \frac{\partial z_4}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 2y_1 & -1 & 2y_1 & 2 \\ -2 & 2y_2 & 2y_2 & 1 \end{pmatrix}.$$

Therefore,

$$\frac{\partial z}{\partial x} = \frac{\partial y}{\partial x} \frac{\partial z}{\partial y} = \begin{pmatrix} 2x_1 & 0 \\ -1 & 3 \\ 0 & 2x_3 \end{pmatrix} \begin{pmatrix} 2y_1 & -1 & 2y_1 & 2 \\ -2 & 2y_2 & 2y_2 & 1 \end{pmatrix} = \begin{pmatrix} 4x_1y_1 & -2x_1 & 4x_1y_1 & 4x_1 \\ -2y_1 - 6 & 1 + 6y_2 & -2y_2 + 6y_2 & 1 \\ -4x_3 & 4x_3y_2 & 4x_3y_2 & 2x_3 \end{pmatrix}$$

List of Differentiation

- Result of differentiating various kinds of aggregates with other kinds of aggregates.

Derivative Formulas

\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
\mathbf{Ax}	\mathbf{A}	\mathbf{Ax}	\mathbf{A}
$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T	$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T
$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$	$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$
$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$	$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$
\mathbf{x}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
\mathbf{Ax}	\mathbf{A}	\mathbf{Ax}	\mathbf{A}
$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T	$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T
$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$	$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$
$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$	$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$
\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
\mathbf{Ax}	\mathbf{A}	\mathbf{Ax}	\mathbf{A}
$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T	$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T
$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$	$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$
$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$	$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$
\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
\mathbf{Ax}	\mathbf{A}	\mathbf{Ax}	\mathbf{A}
$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T	$\mathbf{x}^T \mathbf{A}$	\mathbf{A}^T
$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$	$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}^T$
$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$	$\mathbf{x}^T \mathbf{Ax}$	$\mathbf{x}^T \mathbf{A} + \mathbf{x}^T \mathbf{A}^T$

- Hint: Derive \mathbf{x}

- If you have to differentiate \mathbf{x}^T , transpose the rest.
- If you have two \mathbf{x} -terms, differentiate them separately in turn and then sum up the two derivatives.

A Review of Derivative-based Optimization

Nonlinear Programming

Multivariable Unconstrained Optimization

- For functions with one variable, we use the 1st and 2nd derivatives.
- For functions with multiple variables, we use identical information that is the gradient and the Hessian.
- The gradient is the first derivative with respect to all variables whereas the Hessian is the equivalent of the second derivative

The Gradient

- Review of the gradient (∇):

For a function “ f ”, of variables x_1, x_2, \dots, x_n :

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Example: $f = 15x_1 + 2(x_2)^3 - 3x_1(x_3)^2$

$$\nabla f = \begin{bmatrix} 15 - 3(x_3)^2 & 6(x_2)^2 & -6x_1x_3 \end{bmatrix}$$

The Hessian

- The Hessian (∇^2) of $f(x_1, x_2, \dots, x_n)$ is:

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Hessian Example

- Example :

$$f = 15x_1 + 2(x_2)^3 - 3x_1(x_3)^2$$

$$\nabla f = \begin{bmatrix} 15 - 3(x_3)^2 & 6(x_2)^2 & -6x_1x_3 \end{bmatrix}$$
$$\nabla^2 f = \begin{bmatrix} 0 & 0 & -6x_3 \\ 0 & 12x_2 & 0 \\ -6x_3 & 0 & -6x_1 \end{bmatrix}$$

Unconstrained Optimization

The optimization procedure for multivariable functions is:

1. Solve for the gradient of the function equal to zero to obtain candidate points.
2. Obtain the Hessian of the function and evaluate it at each of the candidate points
 - If the result is “positive definite” then the point is a local minimum.
 - If the result is “negative definite” then the point is a local maximum.

Positive/Negative Definite

- A matrix is “positive definite” if all of the eigenvalues of the matrix are positive (> 0)
- A matrix is “negative definite” if all of the eigenvalues of the matrix are negative (< 0)

Positive/Negative Semi-definite

- A matrix is “positive semi-definite” if all of the eigenvalues are non-negative (≥ 0)
- A matrix is “negative semi-definite” if all of the eigenvalues are non-positive (≤ 0)

Example Matrix

Given the matrix A :

$$A = \begin{bmatrix} 2 & 4 & 5 \\ -5 & -7 & -1 \\ 1 & 1 & 2 \end{bmatrix}$$

The eigenvalues of A are:

$$\lambda_1 = -3.702$$

$$\lambda_2 = -2$$

$$\lambda_3 = 2.702$$

This matrix is negative definite

Unconstrained NLP Example

Consider the problem:

$$\text{Minimize } f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2x_3 + (x_3)^2 + x_3$$

First, we find the gradient with respect to x_i :

$$\nabla f = \begin{bmatrix} 2x_1 + 1 - x_2 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{bmatrix}$$

Unconstrained NLP Example

Next, we set the gradient equal to zero:

$$\nabla f = 0 \quad \Rightarrow \quad \begin{bmatrix} 2x_1 + 1 - x_2 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

So, we have a system of 3 equations and 3 unknowns. When we solve, we get:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Unconstrained NLP Example

So we have only one candidate point to check.

Find the Hessian:

$$\nabla^2 f = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Unconstrained NLP Example

The eigenvalues of this matrix are:

$$\lambda_1 = 3.414 \quad \lambda_2 = 0.586 \quad \lambda_3 = 2$$

All of the eigenvalues are > 0 , so the Hessian is positive definite.

So, the point $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$ is a minimum

Method of Solution

- In the previous example, when we set the gradient equal to zero, we had a system of 3 linear equations & 3 unknowns.
- For other problems, these equations could be nonlinear.
- Thus, the problem can become trying to solve a system of nonlinear equations, which can be very difficult.

Method of Solution

- To avoid this difficulty, NLP problems are usually solved numerically.
- We will now look at examples of numerical methods used to find the optimum point for single-variable NLP problems. These and other methods may be found in any numerical methods reference.

Convex optimization problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_1(x) \leq 0 \\ & && \dots \\ & && f_m(x) \leq 0 \end{aligned}$$

- objective and constraint functions are convex: for $0 \leq \theta \leq 1$

$$f_i(\theta x + (1 - \theta)y) \leq \theta f_i(x) + (1 - \theta)f_i(y)$$

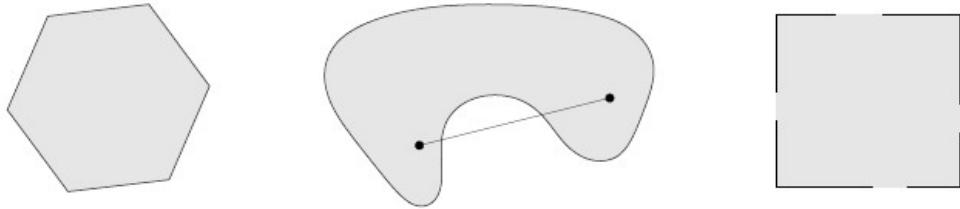
- includes least-squares problems and linear programs as special cases
- can be solved exactly, with similar complexity as LPs
- surprisingly many problems can be solved via convex optimization

Convex set

contains line segment between any two points in the set

$$x_1, x_2 \in C, \quad 0 \leq \theta \leq 1 \quad \Rightarrow \quad \theta x_1 + (1 - \theta)x_2 \in C$$

Examples: one convex, two nonconvex sets



Examples and properties

- solution set of linear equations $Ax = b$ (affine set)
- solution set of linear inequalities $Ax \leq b$ (polyhedron)
- norm balls $\{x \mid \|x\| \leq R\}$ and norm cones $\{(x, t) \mid \|x\| \leq t\}$
- set of positive semidefinite matrices $\mathbf{S}_+^n = \{X \in \mathbf{S}^n \mid X \succeq 0\}$

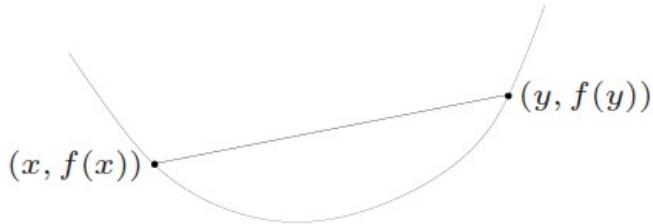
- image of a convex set under a linear transformation is convex
- inverse image of a convex set under a linear transformation is convex
- intersection of convex sets is convex

Convex function

domain $\text{dom } f$ is a convex set and

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$$

for all $x, y \in \text{dom } f$, $0 \leq \theta \leq 1$



f is concave if $-f$ is convex

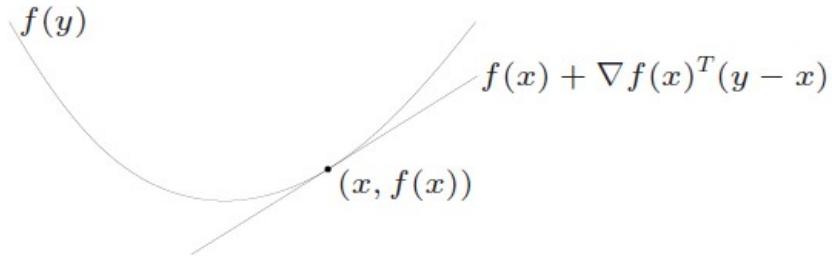
Examples

- $\exp x$, $-\log x$, $x \log x$ are convex
- x^α is convex for $x > 0$ and $\alpha \geq 1$ or $\alpha \leq 0$; $|x|^\alpha$ is convex for $\alpha \geq 1$
- quadratic-over-linear function $x^T x/t$ is convex in x , t for $t > 0$
- geometric mean $(x_1 x_2 \cdots x_n)^{1/n}$ is concave for $x \succeq 0$
- $\log \det X$ is concave on set of positive definite matrices
- $\log(e^{x_1} + \cdots e^{x_n})$ is convex
- linear and affine functions are convex and concave
- norms are convex

Differentiable convex functions

differentiable f is convex if and only if $\text{dom } f$ is convex and

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) \quad \text{for all } x, y \in \text{dom } f$$



twice differentiable f is convex if and only if $\text{dom } f$ is convex and

$$\nabla^2 f(x) \succeq 0 \quad \text{for all } x \in \text{dom } f$$

Operations that preserve convexity

methods for establishing convexity of a function

1. verify definition
2. for twice differentiable functions, show $\nabla^2 f(x) \succeq 0$
3. show that f is obtained from simple convex functions by operations that preserve convexity
 - nonnegative weighted sum
 - composition with affine function
 - pointwise maximum and supremum
 - composition
 - minimization
 - perspective

Convex optimization problem

$$\begin{array}{ll}\text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & Ax = b\end{array}$$

f_0, f_1, \dots, f_m are convex functions

- feasible set is convex
- locally optimal points are globally optimal
- tractable, both in theory and practice

Multivariable Optimization

- Now we will consider unconstrained multivariable optimization
- Nearly all multivariable optimization methods do the following:
 1. Choose a search direction \mathbf{d}^k
 2. Minimize along that direction to find a new point:
$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$$

where k is the current iteration number and α^k is a positive scalar called the step size.

The Step Size

- The step size, α^k , is calculated in the following way:
- We want to minimize the function $f(\mathbf{x}^{k+1}) = f(\mathbf{x}^k + \alpha^k \mathbf{d}^k)$ where the only variable is α^k because \mathbf{x}^k & \mathbf{d}^k are known.
- We set $\frac{df(\mathbf{x}^k + \alpha^k \mathbf{d}^k)}{d\alpha^k} = 0$ and solve for α^k using a single-variable solution method such as the ones shown previously.

Steepest Descent Method

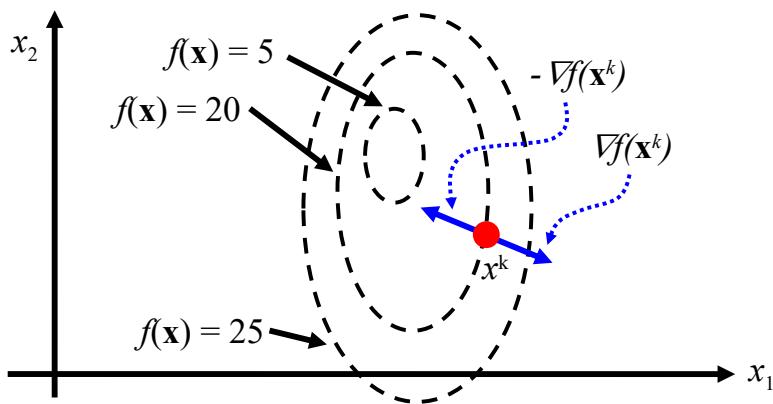
- This method is very simple – it uses the gradient (for maximization) or the negative gradient (for minimization) as the search direction:

$$\mathbf{d}^k = \begin{cases} + \\ - \end{cases} \nabla f(\mathbf{x}^k) \text{ for } \begin{cases} \max \\ \min \end{cases}$$

So, $\mathbf{x}^{k+1} = \mathbf{x}^k \begin{cases} + \\ - \end{cases} \alpha^k \nabla f(\mathbf{x}^k)$

Steepest Descent Method

- Because the gradient is the rate of change of the function at that point, using the gradient (or negative gradient) as the search direction helps reduce the number of iterations needed



Steepest Descent Method Steps

So the steps of the Steepest Descent Method are:

- Choose an initial point \mathbf{x}^0
- Calculate the gradient $\nabla f(\mathbf{x}^k)$ where k is the iteration number
- Calculate the search vector: $\mathbf{d}^k = \pm \nabla f(\mathbf{x}^k)$
- Calculate the next \mathbf{x} : $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$
Use a single-variable optimization method to determine α^k .

Steepest Descent Method Steps

5. To determine convergence, either use some given tolerance ε_1 and evaluate:

$$|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)| < \varepsilon_1$$

for convergence

- Or, use another tolerance ε_2 and evaluate:

$$\|\nabla f(\mathbf{x}^k)\| < \varepsilon_2$$

for convergence

Convergence

- These two criteria can be used for any of the multivariable optimization methods discussed here

Recall: The norm of a vector, $\|\mathbf{x}\|$ is given by:

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \cdot \mathbf{x}} = \sqrt{(x_1)^2 + (x_2)^2 + \cdots + (x_n)^2}$$

Steepest Descent Example

Let's solve the earlier problem with the Steepest Descent Method:

Minimize $f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2x_3 + (x_3)^2 + x_3$

Let's pick $\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Steepest Descent Example

$$\nabla f(\mathbf{x}) = [2x_1 + (1 - x_2) \quad -x_1 + 2x_2 - x_3 \quad -x_2 + 2x_3 + 1]$$

$$\begin{aligned} \mathbf{d}^0 &= -\nabla f(\mathbf{x}^0) = -[2(0) + 1 - 0 \quad -0 + 0 - 0 \quad -0 + 0 + 1] \\ &= -[1 \quad 0 \quad 1] = [-1 \quad 0 \quad -1] \end{aligned}$$

$$\mathbf{x}^1 = [0 \quad 0 \quad 0] + \alpha^0 \cdot [-1 \quad 0 \quad -1]$$

Now, we need to determine α^0

Steepest Descent Example

$$\begin{aligned}f(\mathbf{x}^1) &= (\alpha^0)^2 + (-\alpha^0)(1) + 0 - 0 + (\alpha^0)^2 + (-\alpha^0) \\&= 2(\alpha^0)^2 - 2(\alpha^0)\end{aligned}$$

$$\frac{df(\mathbf{x}^1)}{d\alpha^0} = 4(\alpha^0) - 2$$

Now, set equal to zero and solve:

$$4(\alpha^0) = 2 \Rightarrow \alpha^0 = \frac{2}{4} = \frac{1}{2}$$

Steepest Descent Example

So,

$$\begin{aligned}\mathbf{x}^1 &= [0 \ 0 \ 0] + \alpha^0 \cdot [-1 \ 0 \ -1] \\&= [0 \ 0 \ 0] + \left[-\frac{1}{2} \ 0 \ -\frac{1}{2} \right] \\&\therefore \mathbf{x}^1 = \left[-\frac{1}{2} \ 0 \ -\frac{1}{2} \right]\end{aligned}$$

Steepest Descent Example

Take the negative gradient to find the next search direction:

$$\mathbf{d}^1 = -\nabla f(\mathbf{x}^1) = - \begin{bmatrix} -1+1+0 & \frac{1}{2}+0+\frac{1}{2} & 0-1+1 \end{bmatrix}$$
$$\therefore \mathbf{d}^1 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}$$

Steepest Descent Example

Update the iteration formula:

$$\mathbf{x}^2 = \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} + \alpha^1 \cdot \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} -\frac{1}{2} & -\alpha^1 & -\frac{1}{2} \end{bmatrix}$$

Steepest Descent Example

Insert into the original function & take the derivative so that we can find α^1 :

$$\begin{aligned}f(\mathbf{x}^2) &= \frac{1}{4} + \left(-\frac{1}{2}\right)(1 + \alpha^1) + (\alpha^1)^2 - (\alpha^1)\left(\frac{1}{2}\right) + \frac{1}{4} - \frac{1}{2} \\&= (\alpha^1)^2 - \alpha^1 - \frac{1}{2}\end{aligned}$$

$$\frac{df(\mathbf{x}^1)}{d\alpha^1} = 2(\alpha^1) - 1$$

Steepest Descent Example

Now we can set the derivative equal to zero and solve for α^1 :

$$2(\alpha^1) = 1 \Rightarrow \alpha^1 = \frac{1}{2}$$

Steepest Descent Example

Now, calculate \mathbf{x}^2 :

$$\mathbf{x}^2 = \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} + \alpha^1 \cdot \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} + \begin{bmatrix} 0 & -\frac{1}{2} & 0 \end{bmatrix}$$

$$\therefore \mathbf{x}^2 = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

Steepest Descent Example

$$\mathbf{d}^2 = -\nabla f(\mathbf{x}^2) = -\begin{bmatrix} -1+1+\frac{1}{2} & \frac{1}{2}-1+\frac{1}{2} & \frac{1}{2}-1+1 \end{bmatrix}$$

$$\therefore \mathbf{d}^2 = \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}$$

So,

$$\mathbf{x}^3 = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} + \alpha^2 \cdot \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{2}(\alpha^2+1) & -\frac{1}{2} & -\frac{1}{2}(\alpha^2+1) \end{bmatrix}$$

Steepest Descent Example

Find α^2 : $f(\mathbf{x}^3) = \frac{1}{2}(\alpha^2 + 1)^2 - \frac{3}{2}(\alpha^2 + 1) + \frac{1}{4}$

$$\frac{df(\mathbf{x}^3)}{d\alpha^2} = (\alpha^2 + 1) - \frac{3}{2}$$

Set the derivative equal to zero and solve:

$$(\alpha^2 + 1) = \frac{3}{2} \Rightarrow \alpha^2 = \frac{1}{2}$$

Steepest Descent Example

Calculate \mathbf{x}^3 :

$$\begin{aligned}\mathbf{x}^3 &= \left[-\frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2} \right] + \alpha^2 \cdot \left[-\frac{1}{2} \quad 0 \quad -\frac{1}{2} \right] \\ &= \left[-\frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2} \right] + \left[-\frac{1}{4} \quad 0 \quad -\frac{1}{4} \right]\end{aligned}$$

$$\therefore \mathbf{x}^3 = \left[-\frac{3}{4} \quad -\frac{1}{2} \quad -\frac{3}{4} \right]$$

Steepest Descent Example

Find the next search direction:

$$\mathbf{d}^3 = -\nabla f(\mathbf{x}^3) = -\begin{bmatrix} 0 & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{2} & 0 \end{bmatrix}$$

$$\begin{aligned}\mathbf{x}^4 &= \left[-\frac{3}{4} \quad -\frac{1}{2} \quad -\frac{3}{4} \right] + \alpha^3 \cdot \begin{bmatrix} 0 & -\frac{1}{2} & 0 \end{bmatrix} \\ &= \left[-\frac{3}{4} \quad -\frac{1}{2}(\alpha^3 + 1) \quad -\frac{3}{4} \right]\end{aligned}$$

Steepest Descent Example

Find α^3 : $f(\mathbf{x}^4) = \frac{1}{4}(\alpha^3 + 1)^2 - \frac{3}{2}(\alpha^3) - \frac{3}{2}$

$$\frac{df(\mathbf{x}^4)}{d\alpha^3} = \frac{1}{2}(\alpha^3 + 1) - \frac{9}{8} = 0$$

$$\Rightarrow \alpha^3 = \frac{5}{4}$$

Steepest Descent Example

So, \mathbf{x}^4 becomes:

$$\mathbf{x}^4 = \left[-\frac{3}{4} \quad -\frac{1}{2} \quad -\frac{3}{4} \right] + \left[0 \quad -\frac{5}{8} \quad 0 \right]$$

$$\therefore \mathbf{x}^4 = \left[-\frac{3}{4} \quad -\frac{9}{8} \quad -\frac{3}{4} \right]$$

Steepest Descent Example

The next search direction:

$$\mathbf{d}^4 = -\nabla f(\mathbf{x}^4) = -\left[\frac{5}{8} \quad -\frac{3}{4} \quad \frac{5}{8} \right] = \left[-\frac{5}{8} \quad \frac{3}{4} \quad -\frac{5}{8} \right]$$

$$\begin{aligned}\mathbf{x}^5 &= \left[-\frac{3}{4} \quad -\frac{9}{8} \quad -\frac{3}{4} \right] + \alpha^4 \cdot \left[-\frac{5}{8} \quad \frac{3}{4} \quad -\frac{5}{8} \right] \\ &= \left[-\frac{1}{4}(3 + \frac{5}{2}\alpha^4) \quad -\frac{3}{4}(\frac{3}{2} - \alpha^4) \quad -\frac{1}{4}(3 + \frac{5}{2}\alpha^4) \right]\end{aligned}$$

Steepest Descent Example

Find α^4 : $f(\mathbf{x}^5) = \frac{73}{32}(\alpha^4)^2 - \frac{43}{32}\alpha^4 - \frac{51}{64}$

$$\frac{df(\mathbf{x}^5)}{d\alpha^4} = \frac{73}{16}\alpha^4 - \frac{43}{32} = 0$$

$$\Rightarrow \alpha^4 = \frac{43}{146}$$

Steepest Descent Example

Update \mathbf{x}^5 :

$$\mathbf{x}^5 = \left[-\frac{3}{4} \quad -\frac{9}{8} \quad -\frac{3}{4} \right] + \frac{43}{146} \cdot \left[-\frac{5}{8} \quad \frac{3}{4} \quad -\frac{5}{8} \right]$$

$$\therefore \mathbf{x}^5 = \left[-\frac{1091}{1168} \quad -\frac{66}{73} \quad -\frac{1091}{1168} \right]$$

Steepest Descent Example

Let's check to see if the convergence criteria is satisfied

Evaluate $\|\nabla f(\mathbf{x}^5)\|$:

$$\nabla f(\mathbf{x}^5) = \begin{bmatrix} \frac{21}{584} & \frac{35}{584} & \frac{21}{584} \end{bmatrix}$$

$$\|\nabla f(\mathbf{x}^5)\| = \sqrt{\left(\frac{21}{584}\right)^2 + \left(\frac{35}{584}\right)^2 + \left(\frac{21}{584}\right)^2} = 0.0786$$

Steepest Descent Example

So, $\|\nabla f(\mathbf{x}^5)\| = 0.0786$, which is very small and we can take it to be close enough to zero for our example

Notice that the answer of

$$\mathbf{x} = \begin{bmatrix} -\frac{1091}{1168} & -\frac{66}{73} & -\frac{1091}{1168} \end{bmatrix}$$

is very close to the value of $\mathbf{x}^* = [-1 \quad -1 \quad -1]$ that we obtained analytically

Quadratic Functions

- Quadratic functions are important for the next method we will look at
- A quadratic function can be written in the form: $\mathbf{x}^T \mathbf{Q} \mathbf{x}$ where \mathbf{x} is the vector of variables and \mathbf{Q} is a matrix of coefficients

Example:

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 2(x_1)^2 - 2x_1x_2 + 2(x_2)^2 - 2x_2x_3 + 2(x_3)^2$$

Multivariable Newton's Method

We can approximate the gradient of f at a point \mathbf{x}^0 by:

$$\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x}^0) + \nabla^2 f(\mathbf{x}^0) \cdot (\mathbf{x} - \mathbf{x}^0)$$

We can set the right-hand side equal to zero and rearrange to give:

$$\mathbf{x} = \mathbf{x}^0 - [\nabla^2 f(\mathbf{x}^0)]^{-1} \cdot \nabla f(\mathbf{x}^0)$$

Multivariable Newton's Method

We can generalize this equation to give an iterative expression for the Newton's Method:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - [\nabla^2 f(\mathbf{x}^k)]^{-1} \cdot \nabla f(\mathbf{x}^k)$$

where k is the iteration number

Newton's Method Steps

1. Choose a starting point, \mathbf{x}^0
2. Calculate $\nabla f(\mathbf{x}^k)$ and $\nabla^2 f(\mathbf{x}^k)$
3. Calculate the next \mathbf{x} using the equation

$$\mathbf{x}^{k+1} = \mathbf{x}^k - [\nabla^2 f(\mathbf{x}^k)]^{-1} \cdot \nabla f(\mathbf{x}^k)$$

4. Use either of the convergence criteria discussed earlier to determine convergence. If it hasn't converged, return to step 2.

Comments on Newton's Method

- We can see that unlike the previous two methods, Newton's Method uses both the gradient and the Hessian
- This usually reduces the number of iterations needed, but increases the computation needed for each iteration
- So, for very complex functions, a simpler method is usually faster

Newton's Method Example

For an example, we will use the same problem as before:

$$\begin{aligned} \text{Minimize } f(x_1, x_2, x_3) = & (x_1)^2 + x_1(1 - x_2) + \\ & (x_2)^2 \\ & - x_2x_3 + (x_3)^2 + \\ & x_3 \end{aligned}$$

$$\nabla f(\mathbf{x}) = [2x_1 - x_2 + 1 \quad -x_1 + 2x_2 - x_3 \quad -x_2 + 2x_3 + 1]$$

Newton's Method Example

The Hessian is:

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

And we will need the inverse of the Hessian:

$$[\nabla^2 f(\mathbf{x})]^{-1} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{3}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{3}{4} \end{bmatrix}$$

Newton's Method Example

So, pick $\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Calculate the gradient for the 1st iteration:

$$\nabla f(\mathbf{x}^0) = [0 - 0 + 1 \quad -0 + 0 - 0 \quad -0 + 0 + 1]$$

$$\Rightarrow \nabla f(\mathbf{x}^0) = [1 \quad 0 \quad 1]$$

Newton's Method Example

So, the new \mathbf{x} is:

$$\mathbf{x}^1 = \mathbf{x}^0 - [\nabla^2 f(\mathbf{x}^0)]^{-1} \cdot \nabla f(\mathbf{x}^0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 3/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 3/4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\therefore \mathbf{x}^1 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Newton's Method Example

Now calculate the new gradient:

$$\nabla f(\mathbf{x}^1) = [-2+1+1 \quad 1-2+1 \quad 1-2+1] = [0 \quad 0 \quad 0]$$

Since the gradient is zero, the method has converged

Comments on Example

- Because it uses the 2nd derivative, Newton's Method models quadratic functions exactly and can find the optimum point in one iteration.
- If the function had been a higher order, the Hessian would not have been constant and it would have been much more work to calculate the Hessian and take the inverse for each iteration.

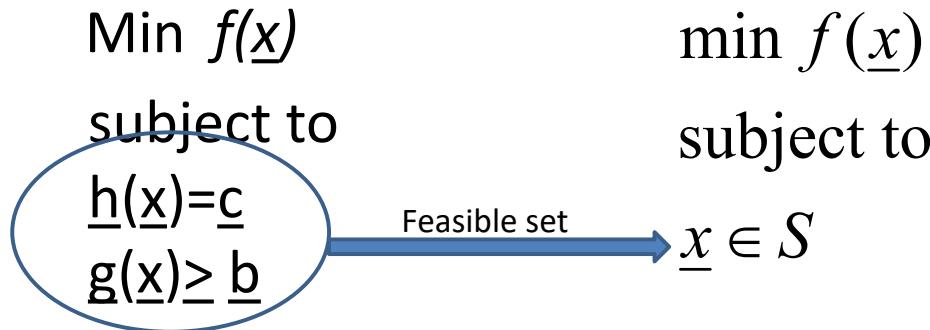
Constrained Nonlinear Optimization

- Previously in this chapter, we solved NLP problems that only had objective functions, with no constraints.
- Now we will look at methods on how to solve problems that include constraints.

Convexity & global vs. local optima

→ When minimizing a function, if we want to be sure that we can get a global solution via differentiation, we need to impose some requirements on our objective function.

→ We will also need to impose some requirements on the feasible set S (set of possible values the solution \underline{x}^* may take).



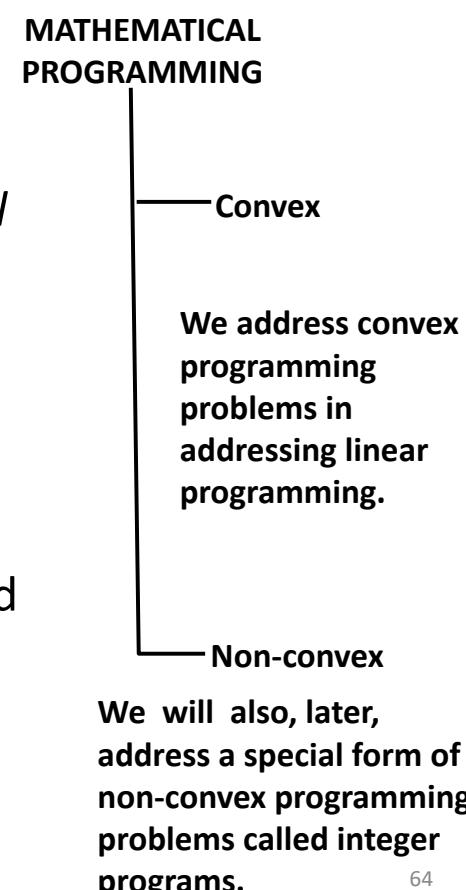
Definition: If $f(\underline{x})$ is a convex function, and if S is a convex set, then the above problem is a *convex programming problem*.

Definition: If $f(\underline{x})$ is not a convex function, or if S is not a convex set, then the above problem is a *non-convex programming problem*. ⁶³

Convex vs. nonconvex programming problems

The desirable quality of a convex programming problem is that any *locally optimal solution* is also a *globally optimal solution*. → If we have a method of finding a locally optimal solution, that method also finds for us the globally optimum solution.

The undesirable quality of a non-convex programming problem is that any method which finds a locally optimal solution does not necessarily find the globally optimum solution.



A convex programming problem

Two variables with one equality-constraint

$$\begin{aligned} & \min f(x_1, x_2) \\ \text{s.t. } & h(x_1, x_2) = c \end{aligned}$$

We focus on this one, but conclusions we derive will also apply to the other two. The benefit of focusing on this one is that we can visualize it.

Multi-variable with one equality-constraint.

$$\begin{aligned} & \min f(\underline{x}) \\ \text{s.t. } & h(\underline{x}) = \underline{c} \end{aligned}$$

Multi-variable with multiple equality-constraints.

$$\begin{aligned} & \min f(\underline{x}) \\ \text{s.t. } & h(\underline{x}) = \underline{c} \end{aligned}$$

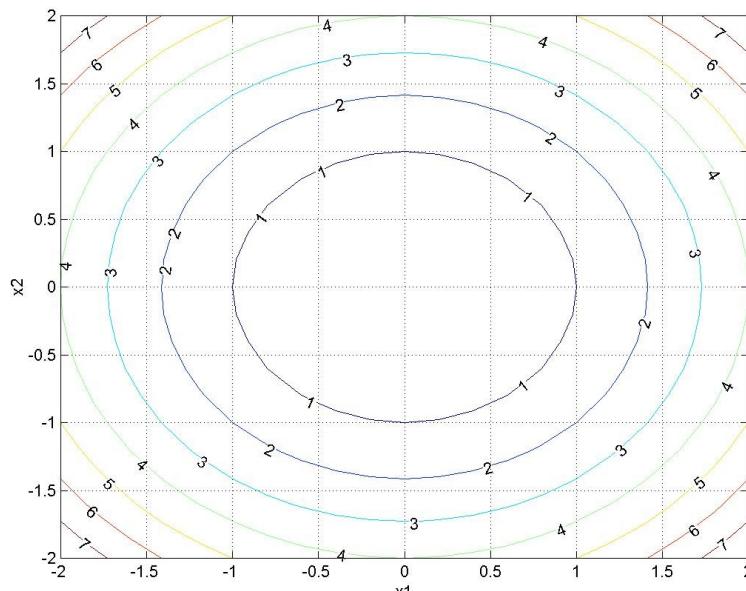
65

Contour maps/Constant Cost Contours/level Sets

Definition: A contour map is a 2-dimensional plane, i.e., a coordinate system in 2 variables, say, x_1, x_2 , that illustrates curves (contours) of constant functional value $f(x_1, x_2)$.

Example: Draw the contour map for $f(x_1, x_2) = x_1^2 + x_2^2$

```
[X,Y] = meshgrid(-2.0:.2:2.0,-2.0:.2:2.0);
Z = X.^2+Y.^2;
[c,h]=contour(X,Y,Z);
clabel(c,h);
grid;
xlabel('x1');
ylabel('x2');
```



66

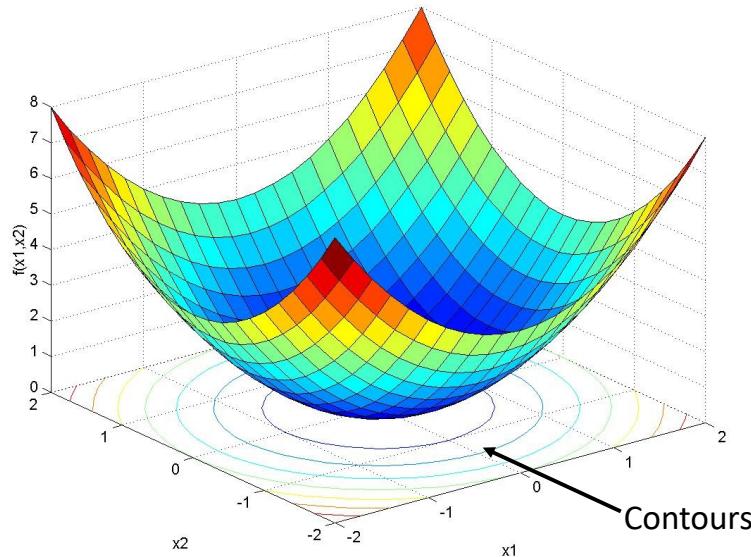
Contour maps and 3-D illustrations

Example: Draw the 3-D surface for $f(x_1, x_2) = x_1^2 + x_2^2$

```
[X,Y] = meshgrid(-2.0:2.0,-2.0:2.0);
Z = X.^2+Y.^2;
surf(X,Y,Z)
xlabel('x1')
ylabel('x2')
zlabel('f(x1,x2)')
```

Height is $f(x)$

Each contour of fixed value f is the projection onto the x_1 - x_2 plane of a horizontal slice made of the 3-D figure at a value f above the x_1 - x_2 plane.

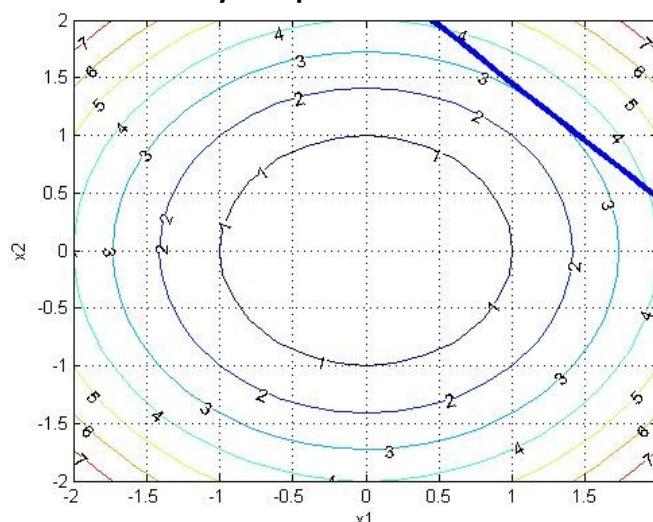


67

Solving a convex program: graphical analysis

Example: Solve this convex program: $\min f(x_1, x_2) = x_1^2 + x_2^2$

A straight line is a convex set because a line segment between any two points on it remain on it.



s.t. $h(x_1, x_2) = x_1 + x_2 = \sqrt{6}$

$x_1 + x_2 = \sqrt{6} \Rightarrow x_2 = -x_1 + \sqrt{6}$

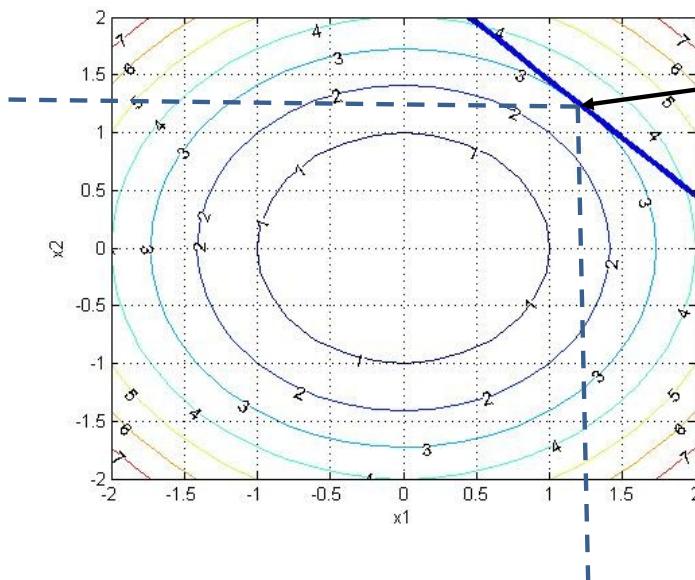


Superimpose this relation on top of the contour plot for $f(x_1, x_2)$.

1. $f(x_1, x_2)$ must be minimized, and so we would like the solution to be as close to the origin as possible;
2. The solution must be on the thick line in the right-hand corner of the plot, since this line represents the equality constraint.

68

Solving a convex program: graphical analysis



Solution:

$$\underline{x}^* = (x_1, x_2)^* \approx (1.25, 1.25)$$

$$f(x_1, x_2)^* = 3$$

Any contour $f < 3$ does not intersect the equality constraint;

Any contour $f > 3$ intersects the equality constraint at two points.

→ The contour $f=3$ and the equality constraint just touch each other at the point \underline{x}^* .

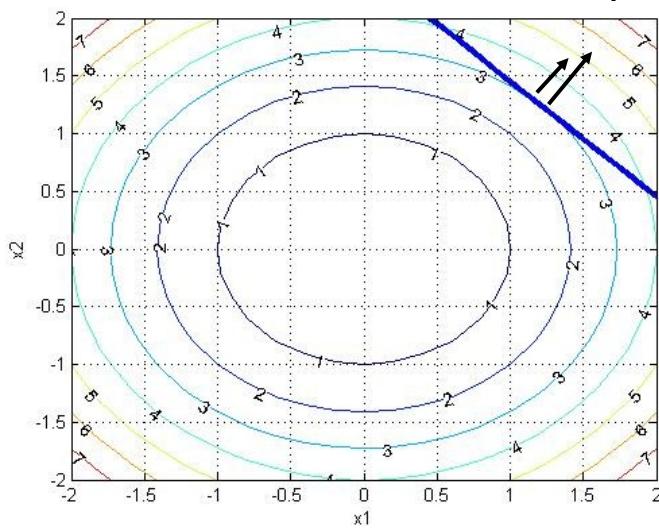
"Just touch":

The two curves are tangent to one another at the solution point.

69

Solving a convex program: graphical analysis

The two curves are tangent to one another at the solution point.



→ The normal (gradient) vectors of the two curves, at the solution (tangent) point, are parallel.

This means the following two vectors are parallel:

$$\nabla\{f(x_1, x_2)^*\} = \nabla\{x_1^2 + x_2^2\}^* = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$$

$$\nabla\{h(x_1, x_2)^* - \sqrt{6}\} = \nabla\{x_1 + x_2 - \sqrt{6}\}^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

"Parallel" means that the two vectors have the same direction. We do not know that they have the same magnitude. To account for this, we equate with a "multiplier" λ :

$$\nabla f(x_1, x_2)^* = \lambda \nabla(h(x_1, x_2)^* - c)$$

70

Solving a convex program: graphical analysis

$$\nabla f(x_1, x_2)^* = \lambda \nabla (h(x_1, x_2)^* - c)$$

Moving everything to the left:

$$\nabla f(x_1, x_2)^* - \lambda \nabla (h(x_1, x_2)^* - c) = 0$$


Alternately:

$$\nabla f(x_1, x_2)^* + \lambda \nabla (c - h(x_1, x_2)^*) = 0$$

Performing the gradient operation (taking derivatives with respect to x_1 and x_2):

$$\begin{bmatrix} \frac{\partial}{\partial x_1} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ \frac{\partial}{\partial x_2} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In this problem, we already know the solution, but what if we did not? Then could we use the above equations to find the solution?

71

Solving a convex program: analytical analysis

In this problem, we already know the solution, but what if we did not? Then could we use the above equations to find the solution?

$$\begin{bmatrix} \frac{\partial}{\partial x_1} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ \frac{\partial}{\partial x_2} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

NO! Because we only have 2 equations, yet 3 unknowns: x_1, x_2, λ .

So we need another equation. Where do we get that equation?

Recall our equality constraint: $h(x_1, x_2) - c = 0$. This must be satisfied!

Therefore:

$$\begin{bmatrix} \frac{\partial}{\partial x_1} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ \frac{\partial}{\partial x_2} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ h(x_1, x_2) - c \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

**Three equations,
three unknowns,
we can solve.**

72

Solving a convex program: analytical analysis

Observation: The three equations are simply partial derivatives of the function

$$f(x_1, x_2) - \lambda(h(x_1, x_2) - c)$$

$$\begin{bmatrix} \frac{\partial}{\partial x_1}(f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ \frac{\partial}{\partial x_2}(f(x_1, x_2) - \lambda|h(x_1, x_2) - c|) \\ h(x_1, x_2) - c \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

This is obviously true for the first two equations , but it is not so obviously true for the last one. But to see it, observe

$$\begin{aligned} \frac{\partial}{\partial \lambda}(f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) &= 0 \\ \Rightarrow -h(x_1, x_2) + c &= 0 \Rightarrow h(x_1, x_2) = c \end{aligned}$$

73

Formal approach to solving our problem

Define the Lagrangian function:

$$L(x_1, x_2, \lambda) = f(x_1, x_2) - \lambda(h(x_1, x_2) - c)$$

In a convex programming problem, the “first-order conditions” for finding the solution is given by

$$\nabla L(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial x_1} L(x_1, x_2, \lambda) = 0$$

OR $\frac{\partial}{\partial x_2} L(x_1, x_2, \lambda) = 0$  **Or more compactly** 

$$\frac{\partial}{\partial \lambda} L(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial \underline{x}} L(\underline{x}, \lambda) = 0$$

$$\frac{\partial}{\partial \lambda} L(\underline{x}, \lambda) = 0$$

where we have
used $\underline{x} = (x_1, x_2)$

74

Applying to our example

Define the Lagrangian function:

$$\begin{aligned} \mathcal{L}(x_1, x_2, \lambda) &= f(x_1, x_2) - \lambda(h(x_1, x_2) - c) \\ &= x_1^2 + x_2^2 - \lambda(x_1 + x_2 - \sqrt{6}) \end{aligned}$$

$$\nabla \mathcal{L}(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial x_1} \mathcal{L}(x_1, x_2, \lambda) = 0 \quad \frac{\partial}{\partial x_1} \mathcal{L}(x_1, x_2, \lambda) = 2x_1 - \lambda = 0$$

OR $\frac{\partial}{\partial x_2} \mathcal{L}(x_1, x_2, \lambda) = 0 \quad \frac{\partial}{\partial x_2} \mathcal{L}(x_1, x_2, \lambda) = 2x_2 - \lambda = 0$

$$\frac{\partial}{\partial \lambda} \mathcal{L}(x_1, x_2, \lambda) = 0 \quad \frac{\partial}{\partial \lambda} \mathcal{L}(x_1, x_2, \lambda) = -(x_1 + x_2 - \sqrt{6}) = 0$$

A set of 3 linear equations and 3 unknowns;
we can write in the form of $\mathbf{Ax}=\mathbf{b}$.

75

Applying to our example

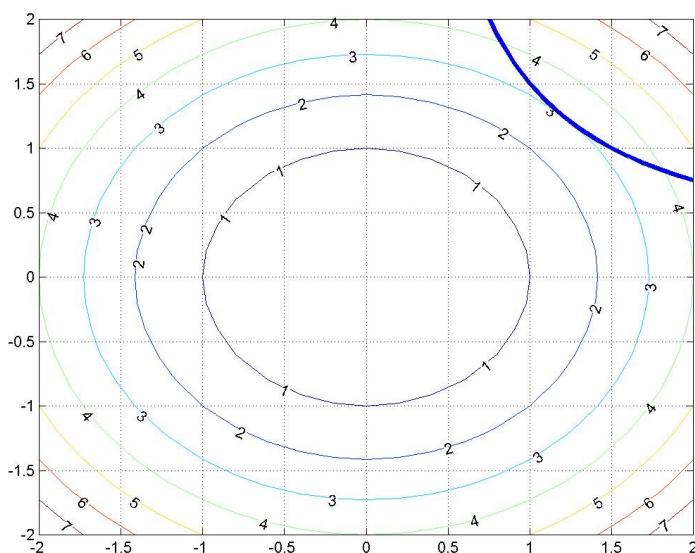
$$\begin{aligned} \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \\ \sqrt{6} \end{bmatrix} \\ \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} &= \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ 1 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \sqrt{6} \end{bmatrix} = \begin{bmatrix} 1.2247 \\ 1.2247 \\ 2.4495 \end{bmatrix} \end{aligned}$$

76

Now, let's go back to our example with a nonlinear equality constraint.

Example with nonlinear equality

Non-convex because a line connecting two points in the set do not remain in the set. (see "notes" of this slide)



$$\begin{aligned} \min \quad & f(x_1, x_2) = x_1^2 + x_2^2 \\ \text{s.t.} \quad & h(x_1, x_2) = 2x_1 x_2 = 3 \end{aligned}$$

↓

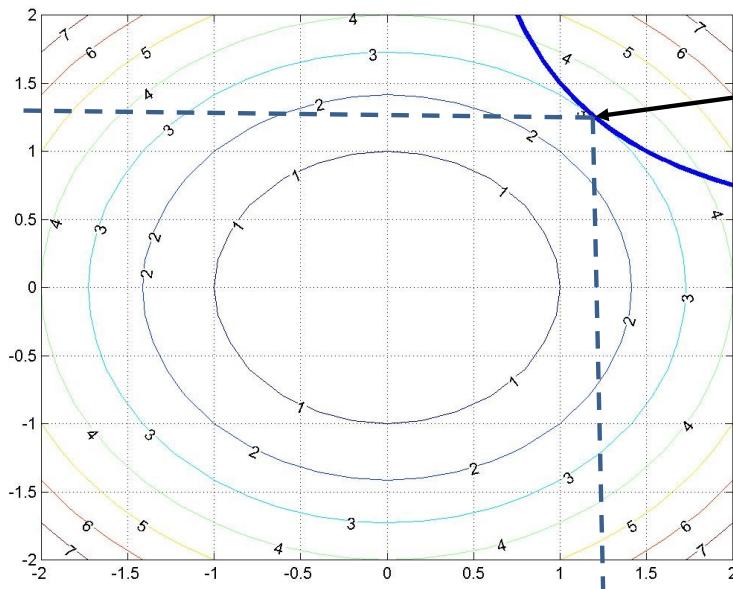
$$2x_1 x_2 = 3 \Rightarrow x_2 = \frac{3}{2x_1}$$

↓

 Superimpose this relation on top of the contour plot for $f(x_1, x_2)$.

1. $f(x_1, x_2)$ must be minimized, and so we would like the solution to be as close to the origin as possible;
2. The solution must be on the thick line in the right-hand corner of the plot, since this line represents the equality constraint.

Example with nonlinear equality



Solution:

$$\underline{x}^* = (x_1, x_2)^* \approx (1.25, 1.25)$$

$$f(x_1, x_2)^* = 3$$

Any contour $f < 3$ does not intersect the equality constraint;

Any contour $f > 3$ intersects the equality constraint at two points.

→ The contour $f=3$ and the equality constraint just touch each other at the point \underline{x}^* .

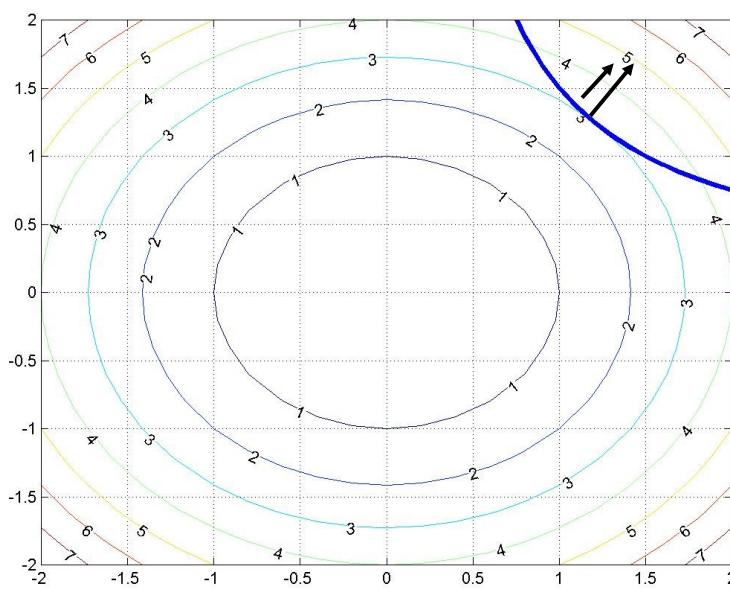
“Just touch”:

The two curves are tangent to one another at the solution point.

79

Example with nonlinear equality

The two curves are tangent to one another at the solution point.



→ The normal (gradient) vectors of the two curves, at the solution (tangent) point, are parallel.

This means the following two vectors are parallel:

$$\nabla\{f(x_1, x_2)^*\} = \nabla\{x_1^2 + x_2^2\}^* = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$$

$$\nabla\{h(x_1, x_2)^* - 3\} = \nabla\{2x_1 x_2 - 3\}^* = \begin{bmatrix} 2x_2 \\ 2x_1 \end{bmatrix}$$

“Parallel” means that the two vectors have the same direction. We do not know that they have the same magnitude. To account for this, we equate with a “multiplier” λ :

$$\nabla f(x_1, x_2)^* = \lambda \nabla(h(x_1, x_2)^* - c)$$

80

Example with nonlinear equality

This gives us the following two equations.

$$\begin{bmatrix} \frac{\partial}{\partial x_1} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ \frac{\partial}{\partial x_2} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

And we add the equality constraint to give 3 equations, 3 unknowns:

$$\begin{bmatrix} \frac{\partial}{\partial x_1} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ \frac{\partial}{\partial x_2} (f(x_1, x_2) - \lambda(h(x_1, x_2) - c)) \\ h(x_1, x_2) - c \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Three equations,
three unknowns,
we can solve.

81

Example with nonlinear equality

Define the Lagrangian function:

$$\begin{aligned} \mathcal{L}(x_1, x_2, \lambda) &= f(x_1, x_2) - \lambda(h(x_1, x_2) - c) \\ &= x_1^2 + x_2^2 - \lambda(2x_1x_2 - 3) \end{aligned}$$

$$\nabla \mathcal{L}(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial x_1} \mathcal{L}(x_1, x_2, \lambda) = 0 \quad \frac{\partial}{\partial x_2} \mathcal{L}(x_1, x_2, \lambda) = 2x_1 - 2\lambda x_2 = 0$$

OR $\frac{\partial}{\partial x_2} \mathcal{L}(x_1, x_2, \lambda) = 0 \Rightarrow \frac{\partial}{\partial x_2} \mathcal{L}(x_1, x_2, \lambda) = 2x_2 - 2\lambda x_1 = 0$

$$\frac{\partial}{\partial \lambda} \mathcal{L}(x_1, x_2, \lambda) = 0 \quad \frac{\partial}{\partial \lambda} \mathcal{L}(x_1, x_2, \lambda) = -(2x_1x_2 - 3) = 0$$

You can solve this algebraically to obtain

$$x_1 = x_2 = \sqrt{\frac{3}{2}} = 1.2247$$

$$x_1 = x_2 = -\sqrt{\frac{3}{2}} = -1.2247$$

and f=3 in
both cases

82

Example with nonlinear equality

Our approach worked in this case, i.e., we found a local optimal point that was also a global optimal point, but because it was not a convex programming problem, we had no guarantee that this would happen.

The conditions we established, below, we call first order conditions.

For convex programming problems, they are first order *sufficient conditions* to provide the global optimal point.

For nonconvex programming problems, they are first order *necessary conditions* to provide the global optimal point.

$$\frac{\partial}{\partial \underline{x}} \mathcal{L}(\underline{x}, \underline{\lambda}) = 0$$

$$\frac{\partial}{\partial \underline{\lambda}} \mathcal{L}(\underline{x}, \underline{\lambda}) = 0$$

Multiple equality constraints

$$\begin{aligned} \min \quad & f(\underline{x}) \\ \text{s.t.} \quad & \underline{h}(\underline{x}) = \underline{c} \end{aligned}$$

We assume that f and \underline{h} are continuously differentiable.

$$\begin{aligned} \mathcal{L}(\underline{x}, \underline{\lambda}) = & f(\underline{x}) - \lambda_1(h_1(\underline{x}) - c_1) - \lambda_2(h_2(\underline{x}) - c_2) \\ & - \dots - \lambda_m(h_m(\underline{x}) - c_m) \end{aligned}$$

First order necessary conditions that $(\underline{x}^*, \underline{\lambda}^*)$ solves the above:

$$\frac{\partial}{\partial \underline{x}} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*) = 0$$

$$\frac{\partial}{\partial \underline{\lambda}} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*) = 0$$

Multiple equality & 1 inequality constraint

$$\begin{aligned} \min \quad & f(\underline{x}) \\ \text{s.t.} \quad & h(\underline{x}) = c \\ & g(\underline{x}) \geq b \end{aligned}$$

We assume that f , h , and g are continuously differentiable.

Solution approach:

- Ignore the inequality constraint and solve the problem.
(this is just a problem with multiple equality constraints).
- If inequality constraint is satisfied, then problem is solved.
- If inequality constraint is violated, then the inequality constraint must be binding \Rightarrow inequality constraint enforced with equality:

$$g(\underline{x}) = b$$

Let's look at this new problem where the inequality is binding.

85

Multiple equality & 1 inequality constraint

$$\begin{aligned} \min \quad & f(\underline{x}) \\ \text{s.t.} \quad & h(\underline{x}) = c \\ & g(\underline{x}) = b \end{aligned}$$

We assume that f , h , and g are continuously differentiable.

$$\begin{aligned} \mathcal{L}(\underline{x}, \underline{\lambda}, \mu) = & f(\underline{x}) - \lambda_1(h_1(\underline{x}) - c_1) - \lambda_2(h_2(\underline{x}) - c_2) \\ & - \dots - \lambda_m(h_m(\underline{x}) - c_m) - \mu(g(\underline{x}) - b) \end{aligned}$$

First order necessary conditions that $(\underline{x}^*, \underline{\lambda}^*, \mu^*)$ solves the above:

$$\frac{\partial}{\partial \underline{x}} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*, \mu^*) = 0$$

$$\frac{\partial}{\partial \underline{\lambda}} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*, \mu^*) = 0$$

$$\frac{\partial}{\partial \mu} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*, \mu^*) = 0$$

We were able to write down this solution only after we knew the inequality constraint was binding. Can we generalize this approach?

86

Multiple equality & 1 inequality constraint

$$\mathcal{L}(\underline{x}, \underline{\lambda}, \mu) = f(\underline{x}) - \lambda_1(h_1(\underline{x}) - c_1) - \lambda_2(h_2(\underline{x}) - c_2) - \dots - \lambda_m(h_m(\underline{x}) - c_m) - \mu(g(\underline{x}) - b)$$

If inequality is not binding, then apply first order necessary conditions by ignoring it:

$\rightarrow \mu = 0$

$\rightarrow g(\underline{x}) - b \neq 0$ (since it is not binding!)

If inequality is binding, then apply first order necessary conditions treating inequality constraint as an equality constraint

$\rightarrow \mu \neq 0$

$\rightarrow g(\underline{x}) - b \neq 0$ (since it is binding!)

Either way:

$$\mu(g(\underline{x}) - b) = 0$$

This relation encodes our solution procedure!

It can be used to generalize our necessary conditions

87

Multiple equality & multiple inequality constraints

$$\begin{aligned} \min \quad & f(\underline{x}) \\ \text{s.t.} \quad & h(\underline{x}) = \underline{c} \\ & g(\underline{x}) = \underline{b} \end{aligned}$$

We assume that f , h , and g are continuously differentiable.

$$\begin{aligned} \mathcal{L}(\underline{x}, \underline{\lambda}, \underline{\mu}) = & f(\underline{x}) - \lambda_1(h_1(\underline{x}) - c_1) - \lambda_2(h_2(\underline{x}) - c_2) - \dots - \lambda_m(h_m(\underline{x}) - c_m) \\ & - \mu_1(g_1(\underline{x}) - b_1) - \mu_2(g_2(\underline{x}) - b_2) - \dots - \mu_n(g_n(\underline{x}) - b_n) \end{aligned}$$

First order necessary conditions that $(\underline{x}^*, \underline{\lambda}^*, \underline{\mu}^*)$ solves the above:

$$\frac{\partial}{\partial \underline{x}} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*, \underline{\mu}^*) = 0$$

$$\frac{\partial}{\partial \underline{\lambda}} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*, \underline{\mu}^*) = 0$$

$$\frac{\partial}{\partial \underline{\mu}} \mathcal{L}(\underline{x}^*, \underline{\lambda}^*, \underline{\mu}^*) = 0$$

These conditions also referred to as the Kurash-Kuhn-Tucker (KKT) conditions

Nonnegativity on inequality multipliers.

$$\mu_k^*(g_k(\underline{x}^*) - b) = 0 \quad \forall k$$

$$\mu_k^* \geq 0 \quad \forall k$$

Complementarity condition: Inactive constraints have a zero multiplier.

88

An additional requirement

$$\begin{aligned} & \min f(\underline{x}) \\ \text{s.t. } & \underline{h}(\underline{x}) = \underline{c} \\ & \underline{g}(\underline{x}) = \underline{b} \end{aligned}$$

We assume that f , h , and g are continuously differentiable.

For KKT to guarantee finds a local optimum, we need the Kuhn-Tucker Constraint Qualification (even under convexity).

This condition imposes a certain restriction on the constraint functions.

Its purpose is to rule out certain irregularities on the boundary of the feasible set, that would invalidate the Kuhn-Tucker conditions should the optimal solution occur there.

We will not try to tackle this idea, but know this:

→ If the feasible region is a convex set formed by *linear* constraints only, then the constraint qualification will be met, and the Kuhn-Tucker conditions will always hold at an optimal solution.

Optimization and Lagrange Multipliers

This material corresponds roughly to sections 14.7 and 14.8 in the book.

Taylor's Formula:

Let $T(x, y, z)$ be a scalar field and consider a small displacement $\Delta \mathbf{r} = (\Delta x, \Delta y, \Delta z)$. Suppose T is of class C^3 , that is, T has partial derivatives of at least third order, each of them being continuous. Then **Taylor's formula** is

$$\begin{aligned} T(x + \Delta x, y + \Delta y, z + \Delta z) &= T(x, y, z) + \frac{\partial T(x, y, z)}{\partial x} \Delta x + \frac{\partial T(x, y, z)}{\partial y} \Delta y + \frac{\partial T(x, y, z)}{\partial z} \Delta z \\ &\quad + \frac{1}{2} \left(\frac{\partial^2 T(x, y, z)}{\partial x^2} (\Delta x)^2 + \frac{\partial^2 T(x, y, z)}{\partial y^2} (\Delta y)^2 + \frac{\partial^2 T(x, y, z)}{\partial z^2} (\Delta z)^2 \right) \\ &\quad + \frac{1}{2} \left(2 \frac{\partial^2 T(x, y)}{\partial x \partial y} \Delta x \Delta y + 2 \frac{\partial^2 T(x, y)}{\partial x \partial z} \Delta x \Delta z + 2 \frac{\partial^2 T(x, y)}{\partial y \partial z} \Delta y \Delta z \right) + R_2(\Delta \mathbf{r}) \end{aligned}$$

where

$$\lim_{\Delta \mathbf{r} \rightarrow \mathbf{0}} \frac{R_2(\Delta \mathbf{r})}{|\Delta \mathbf{r}|^2} = 0 \quad (1)$$

More succinctly, we can write

$$T(\mathbf{r} + \Delta \mathbf{r}) = T(\mathbf{r}) + \nabla T(\mathbf{r}) \Delta \mathbf{r} + \frac{1}{2} (\Delta \mathbf{r})^T H_T(\mathbf{r}) \Delta \mathbf{r} + R_2(\Delta \mathbf{r}) \quad (2)$$

Where we have represented $\Delta \mathbf{r}$ as a column vector (instead of a row vector) and $H_T(\mathbf{r})$ is the Hessian **Hessian** of the scalar field

$$H_T = \begin{pmatrix} \frac{\partial^2 T}{\partial x^2} & \frac{\partial^2 T}{\partial x \partial y} & \frac{\partial^2 T}{\partial x \partial z} \\ \frac{\partial^2 T}{\partial x \partial y} & \frac{\partial^2 T}{\partial y^2} & \frac{\partial^2 T}{\partial y \partial z} \\ \frac{\partial^2 T}{\partial x \partial z} & \frac{\partial^2 T}{\partial y \partial z} & \frac{\partial^2 T}{\partial z^2} \end{pmatrix} \quad (3)$$

For those acquainted with linear algebra, notice that H_T is a symmetric matrix.

Critical points and Relative Extrema of a function $T(x, y)$:

⇒ A **critical point** of T is a point (a, b) in the domain of T such that

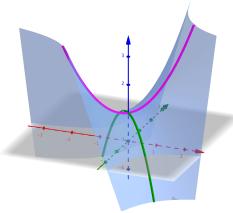
$$\frac{\partial T}{\partial x}(a, b) = 0 \quad \text{and} \quad \frac{\partial T}{\partial y}(a, b) = 0 \quad (4)$$

that is, $\nabla T(a, b) = \mathbf{0}$, or at least one of the partial derivatives does not exist.

⇒ T has a **relative maximum** at (a, b) if $T(x, y) \leq T(a, b)$ for all points (x, y) that are sufficiently close to (a, b) . We say $T(a, b)$ is a **relative maximum value**.

⇒ T has a **relative minimum** at (a, b) with **relative minimum value** $T(a, b)$ if $T(x, y) \geq T(a, b)$ for all points (x, y) that are sufficiently close to (a, b)

⇒ (a, b) is called a **saddle point** if it is a critical point but it is neither a relative minimum nor a relative maximum. For example, the origin $(0, 0)$ is a saddle point for $f(x, y) = xy + 1$



The Second Derivative Test: to classify the relative extrema of a function $T(x, y)$

⇒ Find the critical points of $T(x, y)$ by solving the system of equations

$$\frac{\partial T}{\partial x} = 0 \quad \text{and} \quad \frac{\partial T}{\partial y} = 0 \quad (5)$$

⇒ Define the **discriminant**

$$D(x, y) = T_{xx}T_{yy} - T_{xy}^2 = \det H_T = \det \begin{pmatrix} T_{xx} & T_{xy} \\ T_{yx} & T_{yy} \end{pmatrix} \quad (6)$$

Let (a, b) be a critical point of T .

1. If $D(a, b) > 0$ and $T_{xx}(a, b) < 0$, $T(x, y)$ has a relative maximum at (a, b) . [⊕]
2. If $D(a, b) > 0$ and $T_{xx}(a, b) > 0$, $T(x, y)$ has a relative minimum at (a, b) . [⊖]
3. If $D(a, b) < 0$ then (a, b) is a saddle point.
4. If $D(a, b) = 0$ the test is inconclusive.

Problem 1. Classify the critical points of $f(x, y) = x^2y + \frac{1}{3}y^3 - x^2 - y^2 + 2$ using the second derivative test.

We find the critical points of f . The partial derivatives of f are

$$\begin{aligned}\frac{\partial f}{\partial x} &= 2xy - 2x = 2x(y - 1) \\ \frac{\partial f}{\partial y} &= x^2 + y^2 - 2y = x^2 + y(y - 2)\end{aligned}\tag{7}$$

The critical points must solve the equations $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = 0$, that is,

$$\begin{cases} 2x(y - 1) = 0 \\ x^2 + y(y - 2) = 0 \end{cases}\tag{8}$$

The first equation has solution $x = 0$ or $y = 1$. If we substitute $x = 0$ into the second equation we have

$$y(y - 2) = 0 \implies y = 0 \text{ or } y = 2\tag{9}$$

and so the two critical points corresponding to $x = 0$ are

$$(0, 0), \quad (0, 2)\tag{10}$$

If $y = 1$ we substitute it into the second equation to obtain

$$x^2 - 1 = 0 \implies x = \pm 1\tag{11}$$

and so the corresponding critical points are

$$(1, 1), \quad (-1, 1)\tag{12}$$

Now we proceed to classify these critical points. To compute the discriminant we compute the second order derivatives

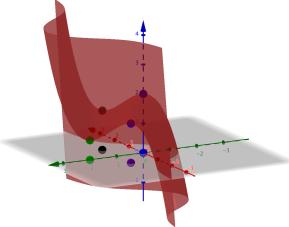
$$\begin{aligned}f_{xx} &= \frac{\partial}{\partial x}(2x(y - 1)) = 2(y - 1) \\ f_{xy} &= \frac{\partial}{\partial y}(2x(y - 1)) = 2x \\ f_{yy} &= \frac{\partial}{\partial y}(x^2 + y^2 - 2y) = 2y - 2 = 2(y - 1)\end{aligned}\tag{13}$$

The discriminant therefore is

$$\begin{aligned}D(x, y) &= f_{xx}f_{yy} - f_{xy}^2 \\ &= 4(y - 1)^2 - 4x^2 \\ &= 4((y - 1)^2 - x^2)\end{aligned}\tag{14}$$

We evaluate the discriminant at each critical point and apply the second derivative test:

Critical Point	$f_{xx}(x, y) = 2(y - 1)$	$D(x, y) = 4 \left((y - 1)^2 - x^2 \right)$	Classification
(0, 0)	-2	4	relative maximum
(0, 2)	2	4	relative minimum
(1, 1)	0	-4	saddle point
(-1, 1)	0	-4	saddle point



Problem 2. Show that the surface $z = xy$ has neither a maximum nor a minimum point.

The partial derivatives are

$$\begin{aligned} \frac{\partial z}{\partial x} &= y \\ \frac{\partial z}{\partial y} &= x \end{aligned} \tag{15}$$

and so the only critical point is the origin, that is, (0, 0). To show that it is a saddle point we compute $D(0, 0)$. The second order partial derivatives are

$$\begin{aligned} \frac{\partial^2 z}{\partial x^2} &= 0 \\ \frac{\partial^2 z}{\partial x \partial y} &= 1 \\ \frac{\partial^2 z}{\partial y^2} &= 0 \end{aligned} \tag{16}$$

and so

$$D(x, y) = f_{xx}f_{yy} - f_{xy}^2 = -1 \tag{17}$$

In particular $D(0, 0) = -1$ which implies by the second derivative test that (0, 0) is a saddle point.

Problem 3. If the product of the sines of the angles of a triangle is a maximum, show that the triangle is equilateral.

Call α, β, γ the three angles of the triangle. The product of the sines of the angles is

$$\sin \alpha \sin \beta \sin \gamma \tag{18}$$

and since these are the angles of a triangle

$$\alpha + \beta + \gamma = \pi \tag{19}$$

which means that we can solve for one the angles in terms of the other two

$$\gamma = \pi - \alpha - \beta \quad (20)$$

and so the function that we are trying to maximize is

$$f(\alpha, \beta) = \sin \alpha \sin \beta \sin(\pi - \alpha - \beta) \quad (21)$$

Since α, β are the angles of a triangle, we can assume that $0 < \alpha < \pi$ and $0 < \beta < \pi$. The partial derivatives of f are

$$\begin{aligned} \frac{\partial f}{\partial \alpha} &= \cos \alpha \sin \beta \sin(\pi - \alpha - \beta) - \sin \alpha \sin \beta \cos(\pi - \alpha - \beta) \\ &= \sin \beta (\cos \alpha \sin(\pi - (\alpha + \beta)) - \sin \alpha \cos(\pi - (\alpha + \beta))) \\ \frac{\partial f}{\partial \beta} &= \sin \alpha \cos \beta \sin(\pi - \alpha - \beta) - \sin \alpha \sin \beta \cos(\pi - \alpha - \beta) \\ &= \sin \alpha (\cos \beta \sin(\pi - (\alpha + \beta)) - \sin \beta \cos(\pi - (\alpha + \beta))) \end{aligned} \quad (22)$$

Before finding the critical points, we also compute the higher order derivatives

$$\begin{aligned} \frac{\partial^2 f}{\partial \alpha^2} &= -2 \sin \beta (\sin \alpha \sin(\pi - (\alpha + \beta)) + \cos \alpha \cos(\pi - (\alpha + \beta))) \\ \frac{\partial}{\partial \alpha} \left(\frac{\partial f}{\partial \beta} \right) &= \cos \alpha (\cos \beta \sin(\pi - (\alpha + \beta)) - \sin \beta \cos(\pi - (\alpha + \beta))) \\ &\quad - \sin \alpha (\cos \beta \cos(\pi - (\alpha + \beta)) + \sin \beta \sin(\pi - (\alpha + \beta))) \\ \frac{\partial^2 f}{\partial \beta^2} &= -2 \sin \alpha (\sin \beta \sin(\pi - (\alpha + \beta)) + \cos \beta \cos(\pi - (\alpha + \beta))) \end{aligned} \quad (23)$$

The critical points must solve the equations $\frac{\partial f}{\partial \alpha} = \frac{\partial f}{\partial \beta} = 0$, that is,

$$\begin{aligned} \sin \beta (\cos \alpha \sin(\pi - (\alpha + \beta)) - \sin \alpha \cos(\pi - (\alpha + \beta))) &= 0 \\ \sin \alpha (\cos \beta \sin(\pi - (\alpha + \beta)) - \sin \beta \cos(\pi - (\alpha + \beta))) &= 0 \end{aligned} \quad (24)$$

Because we are assuming that $0 < \alpha < \pi$ and $0 < \beta < \pi$, $\sin \alpha$ and $\sin \beta$ are never 0, so we must solve the equations

$$\begin{aligned} \cos \alpha \sin(\pi - (\alpha + \beta)) - \sin \alpha \cos(\pi - (\alpha + \beta)) &= 0 \\ \cos \beta \sin(\pi - (\alpha + \beta)) - \sin \beta \cos(\pi - (\alpha + \beta)) &= 0 \end{aligned} \quad (25)$$

If we use the identities $\sin(\theta_1 - \theta_2) = \sin \theta_1 \cos \theta_2 - \cos \theta_1 \sin \theta_2$ and $\cos(\theta_1 - \theta_2) = \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2$ and so the equations are the same as

$$\begin{aligned} \cos \alpha \sin(\alpha + \beta) + \sin \alpha \cos(\alpha + \beta) &= 0 \quad (\bullet) \\ \cos \beta \sin(\alpha + \beta) + \sin \beta \cos(\alpha + \beta) &= 0 \quad (\bullet\bullet) \end{aligned} \quad (26)$$

Multiply the first equation by $\cos \beta$ and the second equation by $\cos \alpha$ to get

$$\begin{aligned} \cos \beta \cos \alpha \sin(\alpha + \beta) + \cos \beta \sin \alpha \cos(\alpha + \beta) &= 0 \quad (\star) \\ \cos \alpha \cos \beta \sin(\alpha + \beta) + \cos \alpha \sin \beta \cos(\alpha + \beta) &= 0 \quad (\star\star) \end{aligned} \tag{27}$$

If we subtract both equations, that is, $(\star) - (\star\star)$ to obtain

$$\begin{aligned} \cos \beta \sin \alpha \cos(\alpha + \beta) - \cos \alpha \sin \beta \cos(\alpha + \beta) &= 0 \\ \Rightarrow (\cos \beta \sin \alpha - \cos \alpha \sin \beta) \cos(\alpha + \beta) &= 0 \\ \Rightarrow \sin(\alpha - \beta) \cos(\alpha + \beta) &= 0 \end{aligned} \tag{28}$$

Therefore, either $\sin(\alpha - \beta) = 0$ or $\cos(\alpha + \beta) = 0$. If $\cos(\alpha + \beta) = 0$ then $\alpha + \beta = \frac{\pi}{2}$ and equations (\bullet) and $(\bullet\bullet)$ become

$$\begin{aligned} \cos \alpha \sin(\alpha + \beta) &= 0 \Rightarrow \cos \alpha = 0 \Rightarrow \alpha = \frac{\pi}{2} \\ \cos \beta \sin(\alpha + \beta) &= 0 \Rightarrow \cos \beta = 0 \Rightarrow \beta = \frac{\pi}{2} \end{aligned} \tag{29}$$

Clearly each of these equations can't be satisfied simultaneously so the case $\cos(\alpha + \beta) = 0$ does not occur. The case $\sin(\alpha - \beta) = 0$ implies that $\alpha = \beta$ and so (\bullet) and $(\bullet\bullet)$ become the same equal to

$$\cos \alpha \sin(2\alpha) + \sin \alpha \cos(2\alpha) = 0 \tag{30}$$

Because $\sin(2\alpha) = 2 \sin \alpha \cos \alpha$ and $\cos(2\alpha) = \cos^2 \alpha - \sin^2 \alpha$ we have the equation

$$\begin{aligned} 2 \sin \alpha \cos^2 \alpha + \sin \alpha \cos^2 \alpha - \sin^3 \alpha &= 0 \\ \Rightarrow 3 \cos^2 \alpha - \sin^2 \alpha &= 0 \\ \Rightarrow 3 - 3 \sin^2 \alpha - \sin^2 \alpha &= 0 \\ \Rightarrow 3 = 4 \sin^2 \alpha & \\ \Rightarrow \frac{3}{4} = \sin^2 \alpha & \\ \Rightarrow \sin \alpha = \pm \frac{\sqrt{3}}{2} & \end{aligned} \tag{31}$$

and since $0 < \alpha < \beta$ we must have $\sin \alpha = \frac{\sqrt{3}}{2}$ which implies that $\alpha = \frac{\pi}{3}$. Therefore we found that

$$\alpha = \beta = \gamma = \frac{\pi}{3} \tag{32}$$

and so the triangle must be equilateral. To show that it maximizes $f(\alpha, \beta)$ we compute

$$f_{xx} \left(\frac{\pi}{3}, \frac{\pi}{3} \right) = -2 \left(\frac{\sqrt{3}}{2} \right) \left(\left(\frac{\sqrt{3}}{2} \right)^2 + \left(\frac{1}{2} \right)^2 \right) \tag{33}$$

which is negative. Similarly

$$\begin{aligned} D\left(\frac{\pi}{3}, \frac{\pi}{3}\right) &= f_{xx}\left(\frac{\pi}{3}, \frac{\pi}{3}\right) f_{yy}\left(\frac{\pi}{3}, \frac{\pi}{3}\right) - \left(f_{xy}\left(\frac{\pi}{3}, \frac{\pi}{3}\right)\right)^2 \\ &= 4\left(\frac{\sqrt{3}}{2}\right)^2 \left(\left(\frac{\sqrt{3}}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)^2 - \left(-\left(\frac{\sqrt{3}}{2}\right)\left(\frac{1}{2}\right)^2 - \left(\frac{\sqrt{3}}{2}\right)^3\right) \end{aligned} \quad (34)$$

which is positive. Therefore, the second derivative test shows that we get a maximum, which is what we wanted to show.

Problem 4. Find the dimensions of a box (top included) which contains a given volume V and uses minimum material (i.e, has minimum surface area)

Let x, y, z be the sides of the box. The volume is

$$V = xyz \quad (35)$$

and the surface area is

$$S = 2xy + 2xz + 2yz \quad (36)$$

Since V is fixed we can solve for z and find

$$z = \frac{V}{xy} \quad (37)$$

Substituting in the formula for S we find the function we want to minimize

$$S(x, y) = 2xy + 2\frac{V}{y} + 2\frac{V}{x} \quad (38)$$

The partial derivatives are

$$\begin{aligned} \frac{\partial S}{\partial x} &= 2y - 2\frac{V}{x^2} \\ \frac{\partial S}{\partial y} &= 2x - 2\frac{V}{y^2} \end{aligned} \quad (39)$$

and the second order derivatives are

$$\begin{aligned} \frac{\partial^2 S}{\partial x^2} &= 4\frac{V}{x^3} \\ \frac{\partial}{\partial y} \left(\frac{\partial S}{\partial x} \right) &= 2 \\ \frac{\partial^2 S}{\partial y^2} &= 4\frac{V}{y^3} \end{aligned} \quad (40)$$

The critical points must satisfy the equations

$$\begin{aligned} 2y - 2\frac{V}{x^2} &= 0 \implies y = \frac{V}{x^2} \implies V = yx^2 (\bullet) \\ 2x - 2\frac{V}{y^2} &= 0 \implies x = \frac{V}{y^2} \implies V = xy^2 (\bullet\bullet) \end{aligned} \quad (41)$$

Setting $(\bullet) = (\bullet\bullet)$ we get $x = y$ and substituting in (\bullet) we $V = x^3$ and so we have the

box must be a cube of sides $x = \sqrt[3]{V}$. To show that it corresponds to a minimum observe that

$$S_{xx} \left(\sqrt[3]{V}, \sqrt[3]{V} \right) = 4 \quad (42)$$

and that

$$\begin{aligned} D(x, y) &= 16 \frac{V^2}{x^3 y^3} - 2 \\ \implies D \left(\sqrt[3]{V}, \sqrt[3]{V} \right) &= 14 \end{aligned} \quad (43)$$

and so by the second derivative test we obtain a relative minimum.

Example 5. Classify the critical points of $f(x, y) = e^{2x+3y} (8x^2 - 6xy + 3y^2)$.

First we compute the partial derivatives of f

$$\frac{\partial f}{\partial x} = 2e^{2x+3y} (8x^2 - 6xy + 3y^2) + e^{2x+3y} (16x - 6y) \quad (44)$$

$$\frac{\partial f}{\partial y} = 3e^{2x+3y} (8x^2 - 6xy + 3y^2) + e^{2x+3y} (-6x + 6y) \quad (45)$$

The critical points must have vanishing partial derivatives, in other words, they must solve the system

$$\begin{cases} 2e^{2x+3y} (8x^2 - 6xy + 3y^2) + e^{2x+3y} (16x - 6y) = 0 \\ 3e^{2x+3y} (8x^2 - 6xy + 3y^2) + e^{2x+3y} (-6x + 6y) = 0 \end{cases} \quad (46)$$

Which is equivalent to the equations

$$\begin{cases} 8x^2 - 6xy + 3y^2 + 8x - 3y = 0 \\ 8x^2 - 6xy + 3y^2 - 2x + 2y = 0 \end{cases} \quad (47)$$

Subtracting both equations we find

$$10x = 5y \quad (48)$$

That is

$$2x = y \quad (49)$$

Now we substitute back in the first equation to find

$$8x^2 - 12x^2 + 12x^2 + 8x - 6x = 0 \quad (50)$$

Notice that this can be factorized as

$$x(8x + 2) = 0 \quad (51)$$

which means that either $x = 0$ or $x = -\frac{1}{4}$. Since $2x = y$ the critical points are $(0, 0)$ and $(-\frac{1}{4}, -\frac{1}{2})$.

To find the Hessian from

$$\begin{aligned}\frac{\partial f}{\partial x} &= 2e^{2x+3y} (8x^2 - 6xy + 3y^2 + 8x - 3y) \\ \frac{\partial f}{\partial y} &= 3e^{2x+3y} (8x^2 - 6xy + 3y^2 - 2x + 2y)\end{aligned}\quad (52)$$

we can compute the second order derivatives

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= 4e^{2x+3y} (8x^2 - 6xy + 3y^2 + 8x - 3y) + 4e^{2x+3y} (8x - 3y + 4) \\ \frac{\partial^2 f}{\partial x \partial y} &= 6e^{2x+3y} (8x^2 - 6xy + 3y^2 + 8x - 3y) + 6e^{2x+3y} (-2x + 2y - 1) \\ \frac{\partial^2 f}{\partial y^2} &= 9e^{2x+3y} (8x^2 - 6xy + 3y^2 - 2x + 2y) + 9e^{2x+3y} (-2x + 2y + \frac{2}{3})\end{aligned}\quad (53)$$

So the Hessian is

$$H_f(x, y) = \begin{pmatrix} 4e^{2x+3y} (8x^2 - 6xy + 3y^2 + 16x - 6y + 4) & 6e^{2x+3y} (8x^2 - 6xy + 3y^2 + 6x - y - 1) \\ 6e^{2x+3y} (8x^2 - 6xy + 3y^2 + 6x - y - 1) & 9e^{2x+3y} (8x^2 - 6xy + 3y^2 - 4x + 4y + \frac{2}{3}) \end{pmatrix} \quad (54)$$

Evaluating at the critical point $(0, 0)$ we have

$$H_f(0, 0) = \begin{pmatrix} 16 & -6 \\ -6 & 6 \end{pmatrix} \quad (55)$$

Since $\det H_f(0, 0) = 60$ and the first entry is positive we conclude that $(0, 0)$ is a relative minimum.

For the critical point $(-\frac{1}{4}, -\frac{1}{2})$

$$H_f\left(-\frac{1}{4}, -\frac{1}{2}\right) = \begin{pmatrix} 4e^{-2} (\frac{7}{2}) & 6e^{-2} (-\frac{3}{2}) \\ 6e^{-2} (-\frac{3}{2}) & 9e^{-2} (\frac{1}{6}) \end{pmatrix} \quad (56)$$

Since $\det H_f(-\frac{1}{4}, -\frac{1}{2}) = e^{-4} (-60) < 0$ we find that this corresponds to a saddle point.

Example 6. Find and classify the absolute maxima and minima of the function $f(x, y) = x^2 - xy + y^2 + 3x - 2y + 1$ defined on the rectangle $-2 \leq x \leq 0, 0 \leq y \leq 1$

Whenever a continuous scalar field is defined on a bounded region which is also closed, it will achieve an absolute maximum and absolute minimum, similar to what happened for functions of one variable defined on a closed interval.

To find these in our case, we work first on the interior of the rectangle, where we can find the critical points in the usual way, and then work with the four boundary pieces of the rectangle separately.

As usual

$$\frac{\partial f}{\partial x} = 2x - y + 3 \quad \frac{\partial f}{\partial y} = -x + 2y - 2 \quad (57)$$

so the critical points must solve

$$\begin{cases} 2x - y + 3 = 0 \\ -x + 2y - 2 = 0 \end{cases} \quad (58)$$

The previous system has solution $(-\frac{4}{3}, \frac{1}{3})$, which does belong to the rectangle. Given

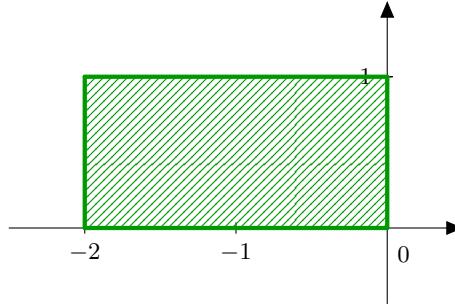


Figure 1: Optimization region

that

$$\frac{\partial^2 f}{\partial x^2} = 2 \quad \frac{\partial^2 f}{\partial x \partial y} = -1 \quad \frac{\partial^2 f}{\partial y^2} = 2 \quad (59)$$

the Hessian of the function is

$$H_f \left(-\frac{4}{3}, \frac{1}{3} \right) = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \quad (60)$$

Since $\det H_f = 3$ and the first entry is negative this critical point is a relative minimum.

It will also be useful to notice that $f \left(-\frac{4}{3}, \frac{1}{3} \right) = -\frac{4}{3}$

Now we analyze each side of the rectangle separately.

1. **Side $-2 \leq x \leq 0 \quad y = 0$:** Here $f(x, 0) = x^2 + 3x + 1$ and now we have turned the problem into a single variable function defined on a closed interval $[-2, 0]$. Just as we did when we were working with the rectangle, we analyze f at the endpoints $-2, 0$ and on the open interval $(-2, 0)$. For the endpoints notice that $f(-2, 0) = -1$ and $f(0, 0) = 1$. On the interval $(-2, 0)$ we have $\frac{\partial f(x, 0)}{\partial x} = 2x + 3$ so the point $x = -\frac{3}{2}, y = 0$ is a candidate for an absolute maximum or minimum for the function. Notice that $f \left(-\frac{3}{2}, 0 \right) = -\frac{5}{4}$
2. **Side $x = 0 \quad 0 \leq y \leq 1$:** Here $f(0, y) = y^2 - 2y + 1$. On the endpoints we have $f(0, 0) = 1$ and $f(0, 1) = 0$. For the interval $(0, 1)$, since $\frac{\partial f(0, y)}{\partial y} = 2y - 2$ we find the critical point $(0, 1)$, which we just computed.
3. **Side $-2 \leq x \leq 0 \quad y = 1$:** Here $f(x, 1) = x^2 + 2x$. Again $f(-2, 1) = 0$ and $\frac{\partial f(x, 1)}{\partial x} = 2x + 2$ so there is a critical point $(-1, 1)$ and $f(-1, 1) = -1$
4. **Side $x = -2 \quad 0 \leq y \leq 1$:** Here $f(-2, y) = y^2 - 1$. Since $\frac{\partial f(-2, y)}{\partial y} = 2y$ we find the critical point $(-2, 0)$ which had already analyzed.

Therefore, our list of candidates for maxima and minima are the points (with corresponding values)

(x, y)	$f(x, y)$
$(-\frac{4}{3}, \frac{1}{3})$	$-\frac{4}{3}$
$(-2, 0)$	-1
$(0, 0)$	1
$(-\frac{3}{2}, 0)$	$-\frac{5}{4}$
$(0, 1)$	0
$(-2, 1)$	0
$(-1, 1)$	-1

From here we can see that $(0, 0)$ corresponds to the absolute maximum while the absolute minimum coincides with the relative minimum and happens at the point $(-\frac{4}{3}, \frac{1}{3})$.

Optimization with and without constraints.

So far we have been dealing with functions of two variables $f(x, y)$. However, our previous discussion carries easily to the case of functions of more variables. For example, if we have a function $f(x, y, z)$, the critical points would be found by solving the system of equations

$$\frac{\partial f}{\partial x} = 0, \quad \frac{\partial f}{\partial y} = 0, \quad \frac{\partial f}{\partial z} = 0 \quad (61)$$

In other words, we are interested in finding the points $P = (a, b, c)$ where

$$\nabla f(P) = \mathbf{0} = (0, 0, 0) \quad (62)$$

Notice that for such a point P , all the directional derivatives vanish, that is,

$$D_{\mathbf{v}} f(P) = \underbrace{\nabla f(P) \cdot \mathbf{v}}_{\mathbf{0}} = \mathbf{0} \cdot \mathbf{v} = 0 \quad (63)$$

Therefore, we can make the following definition of a critical point for a scalar field f :

Critical point of a scalar field (unconstrained case)

Let $f(x, y, z)$ denote a scalar field [for example, $f(x, y, z)$ could represent the temperature at the point (x, y, z)]. Then we say that $P = (a, b, c)$ is a critical point of f if the directional derivatives $D_{\mathbf{v}} f$ of f at P vanish in *all* possible directions, that is,

$$D_{\mathbf{v}} f(P) = 0 \text{ for all direction vectors } \mathbf{v} \quad (64)$$

This occurs if and only if $\nabla f(P) = \mathbf{0}$, so we recover the original definition of critical point as being one where all partial derivatives vanish.

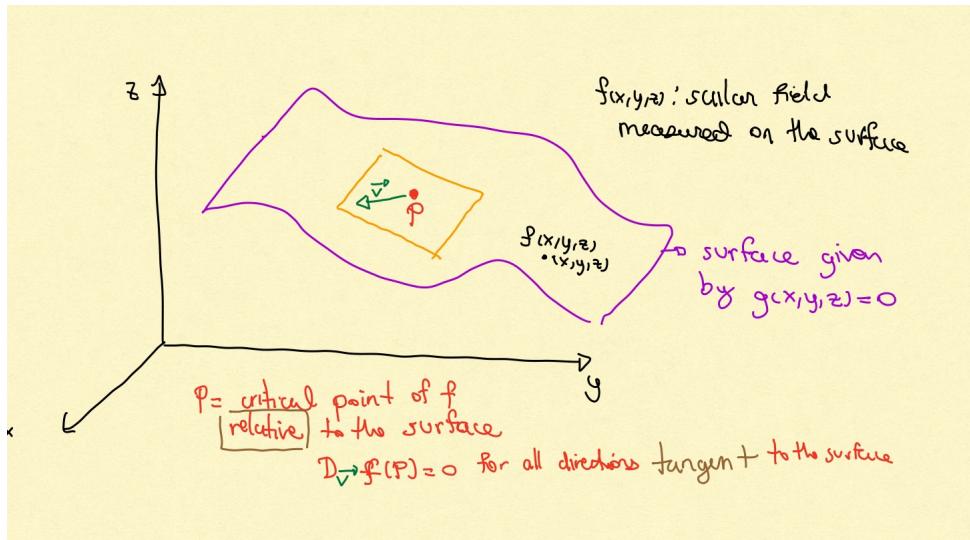
Now suppose you are measuring a scalar field f , but you are no longer allowed to move anywhere you want in space. More concretely, you could be trying to measure the values of f on the surface of a planet. Remember that we think of surfaces as being given by an equation of the form $g(x, y, z) = 0$. For example,

$$g(x, y, z) = x^2 + y^2 + z^2 - R^2 = 0 \quad (65)$$

represents the sphere of radius R centered at the origin, which is a surface.

In this new setup, we are interested in determining the local maxima and minima of f , but relative to the points on the surface determined by the equation $g(x, y, z) = 0$. In other words, if we say that the north pole is a relative minimum for the temperature f , what we mean is that compared to all nearby points *on the surface of the Earth*, then the north pole is a relative minimum. We are no longer interested in comparing the temperature at the north pole against the temperature at points in outer space.

This has implications for what the definition of a critical point of f should be *relative* to the surface $g(x, y, z) = 0$. The reason there is a difference is that we are no longer allowed to move freely in space. In other words, you can only move in ways which are *tangent* to the surface of the Earth.



Remember that at each point of the surface there is a tangent plane, which encodes precisely all the direction vectors which are tangent to the surface at that point. Therefore, we modify our definition of critical point as follows:

Critical point of a scalar field relative to some surface (constrained case)

Let $f(x, y, z)$ denote a scalar field [for example, $f(x, y, z)$ could represent the temperature at the point (x, y, z)]. Suppose that $g(x, y, z) = 0$ represents the equation of a surface and you measure f only at the points which belong to this surface.

Then we say that $P = (a, b, c)$ is a **critical point of f relative to the surface g** if P is a point on the surface and all the directional derivatives $D_{\mathbf{v}} f$ of f at P vanish in *all tangent directions to the surface*, that is,

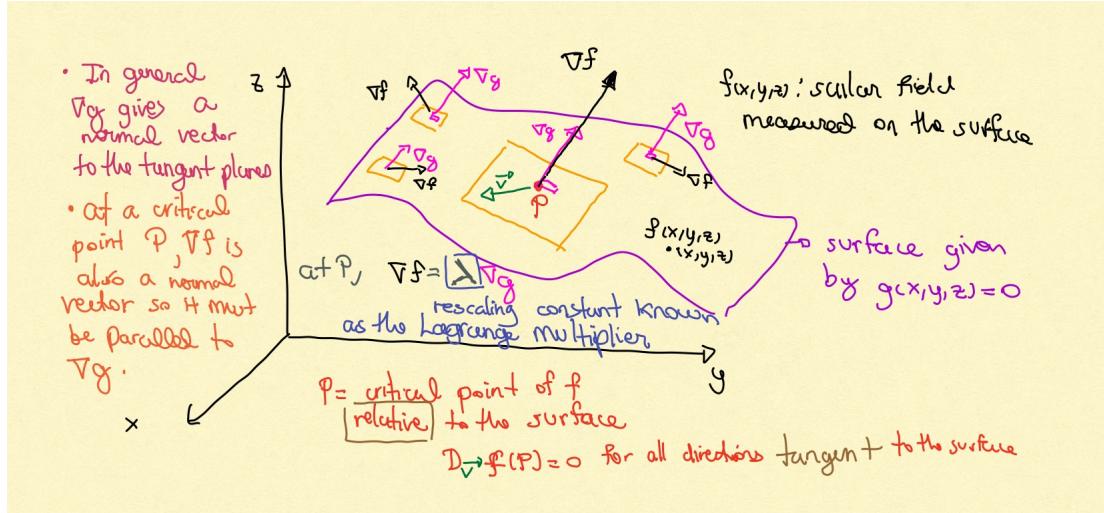
$$D_{\mathbf{v}} f(P) = 0 \text{ for all direction vectors } \mathbf{v} \text{ which belong to the tangent plane at } P \quad (66)$$

Notice that if \mathbf{v} is a (unit) vector which belongs to the tangent plane to the surface through point P , then as before we have that

$$D_{\mathbf{v}}(P) = 0 \text{ is the same as } \nabla f(P) \cdot \mathbf{v} = 0 \quad (67)$$

So $\nabla f(P)$ must be perpendicular (orthogonal) to all the tangent vectors to the surface. But this is precisely the property the normal vector n to a plane satisfies. Moreover, we already know a vector with such a property. Namely, the gradient ∇g of the equation

defining the surface is a normal vector to the tangent planes. This observation is the main idea behind the Lagrange Multipliers method.



Key idea behind Lagrange Multiplier's Method:

Let $f(x, y, z)$ denote a scalar field [for example, $f(x, y, z)$ could represent the temperature at the point (x, y, z)]. Suppose that $g(x, y, z) = 0$ represents the equation of a surface and you measure f only at the points which belong to this surface.

If $P = (a, b, c)$ is a critical point of f relative to the surface g , then at the point P the vector $\nabla f(P)$ is a normal vector for the tangent plane to the surface passing through P . Therefore, it must be parallel to $\nabla g(P)$, which always is a normal vector. In other words

$$\nabla f(P) = \lambda \nabla g \quad \text{for a critical point } P \quad (68)$$

The constant λ is a rescaling factor and it is known as the **Lagrange multiplier**. We think of f as the function we want to optimize, and g as the constraint equation.

Optimization with constraints: Lagrange's multiplier method

To find the critical points of the function $f(x, y, z)$ subject to the constraint $g(x, y, z) = 0$ we must solve the system of equations

$$\begin{cases} \nabla f = \lambda \nabla g \\ g(x, y, z) = 0 \end{cases} \quad (69)$$

When the function $f(x, y, z)$ is subject to the constraints $g(x, y, z) = 0$ and $h(x, y, z) = 0$ we must solve instead

$$\begin{cases} \nabla f = \lambda \nabla g + \mu \nabla h \\ g(x, y, z) = 0 \\ h(x, y, z) = 0 \end{cases} \quad (70)$$

Example 7. Find the critical points of $f(x, y, z) = 2x + 3y + z$ subject to the constraint $g(x, y, z) = 4x^2 + 3y^2 + z^2 - 80$

First we compute the gradients of f and g

$$\nabla f = 2\mathbf{i} + 3\mathbf{j} + \mathbf{k} \quad (71)$$

$$\nabla g = 8x\mathbf{i} + 6y\mathbf{j} + 2z\mathbf{k} \quad (72)$$

From equation 69 we must solve the system

$$\begin{cases} 2 = \lambda 8x \\ 3 = \lambda 6y \\ 1 = \lambda 2z \\ 4x^2 + 3y^2 + z^2 = 80 \end{cases} \quad (73)$$

In order to have a solution we clearly need $\lambda \neq 0$ in which case we find

$$x = \frac{1}{4\lambda} \quad y = \frac{1}{2\lambda} \quad z = \frac{1}{2\lambda} \quad (74)$$

substituting in the last equation

$$\frac{1}{4\lambda^2} + \frac{3}{4\lambda^2} + \frac{1}{4\lambda^2} = 80 \quad (75)$$

which is the same as

$$4\lambda^2 = \frac{5}{80} \quad (76)$$

so the values for λ are

$$\lambda = \pm \frac{1}{8} \quad (77)$$

and the critical points are $(2, 4, 4)$ and $(-2, -4, -4)$.

Example 8. Find the maximum volume of a box of rectangular base that must be inside the ellipsoid $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$.

In this case we must maximize $V(x, y, z) = 8xyz$ subject to the constraint $g(x, y, z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1$. Using the Lagrange multipliers we must solve $\nabla V = \lambda \nabla g$, that is,

$$(8yz, 8xz, 8xy) = \lambda \left(\frac{2x}{a^2}, \frac{2y}{b^2}, \frac{2z}{c^2} \right) \quad (78)$$

From the system of equations

$$\begin{cases} 4yz = \frac{\lambda x}{a^2} \\ 4xz = \frac{\lambda y}{b^2} \\ 4xy = \frac{\lambda z}{c^2} \\ \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \end{cases} \quad (79)$$

we clearly see that $x, y, z \neq 0$ and $\lambda \neq 0$. We can multiply the first three equations to

obtain

$$64x^2y^2z^2 = \frac{\lambda^3 xyz}{a^2 b^2 c^2} \quad (80)$$

in which case

$$\lambda = 4\sqrt[3]{a^2 b^2 c^2 xyz} \quad (81)$$

Dividing the first equation by the second equation we obtain

$$\frac{y}{x} = \frac{b^2 x}{a^2 y} \quad (82)$$

Since $x, y, z > 0$ we conclude that

$$y = \frac{b}{a}x \quad (83)$$

Similarly, we divide the first equation by the third one to obtain

$$z = \frac{c}{a}x \quad (84)$$

Substituting this information in the equation for the ellipsoid we find that

$$3\frac{x^2}{a^2} = 1 \quad (85)$$

Which gives us the values

$$x = \frac{a}{\sqrt{3}} \quad y = \frac{b}{\sqrt{3}} \quad z = \frac{c}{\sqrt{3}} \quad (86)$$

and from this we conclude that the volume must be $\frac{8abc}{3\sqrt{3}}$.

Example 9. Find the absolute maxima and minima of the function $f(x, y, z) = x + y + z$ inside the region $A = \{(x, y, z) : x^2 + y^2 + z^2 \leq 1\}$

Since this set is closed and bounded then an absolute maximum and minimum will be achieved. First we work with the interior of the sphere, that is, the region $x^2 + y^2 + z^2 < 1$. Here it is easy to see that $\nabla f(x, y, z) = \mathbf{0}$ has no solutions.

Therefore we can focus on the region $x^2 + y^2 + z^2 = 1$. In this case we use the constrain $g(x, y, z) = x^2 + y^2 + z^2 - 1$ and so the method of Lagrange multipliers requires us to solve $\nabla f = \lambda \nabla g$, that is

$$(1, 1, 1) = \lambda(2x, 2y, 2z) \quad (87)$$

We obtain the system of equations

$$\begin{cases} 1 = 2\lambda x \\ 1 = 2\lambda y \\ 1 = 2\lambda z \\ x^2 + y^2 + z^2 = 1 \end{cases} \quad (88)$$

Again, $\lambda \neq 0$ so $x = y = z = \frac{1}{2\lambda}$. Substituting in the last equation we find $x = \pm\frac{1}{\sqrt{3}}$ which means that the critical points are $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$ and $(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})$.

Since the maximum and minimum must be achieved and there are only two candidates $\left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$ will correspond to the absolute maximum while $\left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)$ will correspond to the absolute minimum.

Problem 10. Find the dimensions of a box (top included) which contains a given volume V and uses minimum material (i.e, has minimum surface area) using the Lagrange multipliers method.

Notice that we solved this problem before, but now we are going to use the Lagrange multipliers method. The function we want to minimize is the surface area [called before S]

$$f(x, y, z) = 2xy + 2xz + 2yz \quad (89)$$

subject to the constraint equation

$$g(x, y, z) = xyz - V = 0 \quad (90)$$

According to the Lagrange multiplier method, we must solve

$$\begin{cases} \nabla f = \lambda \nabla g \\ g = 0 \end{cases} \quad (91)$$

which is equivalent to

$$\begin{cases} (2y + 2z, 2x + 2z, 2x + 2y) = \lambda(yz, xz, xy) \\ xyz = V \end{cases} \quad (92)$$

We obtain the system of equations

$$\begin{cases} 2y + 2z = \lambda yz & (1) \\ 2x + 2z = \lambda xz & (2) \\ 2x + 2y = \lambda xy & (3) \\ xyz = V & (4) \end{cases} \quad (93)$$

To solve this system, multiply equation (1) by x , equation (2) by y , and equation (3) by z , in order to obtain

$$\begin{cases} 2xy + 2xz = \lambda xyz & (A) \\ 2xy + 2yz = \lambda xyz & (B) \\ 2xz + 2yz = \lambda xyz & (C) \\ xyz = V & (4) \end{cases} \quad (94)$$

Now use the last equation and substitute in the other three to obtain

$$\begin{cases} 2xy + 2xz = \lambda V & (A) \\ 2xy + 2yz = \lambda V & (B) \\ 2xz + 2yz = \lambda V & (C) \end{cases} \quad (95)$$

The left hand side of (A) must now equal the left hand side of (B), which means

$$\begin{aligned} 2xy + 2xz &= 2xy + 2yz \\ \implies 2xz &= 2yz \\ \implies \boxed{x = y} \end{aligned}$$

Notice that here we do not need to worry about the case where $z = 0$, since that is not physically interesting. Likewise, equating (A) with (C) you will find that

$$\boxed{y = z} \quad (96)$$

So we conclude that

$$\boxed{x = y = z} \quad (97)$$

We can substitute back this information in equation (4) to find that

$$x^3 = V \quad (98)$$

or

$$\boxed{x = y = z = V^{1/3}} \quad (99)$$

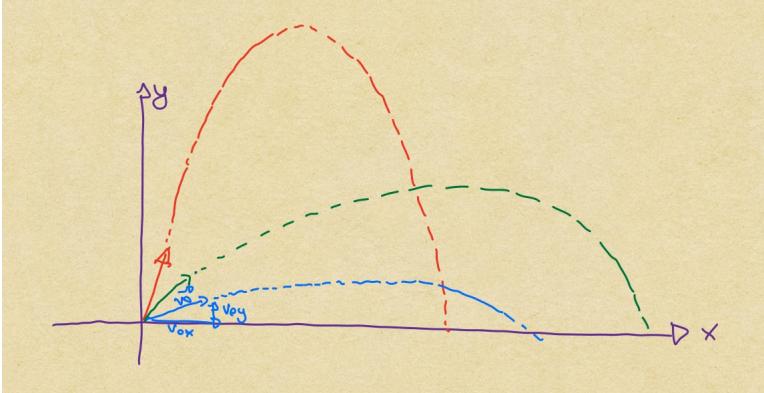
which is what we found before! The main differences to observe are:

- ☞ We treated all the equations in a symmetric fashion. That is, we did not eliminate one of the variables in terms of the others, as we had done originally when we wrote $z = \frac{V}{xy}$.
 - ☞ In this particular problem we did not need to find the value of λ before obtaining the values for x, y, z . Sometimes it will be necessary to find λ first before being able to completely solve the problem.
 - ☞ We found a critical point, but strictly speaking we do not know if this yields a local max, local min, or neither. It is possible to classify critical points using the Lagrange multipliers method, but we will avoid doing this since it is more complicated.
-

Physical Interpretation of the Lagrange Multipliers [optional]

So far it is not clear if the Lagrange multiplier λ has any particular meaning. To discuss the physical interpretation of the Lagrange multiplier, consider the following experiment.

Suppose you have a cannon and start launching balls from it with different initial velocities \mathbf{v}_0 . Depending on the initial velocity chosen, the ball will hit the floor at different distances from the cannon, in other words, the range of each projectile will depend on the initial velocity.



In fact, if you write the initial velocity in terms of its components in the x and y directions, that is,

$$\mathbf{v}_0 = (v_{0x}, v_{0y}) \quad (100)$$

Then it is a simple physics exercise (assuming there is no friction, etc) to show that the range R of the projectile is given as

$$R(v_{0x}, v_{0y}) = \frac{2}{g} v_{0x} v_{0y} \quad (101)$$

where g is the acceleration due to gravity (a constant).

Now, if we were interested in *maximizing* the range R , then clearly a constraint needs to be imposed, since otherwise you could keep launching the projectiles with arbitrarily large velocities, thereby increasing the range indefinitely. Since the total energy is conserved in this problem, and it must equal the initial kinetic energy of the balls, it is reasonable to impose the condition that we will only launch balls with a particular kinetic energy. That is, our constraint equation is

$$C(v_{0x}, v_{0y}) = \frac{1}{2} m(v_{0x}^2 + v_{0y}^2) - E = 0 \quad (102)$$

where E is the kinetic energy. Therefore, we have naturally found a Lagrange multiplier problem: we want to optimize R subject to the constraint $C = 0$. According to Lagrange's method, we must solve

$$\begin{cases} \nabla R = \lambda \nabla C \\ C = 0 \end{cases} \quad (103)$$

This is equivalent to

$$\begin{cases} \frac{2}{g} v_{0y} = \lambda m v_{0x} & (1) \\ \frac{2}{g} v_{0x} = \lambda m v_{0y} & (2) \\ \frac{1}{2} m(v_{0x}^2 + v_{0y}^2) = E & (3) \end{cases} \quad (104)$$

Similar to the box problem, in order to solve this system we multiply equation (1) by

v_{0y} , equation (2) by v_{0x} in order to obtain

$$\begin{cases} \frac{2}{g}v_{0y}^2 = \lambda m v_{0x} v_{0y} & (A) \\ \frac{2}{g}v_{0x}^2 = \lambda m v_{0y} v_{0x} & (B) \\ \frac{1}{2}m(v_{0x}^2 + v_{0y}^2) = E & (3) \end{cases} \quad (105)$$

We equate the left hand sides of (A) and (B) to obtain

$$\frac{2}{g}v_{0y}^2 = \frac{2}{g}v_{0x}^2 \quad (106)$$

which in this case gives

$$v_{0y} = v_{0x} \quad (107)$$

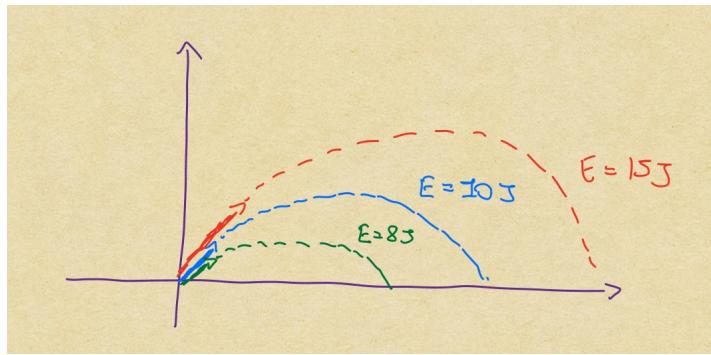
Notice that the case $v_{0y} = -v_{0x}$ is also possible but not interesting in this particular situation. So we just found that the initial velocity which maximizes the range is the one where the ball is launched at a 45° angle. Using equation (3) we can determine v_{0x} in terms of E as

$$v_{0x} = v_{0y} = \sqrt{\frac{E}{m}} \quad (108)$$

Again, in this case we did not find the value of λ at all, but using equation (A) and the fact that $v_{0x} = v_{0y}$ we can see that

$$\lambda = \frac{2}{mg} \quad (109)$$

In order to interpret this particular expression for λ , we must do a slightly different experiment.



Now that we know that for a given value of E , the velocity that maximizes the range R is one where $v_{0x} = v_{0y}$, we can do an experiment where we launch projectiles with *different* values of E , but all being launched at an 45° angle, which is the angle which maximizes the range.

Since we are fixing the angle, the range now depends solely on the energy E chosen for each launch. In fact, the formulas $v_{0x} = v_{0y} = \sqrt{\frac{E}{m}}$ and $R(v_{0x}, v_{0y}) = \frac{2}{g}v_{0x}v_{0y}$ we can

think of R as given in terms of E :

$$R(E) = \frac{2}{g} \frac{E}{m} \quad (110)$$

In particular, notice that

$$\frac{dR}{dE} = \frac{2}{gm} = \lambda \quad (111)$$

This is not a coincide, in fact, we can think of the Lagrange multiplier in the following way:

Physical Meaning of the Lagrange Multiplier:

The Lagrange multiplier is the rate of change of the optimized quantity R with respect to value of the constraint parameter E .

Lagrange Multipliers and Absolute Max/Min Problems

Suppose you want to find the absolute maxima and minima of a function f on some bounded region R . Moreover, suppose that the boundary of the region R is given by some equation $g = 0$, which is interpreted as the constraint equation.

1. Find the critical points of the function f , that is, the points where $\nabla f = \mathbf{0}$. Make a list with the critical points which belong to the region R .
2. Find the critical points given by the Lagrange multiplier problem, that is, solve $\nabla f = \lambda \nabla g$.
3. Make a table with the critical points from steps 1. and 2., and evaluate f at all of these points. The absolute max and min will correspond the largest and smallest values of f that appear on this table.

Example 11. Find the absolute maxima and minima of the function $f(x, y) = 8x^2 - 2y^2$ when restricted to the region R on the xy plane given by the inequality $x^2 + y^2 \leq 1$.

We follow the steps:

1. The gradient of f is $\nabla f = (16x, -4y)$ so $\nabla f = \mathbf{0}$ only when $x = 0$ and $y = 0$. Therefore, $(0, 0)$ is the only (ordinary) critical point. Moreover, notice that $(0, 0)$ belongs to the region R so we will keep it.
2. We will solve $\nabla f = \lambda \nabla g$, where $g(x, y) = x^2 + y^2 - 1$ is the equation of the circle. Since $\nabla g = (2x, 2y)$, we must solve

$$\begin{cases} 16x = 2\lambda x \\ -4y = 2\lambda y \\ x^2 + y^2 = 1 \end{cases} \quad (112)$$

The first equation is the same as $2x(8 - \lambda) = 0$. **Case $x = 0$:** then the third equation implies that $y = \pm 1$ and so the critical points are $(0, 1)$ and $(0, -1)$. **Case $\lambda = 8$:** the second equation becomes $-4y = 16y$ so $y = 0$, and the third equation implies that $x = \pm 1$. Therefore, the critical points are $(1, 0)$ and $(-1, 0)$.

3. The table with all the critical points is

(x, y)	$f(x, y) = 8x^2 - 2y^2$	
$(0, 0)$	0	
$(0, 1)$	-2	
$(0, -1)$	-2	
$(1, 0)$	8	
$(-1, 0)$	8	

So $(0, 1)$ and $(0, -1)$ yield the absolute minimum, which is -2 , while $(1, 0)$ and $(-1, 0)$ yield the absolute maximum, which is 8 . Notice that in this case it is not necessary to apply the second derivative test to $(0, 0)$, since that would only say whether or not it yields a *relative* max, *relative* min, or a saddle point, which is different from being an absolute max or min.

A First Look at the Decision Tree Classifiers

Introduction

- Decision tree learning is one of the most widely used techniques for classification.
 - Its classification accuracy is competitive with other methods, and
 - it is very efficient.
- The classification model is a tree, called **decision tree**.
- C4.5 by Ross Quinlan is perhaps the best known system and the codes are freely available from internet.

The loan data (reproduced)

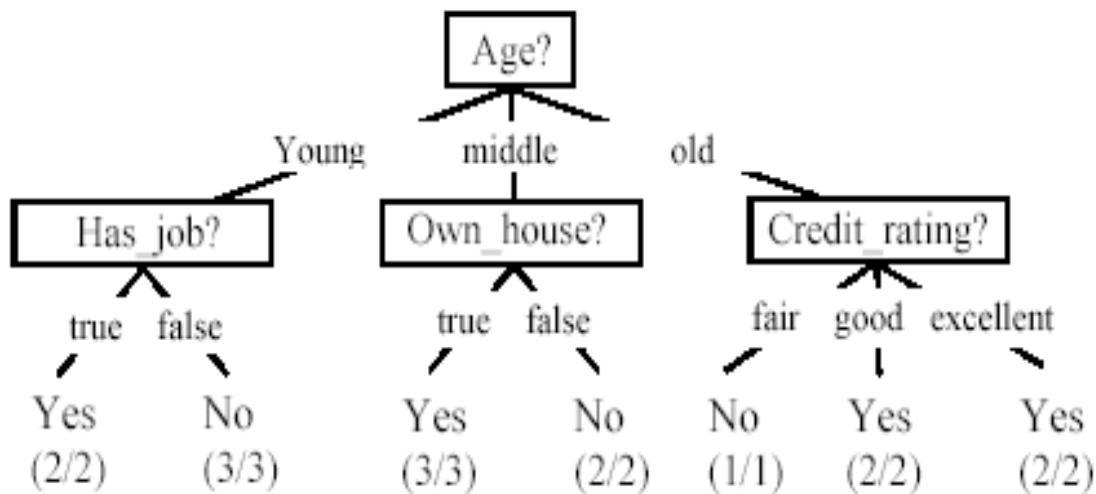
Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

3

A decision tree from the loan data

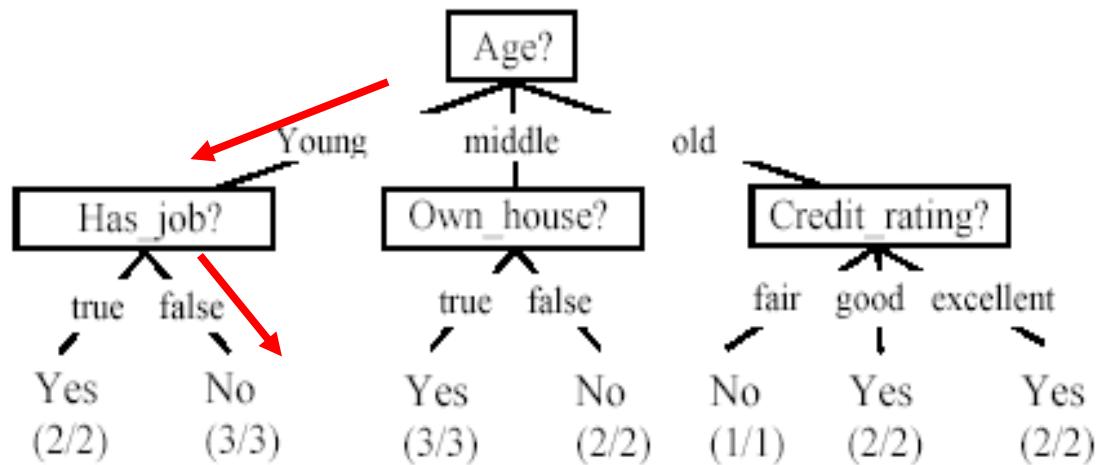
Decision nodes and leaf nodes (classes)



4

Use the decision tree

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?



5

Is the decision tree unique?

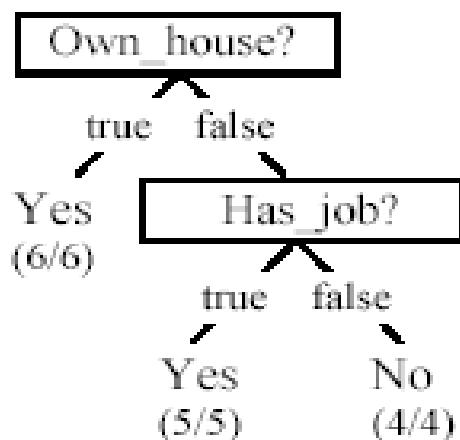
No. Here is a simpler tree.

We want **smaller tree** and **accurate tree**.

Easy to understand and perform better.

Finding the best tree is NP-hard.

All current tree building algorithms are heuristic algorithms

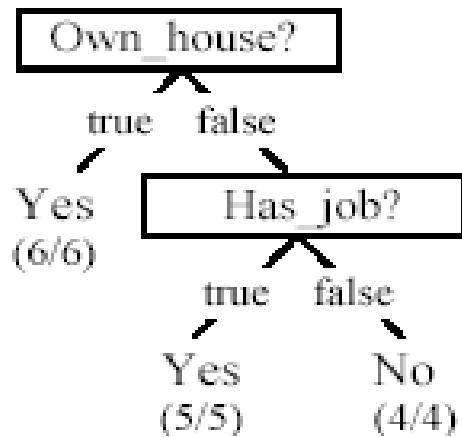


6

From a decision tree to a set of rules

A decision tree can be converted to a set of rules

Each path from the root to a leaf is a rule.



Own_house = true \rightarrow Class = Yes [sup=6/15, conf=6/6]

Own_house = false, Has_job = true \rightarrow Class = Yes [sup=5/15, conf=5/5]

Own_house = false, Has_job = false \rightarrow Class = No [sup=4/15, conf=4/4]

7

Algorithm for decision tree learning

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
 - Assume attributes are categorical now (continuous attributes can be handled too)
 - Tree is constructed in a **top-down recursive manner**
 - At start, all the training examples are at the root
 - Examples are partitioned recursively based on selected attributes
 - Attributes are selected on the basis of an impurity function (e.g., **information gain**)
- Conditions for stopping partitioning
 - All examples for a given node belong to the same class
 - There are no remaining attributes for further partitioning – majority class is the leaf
 - There are no examples left

8

Decision tree learning algorithm

```
. Algorithm decisionTree( $D, A, T$ )
1   if  $D$  contains only training examples of the same class  $c_f \in C$  then
2       make  $T$  a leaf node labeled with class  $c_f$ ;
3   elseif  $A = \emptyset$  then
4       make  $T$  a leaf node labeled with  $c_j$ , which is the most frequent class in  $D$ 
5   else //  $D$  contains examples belonging to a mixture of classes. We select a single
6       // attribute to partition  $D$  into subsets so that each subset is purer
7        $p_0 = \text{impurityEval-1}(D)$ ;
8       for each attribute  $A_i \in \{A_1, A_2, \dots, A_k\}$  do
9            $p_i = \text{impurityEval-2}(A_i, D)$ 
10      end
11      Select  $A_g \in \{A_1, A_2, \dots, A_k\}$  that gives the biggest impurity reduction,
12          computed using  $p_0 - p_i$ ;
13      if  $p_0 - p_g < \text{threshold}$  then //  $A_g$  does not significantly reduce impurity  $p_0$ 
14          make  $T$  a leaf node labeled with  $c_j$ , the most frequent class in  $D$ .
15      else //  $A_g$  is able to reduce impurity  $p_0$ 
16          Make  $T$  a decision node on  $A_g$ ;
17          Let the possible values of  $A_g$  be  $v_1, v_2, \dots, v_m$ . Partition  $D$  into  $m$ 
18          disjoint subsets  $D_1, D_2, \dots, D_m$  based on the  $m$  values of  $A_g$ .
19          for each  $D_j$  in  $\{D_1, D_2, \dots, D_m\}$  do
20              if  $D_j \neq \emptyset$  then
21                  create a branch (edge) node  $T_j$  for  $v_j$  as a child node of  $T$ ;
22                  decisionTree( $D_j, A - \{A_g\}, T_j$ ) //  $A_g$  is removed
23              end
24          end
25      end
26  end
```

9

Choose an attribute to partition data

- The **key** to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
 - A subset of data is **pure** if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum **Information Gain** or **Gain Ratio** based on information theory.

The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

11

Two possible roots, which is better?

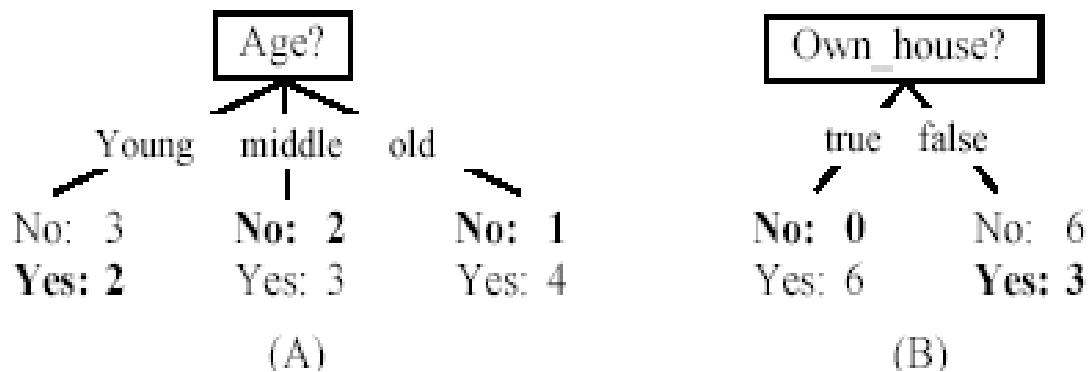


Fig. (B) seems to be better.

12

Information theory

- **Information theory** provides a mathematical basis for measuring the information content.
- To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.
 - If one already has a good guess about the answer, then the actual answer is less informative.
 - If one already knows that the coin is rigged so that it will come with heads with probability 0.99, then a message (advanced information) about the actual outcome of a flip is worth less than it would be for a honest coin (50-50).

13

Information theory (cont ...)

- For a fair (honest) coin, you have no information, and you are willing to pay more (say in terms of \$) for advanced information - less you know, the more valuable the information.
- **Information theory** uses this same intuition, but instead of measuring the value for information in dollars, it measures information contents in **bits**.
- One bit of information is enough to answer a yes/no question about which one has no idea, such as the flip of a fair coin

14

Information theory: Entropy measure

- The entropy formula,

$$\text{entropy}(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1,$$

- $\Pr(c_j)$ is the probability of class c_j in data set D
- We use entropy as a **measure of impurity or disorder** of data set D . (Or, a measure of information in a tree)

15

Entropy measure: let us get a feeling

1. The data set D has 50% positive examples ($\Pr(\text{positive}) = 0.5$) and 50% negative examples ($\Pr(\text{negative}) = 0.5$).

$$\text{entropy}(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples ($\Pr(\text{positive}) = 0.2$) and 80% negative examples ($\Pr(\text{negative}) = 0.8$).

$$\text{entropy}(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples ($\Pr(\text{positive}) = 1$) and no negative examples, ($\Pr(\text{negative}) = 0$).

$$\text{entropy}(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

16

Information gain

- Given a set of examples D , we first compute its entropy:

$$\text{entropy}(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

- If we make attribute A_i , with v values, the root of the current tree, this will partition D into v subsets D_1, D_2, \dots, D_v . The expected entropy if A_i is used as the current root:

$$\text{entropy}_{A_i}(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{entropy}(D_j)$$

17

Information gain (cont ...)

- Information gained by selecting attribute A_i to branch or to partition the data is

$$\text{gain}(D, A_i) = \text{entropy}(D) - \text{entropy}_{A_i}(D)$$

- We choose the attribute with the highest gain to branch/split the current tree.

18

An example

$$\text{entropy}(D) = -\frac{6}{15} \times \log_2 \frac{6}{15} - \frac{9}{15} \times \log_2 \frac{9}{15} = 0.971$$

$$\begin{aligned}\text{entropy}_{\text{Own_house}}(D) &= -\frac{6}{15} \times \text{entropy}(D_1) - \frac{9}{15} \times \text{entropy}(D_2) \\ &= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918 \\ &= 0.551\end{aligned}$$

$$\begin{aligned}\text{entropy}_{\text{Age}}(D) &= -\frac{5}{15} \times \text{entropy}(D_1) - \frac{5}{15} \times \text{entropy}(D_2) - \frac{5}{15} \times \text{entropy}(D_3) \\ &= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722 \\ &= 0.888\end{aligned}$$

Own_house is the best choice for the root.

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Age	Yes	No	entropy(Di)
young	2	3	0.971
middle	3	2	0.971
old	4	1	0.722

$$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$$

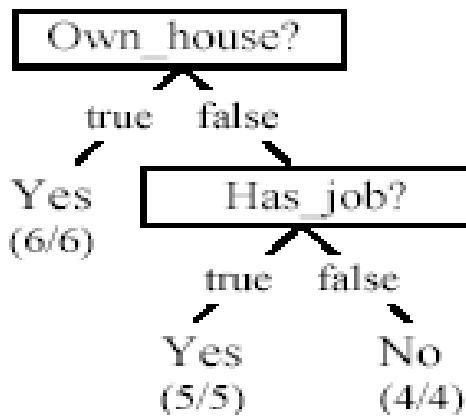
$$gain(D, \text{Own_house}) = 0.971 - 0.551 = 0.420$$

$$gain(D, \text{Has_Job}) = 0.971 - 0.647 = 0.324$$

$$gain(D, \text{Credit_Rating}) = 0.971 - 0.608 = 0.363$$

19

We build the final tree



We can use information gain ratio to evaluate the impurity as well

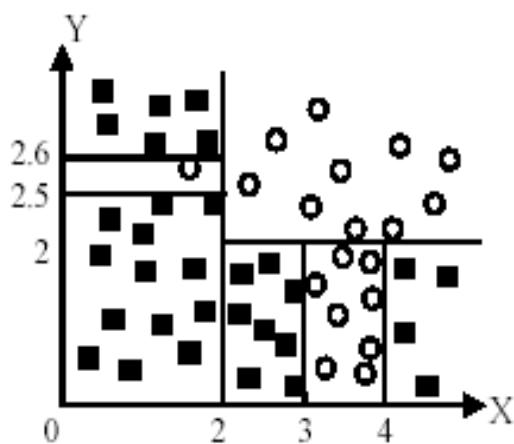
20

Handling continuous attributes

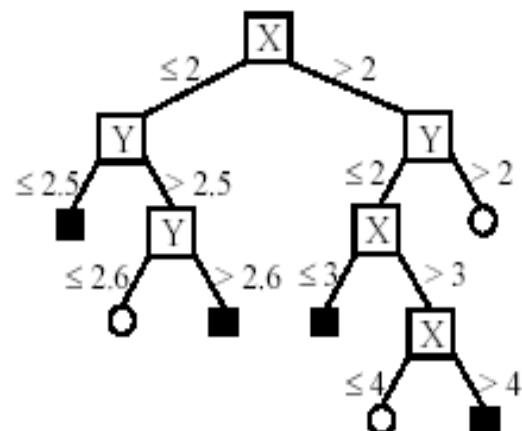
- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
 - Use information gain or gain ratio again
 - Sort all the values of an continuous attribute in increasing order $\{v_1, v_2, \dots, v_r\}$,
 - One possible threshold between two adjacent values v_i and v_{i+1} . Try all possible thresholds and find the one that maximizes the gain (or gain ratio).

21

An example in a continuous space



(A) A partition of the data space



(B). The decision tree

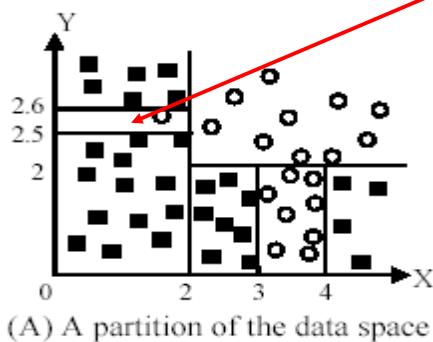
22

Avoid overfitting in classification

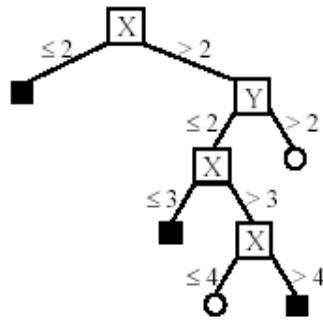
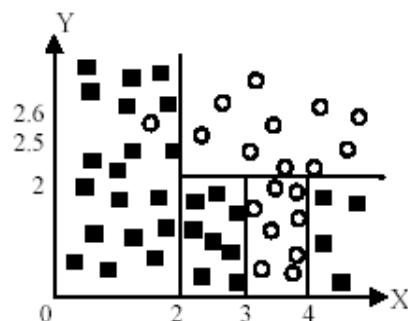
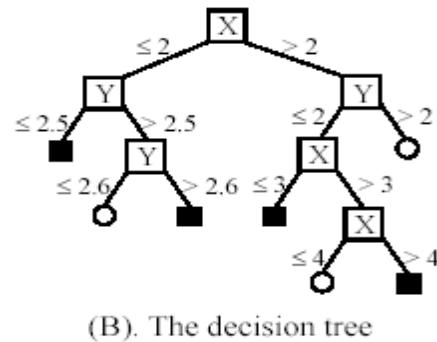
- **Overfitting:** A tree may overfit the training data
 - Good accuracy on training data but poor on test data
 - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Two approaches to avoid overfitting
 - **Pre-pruning:** Halt tree construction early
 - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
 - **Post-pruning:** Remove branches or sub-trees from a “fully grown” tree.
 - This method is commonly used. C4.5 uses a statistical method to estimate the errors at each node for pruning.
 - A validation set may be used for pruning as well.

23

An example



Likely to overfit the data



24

Other issues in decision tree learning

- From tree to rules, and rule pruning
- Handling of miss values
- Handing skewed distributions
- Handling attributes and classes with different costs.
- Attribute construction
- Etc.

Eigenvalues & Eigenvectors

- **Eigenvectors** (for a square $m \times m$ matrix S)

$$S\mathbf{v} = \lambda\mathbf{v}$$

(right) eigenvector eigenvalue
 $\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$ $\lambda \in \mathbb{R}$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- How many eigenvalues are there at most?

$$S\mathbf{v} = \lambda\mathbf{v} \iff (S - \lambda I)\mathbf{v} = 0$$

only has a non-zero solution if $|S - \lambda I| = 0$

this is a m -th order equation in λ which can have **at most m distinct solutions** (roots of the characteristic polynomial) - can be complex even though S is real.

Matrix-vector multiplication

$$S = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{has eigenvalues 3, 2, 0 with corresponding eigenvectors}$$

$$v_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

On each eigenvector, S acts as a multiple of the identity matrix: but as a different multiple on each.

Any vector (say $x = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$) can be viewed as a combination of the eigenvectors: $x = 2v_1 + 4v_2 + 6v_3$

Matrix vector multiplication

- Thus a matrix-vector multiplication such as Sx (S , x as in the previous slide) can be rewritten in terms of the eigenvalues/vectors:

$$Sx = S(2v_1 + 4v_2 + 6v_3)$$

$$Sx = 2Sv_1 + 4Sv_2 + 6Sv_3 = 2\lambda_1 v_1 + 4\lambda_2 v_2 + 6\lambda_3 v_3$$

- Even though x is an arbitrary vector, the action of S on x is determined by the eigenvalues/vectors.
- Suggestion: the effect of “small” eigenvalues is small.

Eigenvalues & Eigenvectors

For symmetric matrices, eigenvectors for distinct eigenvalues are **orthogonal**

$$Sv_{\{1,2\}} = \lambda_{\{1,2\}} v_{\{1,2\}}, \text{ and } \lambda_1 \neq \lambda_2 \Rightarrow v_1 \bullet v_2 = 0$$

All eigenvalues of a real symmetric matrix are **real**.

for complex λ , if $|S - \lambda I| = 0$ and $S = S^T \Rightarrow \lambda \in \mathbb{R}$

All eigenvalues of a **positive semidefinite** matrix are **non-negative**

$$\overbrace{\forall w \in \mathbb{R}^n, w^T S w \geq 0}^{\text{then if } Sv = \lambda v \Rightarrow \lambda \geq 0}$$

Example

- Let

$$S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad \text{Real, symmetric.}$$

- Then

$$S - \lambda I = \begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix} \Rightarrow (2 - \lambda)^2 - 1 = 0.$$

- The eigenvalues are 1 and 3 (nonnegative, real).
- The eigenvectors are orthogonal (and real):

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Plug in these values
and solve for
eigenvectors.

Eigen/diagonal Decomposition

- Let $S \in \mathbb{R}^{m \times m}$ be a **square** matrix with **m linearly independent eigenvectors** (a “non-defective” matrix)
- Theorem:** Exists an **eigen decomposition**

$$S = U \Lambda U^{-1}$$

diagonal
Unique
for
distinct
eigen-
values

- (cf. matrix diagonalization theorem)
- Columns of U are **eigenvectors** of S
- Diagonal elements of Λ are **eigenvalues** of S

$$\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

Diagonal decomposition: why/how

Let \mathbf{U} have the eigenvectors as columns: $U = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$

Then, $\mathbf{S}\mathbf{U}$ can be written

$$SU = S \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} = \begin{bmatrix} \lambda_1 v_1 & \dots & \lambda_n v_n \end{bmatrix} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$$

Thus $\mathbf{S}\mathbf{U} = \mathbf{U}\Lambda$, or $\mathbf{U}^{-1}\mathbf{S}\mathbf{U} = \Lambda$

And $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^{-1}$.

Eigendecomposition - example

Recall $S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \lambda_1 = 1, \lambda_2 = 3.$

The eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ form $U = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

Inverting, we have

$$U^{-1} = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix} \quad \text{Recall } UU^{-1} = 1.$$

Then, $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

Example continued

Let's divide \mathbf{U} (and multiply \mathbf{U}^{-1}) by $\sqrt{2}$

Then, $\mathbf{S} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$

\mathbf{Q}

Λ

$(\mathbf{Q}^{-1} = \mathbf{Q}^T)$

Why? Stay tuned ...

Symmetric Eigen Decomposition

- If $\mathbf{S} \in \mathbb{R}^{m \times m}$ is a **symmetric** matrix:
- **Theorem:** Exists a (unique) **eigen decomposition**
- where \mathbf{Q} is **orthogonal**: $S = Q\Lambda Q^T$
 - $\mathbf{Q}^{-1} = \mathbf{Q}^T$
 - Columns of \mathbf{Q} are normalized eigenvectors
 - Columns are orthogonal.
 - (everything is real)

Exercise

- Examine the symmetric eigen decomposition, if any, for each of the following matrices:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \\ -2 & 3 \end{bmatrix} \quad \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$$

Singular Value Decomposition

For an $m \times n$ matrix \mathbf{A} of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$A = U\Sigma V^T$$

$m \times m$ $m \times n$ $V \text{ is } n \times n$

The columns of \mathbf{U} are orthogonal eigenvectors of \mathbf{AA}^T .

The columns of \mathbf{V} are orthogonal eigenvectors of $\mathbf{A}^T\mathbf{A}$.

Eigenvalues $\lambda_1 \dots \lambda_r$ of \mathbf{AA}^T are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

$$\sigma_i = \sqrt{\lambda_i}$$
$$\Sigma = \text{diag}(\sigma_1 \dots \sigma_r)$$

← **Singular values.**

Singular Value Decomposition

- Illustration of SVD dimensions and sparseness

$$\underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_A = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{V^T}$$
$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_A = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}$$

SVD example

Let $A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

Thus $m=3, n=2$. Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Typically, the singular values arranged in decreasing order.

Singular Value Decomposition

A is $m \times n$

$A = (\text{orthogonal}) (\text{diagonal}) (\text{orthogonal})$

$$A = U \Sigma V^T$$

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} = \begin{bmatrix} & & & & \sigma_1 \\ & & & & \ddots \\ & & & & \sigma_r \\ & & & & \\ & & & & V^T \end{bmatrix}$$

U

8)

$$A = U \Sigma V^T \rightarrow A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

Applications of the SVD

G. Strang, Linear Algebra and its Applications p444

I. Image processing

Suppose a satellite takes a picture, and wants to send it to earth. The picture may contain 1000 by 1000 "pixels"—little squares each with a definite color. We can code the colors, in a range between black and white, and send back 1,000,000 numbers



picture

$$\begin{bmatrix} x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \end{bmatrix}$$

matrix

Applications of the SVD

G. Strang, Linear Algebra and its Applications p444

1. Image processing

It is better to find the essential information in the 1000 by 1000 matrix, and send only that.

Suppose we know the SVD.

The key is in the singular values.

Typically, some are significant and others are extremely small.

If we keep 60 and throw away 940, then we send only the corresponding 60 columns of U , and V .

The other 940 columns are multiplied by small singular values that are being ignored. In fact, we can do the matrix multiplication as columns times rows:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$



If only 60 terms are kept, we send 60 times 2000 numbers instead of a million.

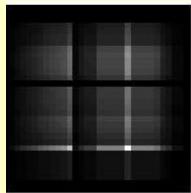
Applications of the SVD

the SVD of a 32-times-32 digital image A is computed

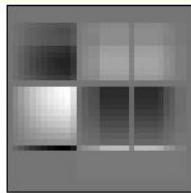
the activities are lead by Prof. Per Christian Hansen.



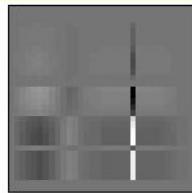
$$\sigma_1 u_1 v_1^T$$



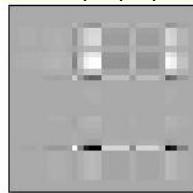
$$\sigma_2 u_2 v_2^T$$



$$\sigma_3 u_3 v_3^T$$



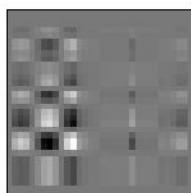
$$\sigma_4 u_4 v_4^T$$



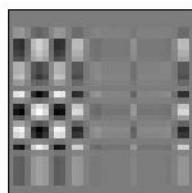
$$\sigma_5 u_5 v_5^T$$



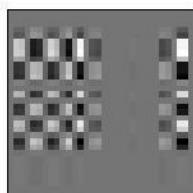
$$\sigma_6 u_6 v_6^T$$



$$\sigma_7 u_7 v_7^T$$



$$\sigma_8 u_8 v_8^T$$

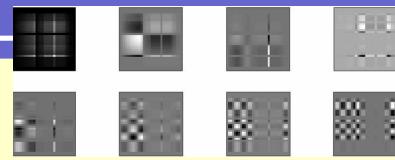


Applications of the SVD



$$A_s = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_s u_s v_s^T$$

$$s \leq r$$

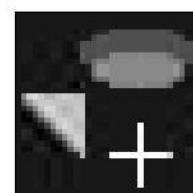
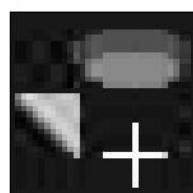
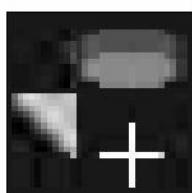
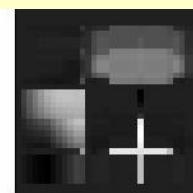
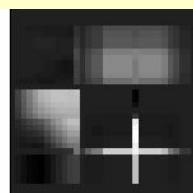
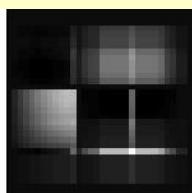
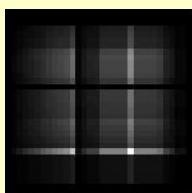


A_1

A_2

A_3

A_4



A_5

A_6

A_7

A_8

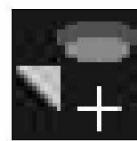
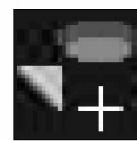
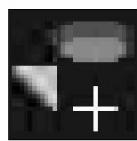
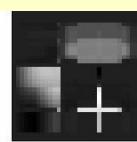
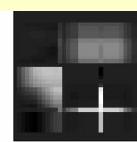
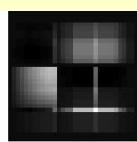
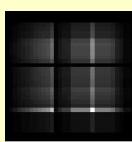
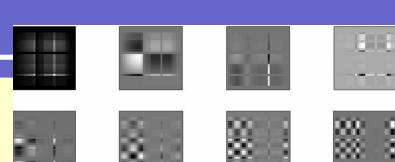
G. Strang "at first you see nothing, and suddenly you recognize everything."

Applications of the SVD



$$A_s = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_s u_s v_s^T$$

$$s \leq r$$



How to compute SVD (by hand)

Eigenvalue Decomposition

If A is real n -by- n matrix, then

$$A = S \Sigma S^{-1}$$

$S = [S_1, S_2, \dots, S_m]$ eigenvectors $\Sigma = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$

If A is symmetric

$$A = Q \Sigma Q^T \quad Q^T Q = I$$

Example:

$$A \quad Q \quad \Sigma \quad Q^T$$

$$\begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

How to compute SVD (by hand)

$$\left. \begin{array}{l} A = U \Sigma V^T \\ A^T = V \Sigma^T U^T \end{array} \right\} \rightarrow AA^T = (U \Sigma V^T)(V \Sigma^T U^T)$$

$$AA^T = U \Sigma \Sigma^T U^T \quad AA^T \text{ is symmetric}$$

$$A^T A = (V \Sigma^T U^T)(U \Sigma V^T)$$

$$\sigma(A) = \sqrt{\lambda(AA^T)}$$

$$A^T A = V \Sigma^T \Sigma V^T \quad A^T A \text{ is symmetric}$$

$$\sigma(A) = \sqrt{\lambda(A^T A)}$$

Theorem 5.4. The nonzero singular values of A are the square roots of the nonzero eigenvalues of A^*A or AA^* . (These matrices have the same nonzero eigenvalues.)

How to compute SVD (by hand)

$$\left. \begin{array}{l} A = U\Sigma V^T \\ A^T = V\Sigma^T U^T \end{array} \right\} \rightarrow \begin{array}{l} AA^T = (U\Sigma V^T)(V\Sigma^T U^T) \\ A^T A = (V\Sigma^T U^T)(U\Sigma V^T) \end{array} \rightarrow \begin{array}{l} AA^T = U\Sigma\Sigma^T U^T \\ A^T A = V\Sigma^T \Sigma V^T \end{array} \right\} \begin{array}{l} \sigma(A) = \sqrt{\lambda(A^T A)} \\ \sigma(A) = \sqrt{\lambda(AA^T)} \end{array}$$

Example: $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \rightarrow W = A^T A = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \rightarrow \det(W - \lambda I) = \begin{vmatrix} 2-\lambda & 2 \\ 2 & 2-\lambda \end{vmatrix} = 0$

$$u_1 = \frac{1}{\sigma_1} Av_1 = \frac{1}{2\sqrt{2}} A \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \sigma_1 = 2, \sigma_2 = 0 \quad \lambda_1 = 4, \lambda_2 = 0$$

$$u_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \quad u_2, u_3 \text{ orthonormal basis for Null}(AA^T) \quad AA^T = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{Range}(A) \quad \text{Rank}(A) \quad \text{Null}(A) \quad u_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \quad u_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

How to compute SVD (Algorithm)

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

Matrix

Householder transformations

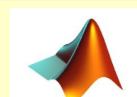
$$\begin{bmatrix} x & x \\ x & x \\ x & x \\ x & x \\ x & x \end{bmatrix}$$

Bidiagonal

$$\begin{bmatrix} x & & & & \\ & x & & & \\ & & x & & \\ & & & x & \\ & & & & x \end{bmatrix}$$

Diagonal

QR transformations



`>> [U,S,V] = svd(A)`

Singular Value Decomposition

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} = U \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} V^T$$

9) If A is a square matrix then

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$$

$$\|A\|_2 = \sigma_1$$

10) If A is a square matrix then

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$$

$$\|A\|_F^2 = \sigma_1^2 + \sigma_2^2 + \dots + \sigma_r^2$$

Example:

$$A = \begin{bmatrix} .96 & 1.72 \\ 2.28 & .96 \end{bmatrix} = U \Sigma V^T = \begin{bmatrix} .6 & -.8 \\ .8 & .6 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} .8 & .6 \\ .6 & -.8 \end{bmatrix}^T.$$

$$\|A\|_2 = 3 \quad \|A\|_F = \sqrt{10}$$

Singular Value Decomposition

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} = U \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} V^T$$

11) If A is a square symmetric matrix then the singular values of A are the absolute values of the eigenvalues of A.

$$A = Q \Sigma Q^T = Q |\Sigma| \text{sign}(\Sigma) Q^T$$

12) If A is a square matrix then

$$|\det(A)| = \prod_{i=1}^n \sigma_i$$

SVD and Eigenvalue Decomposition

SVD	Eigen Decomp
$A = U \Sigma V^T$	$A = S \Sigma S^{-1}$
Uses two different bases U, V	Uses just one (eigenvectors)
Uses orthonormal bases	Generally is not orthogonal
All matrices (even rectangular)	Not all matrices (even square) (only diagonalizable)

Reduced SVD

$$A = U \Sigma V^T$$

$$A = \hat{U} \hat{\Sigma} V^T$$

$$A = \hat{U} \hat{\Sigma} V^T$$

Reduced SVD

Singular Value Decomposition

Example : Luke Olson\illinois



svd_test.m

iguana.jpg

```
1 - clear;
2 -
3 - I = im2double(imread('iguana.jpg'));
4 -
5 - [U,S,V]=svd(I);
6 -
7 - J = zeros(size(I));
8 -
9 - figure(2);clf;
10 - imshow(I)
11 -
12 - figure(1);clf;
13 - for j=1:nnz(S)
14 -     Itmp = S(j,j)*U(:,j)*V(:,j).';
15 -     J = J + Itmp;
16 -
17 -     figure(1);
18 -     imshow(J)
19 -     title(['using vector k = ' num2str(j) ', and \sigma = ' num2str(S(j,j))]);
20 -     pause;
21 - end
```

Singular Value Decomposition

$$A = U \Sigma V^T$$

$$A = U \Sigma V^T$$

Theorem: (Singular Value Decomposition) SVD

If A is real m -by- n matrix, then there exist orthogonal matrices

$$U = [u_1, u_2, \dots, u_m] \in R^{m \times m}$$

$$V = [v_1, v_2, \dots, v_n] \in R^{n \times n}$$

such that

$$A = U \Sigma V^T \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_p)$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$

$$p = \min(m, n)$$

Proof:

Singular Value Decomposition

First approximation to A is

$$A_1 = \sigma_1 u_1 v_1^T$$

second approximation to A is

$$A_2 = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T$$

.....

$$A_\mu = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_\mu u_\mu v_\mu^T$$

.....

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_r u_r v_r^T$$

Theorem : For any μ with $0 \leq \mu \leq r$, the matrix A_μ also satisfies

$$\|A - A_\mu\|_F = \inf_{\substack{B \in R^{m \times n} \\ \text{rank}(B) \leq \mu}} \|A - B\|_F = \sqrt{\sigma_{\mu+1}^2 + \dots + \sigma_r^2}$$

Image Compression

There are hundreds of ways to compress images. Some basic ways use singular value decomposition

Suppose we have an 9 megapixel gray-scale image, which is 3000×3000 pixels (a 3000×3000 matrix). For each pixel, we have some level of black and white, given by some integer between 0 and 255. Each of these integers (and hence each pixel) requires approximately 1 byte to store, resulting in an approximately 8.6 Mb image.

A color image usually has three components, a red, a green, and a blue (RGB). EACH of these is represented by a matrix, so storing color images takes three times the space (25.8 Mb).

We will look at compressing this image through computing the singular value decomposition (SVD).

Reducing the SVD

Using the SVD we can write an $n \times n$ invertible matrix A as:

$$A = P\Sigma Q^T = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n) \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma_n \end{pmatrix} \begin{pmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_n^T \end{pmatrix}$$
$$= \mathbf{p}_1\sigma_1\mathbf{q}_1^T + \mathbf{p}_2\sigma_2\mathbf{q}_2^T + \cdots + \mathbf{p}_n\sigma_n\mathbf{q}_n^T$$

Since $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ the terms $\mathbf{p}_i\sigma_i\mathbf{q}_i^T$ with small i contribute most to the sum, and hence contain the most information about the image. Keeping only some of these terms may result in a lower image quality, but lower storage size. This process is sometimes called Principal Component Analysis (PCA).

Examples: 1 Term

The following is a 500×500 image. The reduced SVD was applied equally to each color:

Original



Using 1 terms



Examples: 3 Terms

The following is a 500×500 image. The reduced SVD was applied equally to each color:

Original



Using 3 terms



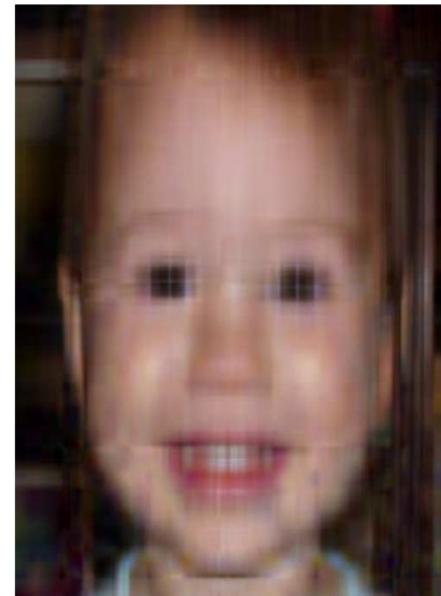
Examples: 10 Terms

The following is a 500×500 image. The reduced SVD was applied equally to each color:

Original



Using 10 terms



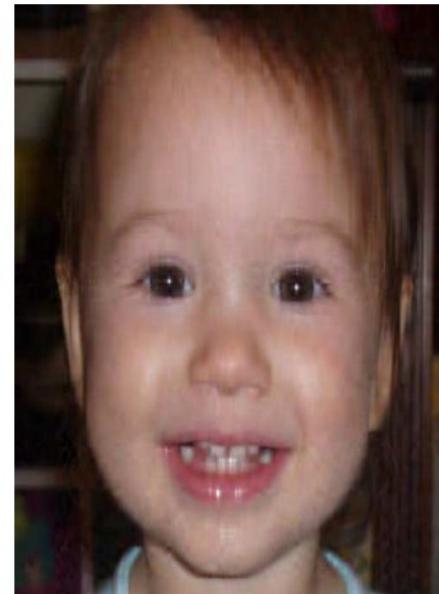
Examples: 30 Terms

The following is a 500×500 image. The reduced SVD was applied equally to each color:

Original



Using 30 terms



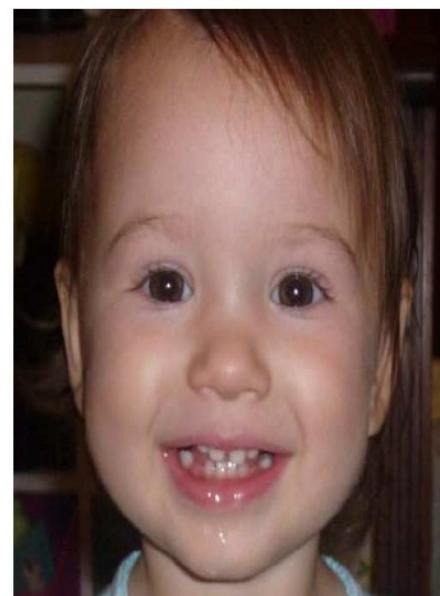
Examples: 100 Terms

The following is a 500×500 image. The reduced SVD was applied equally to each color:

Original



Using 100 terms



How much space is this process saving?

Given an image with a x b pixels:

$$r + r(a) + r(b) = \text{size, where } r = \text{singular values} = \text{rank}$$

$$8 + 8(126) + 8(128) = 2040$$

$$20 + 20(126) + 20(128) = 5100 (\sim 31.6\%)$$

$$50 + 50(126) + 50(128) = 12750 (\sim 79.0\%)$$

Original size:

$$(126) * (128) = 16128$$

Results & Other Applications

- $k = 100$ gives a fairly accurate reproduction, with 7.53% error.
 - The reduced SVD stores $k (2n + 1) = 100 \cdot (1001)$ numbers, $\approx 40\%$ of the original image size.
 - Many uses besides image compression, such as parameterizing possible permeability profiles for underground reservoirs.
-
- Moral of the story: *take more linear algebra and numerical analysis.*
There are hundreds of fun applications!

Support Vector Machines: A Simple Introduction

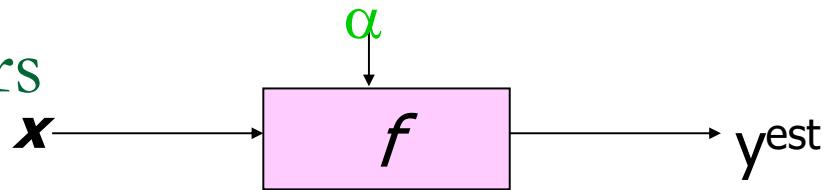
Introduction

- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are **linear classifiers** that find a hyperplane to separate **two class** of data, positive and negative.
- **Kernel functions** are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.
- It is perhaps the best (non-deep) classifier for text classification.

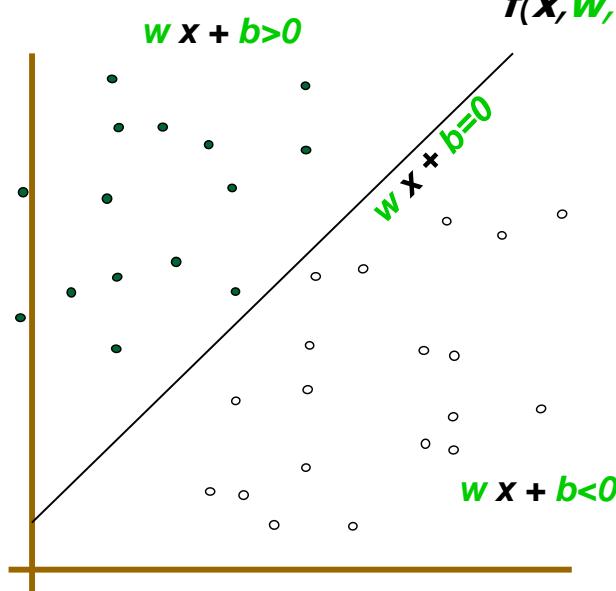
An excellent resource:

https://www.syncfusion.com/ebooks/support_vector_machines_succinctly/solving-the-optimization-problem

Linear Classifiers



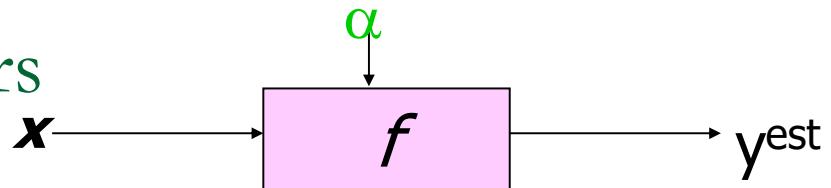
- denotes +1
- denotes -1



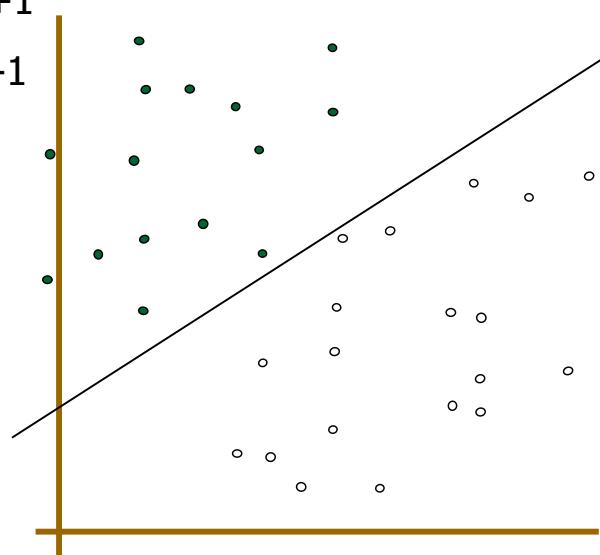
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}\mathbf{x} + b)$$

How would you classify this data?

Linear Classifiers



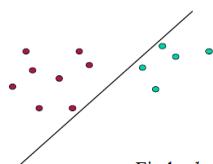
- denotes +1
- denotes -1



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}\mathbf{x} + b)$$

How would you classify this data?

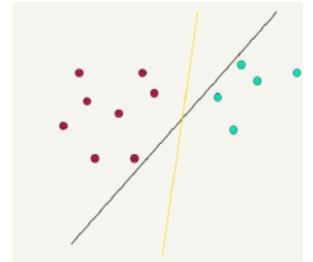
2-D Case



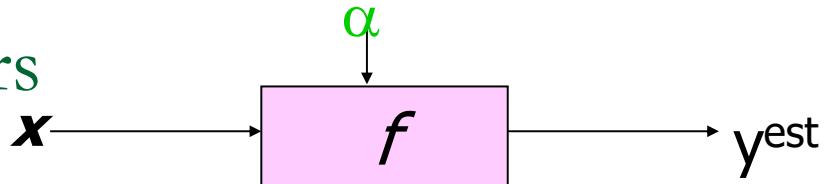
Find a, b, c , such that
 $ax + by \geq c$ for red points
 $ax + by \leq (or <) c$ for green points.

Which Hyperplane to pick?

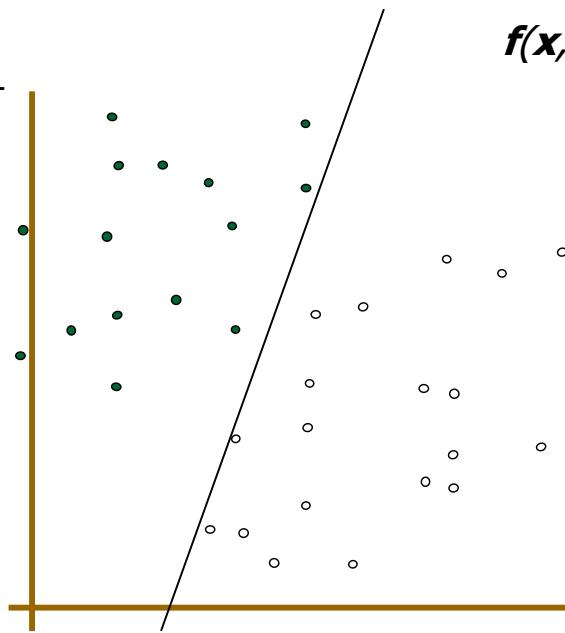
- Lots of possible solutions for a, b, c .
- Some methods find a separating hyperplane, but not the optimal one (e.g., neural net)
- But: Which points should influence optimality?
 - All points?
 - Linear regression
 - Neural nets
 - Or only “difficult points” close to decision boundary
 - Support vector machines



Linear Classifiers



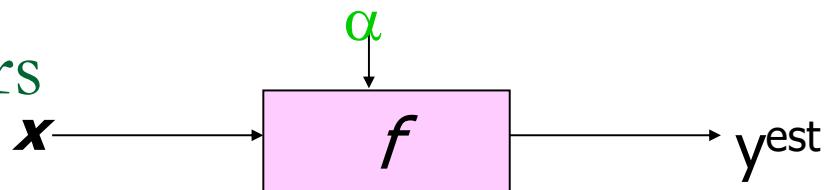
- denotes +1
- denotes -1



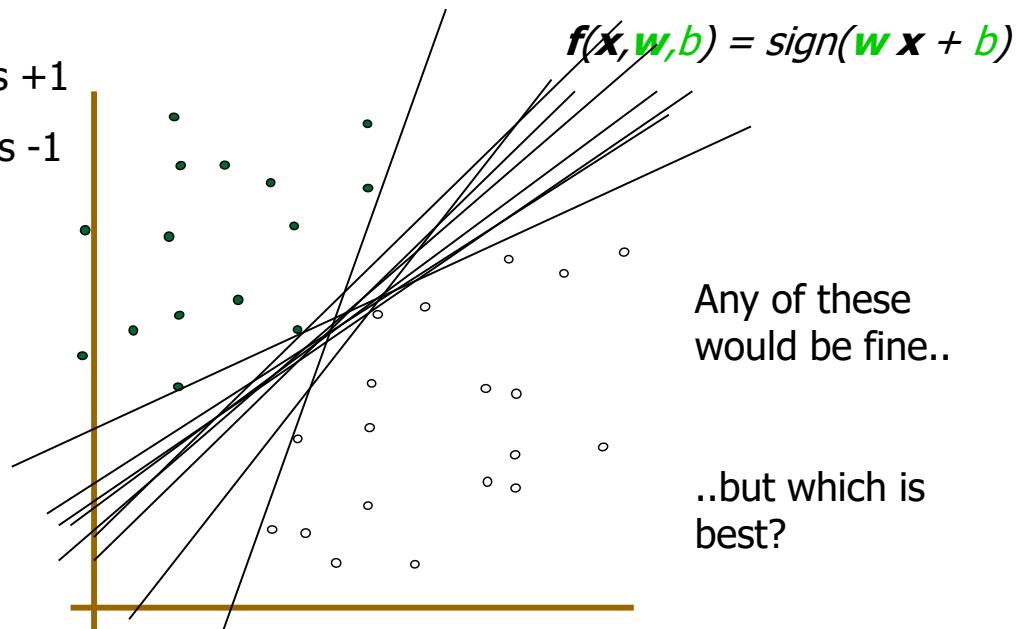
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

How would you
classify this data?

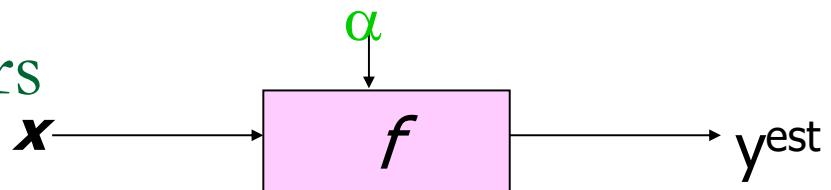
Linear Classifiers



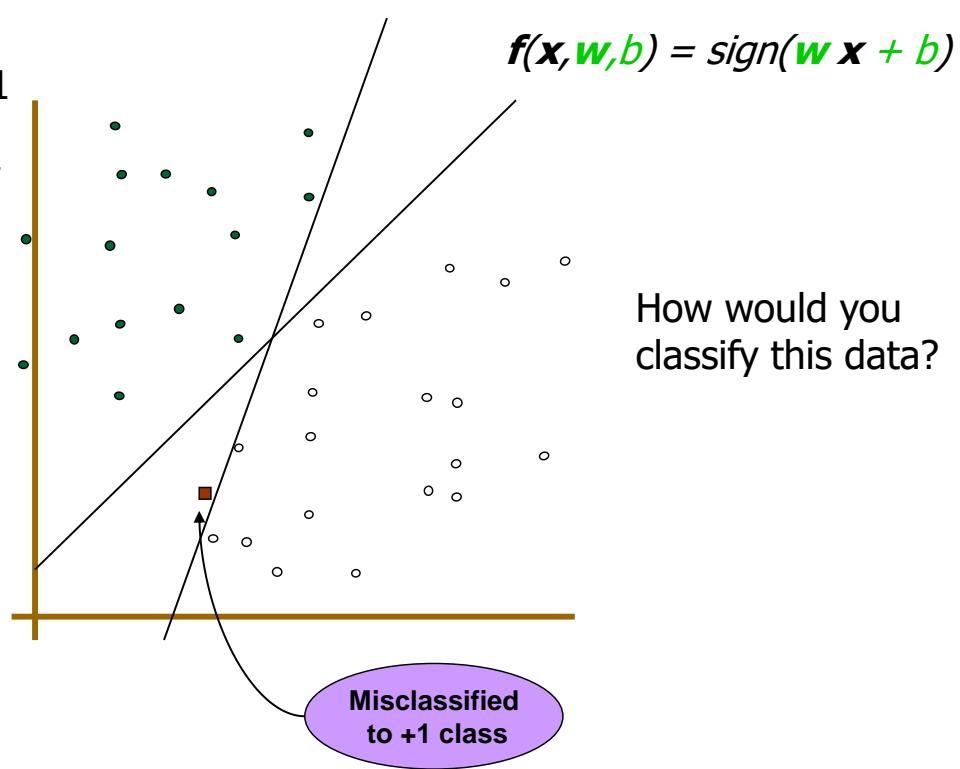
- denotes +1
- denotes -1



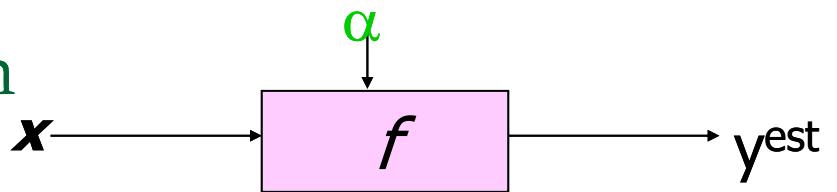
Linear Classifiers



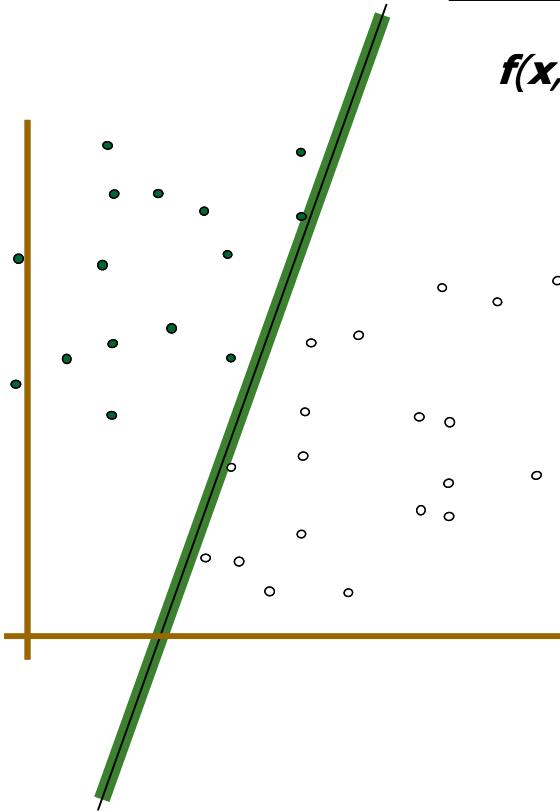
- denotes +1
- denotes -1



Classifier Margin



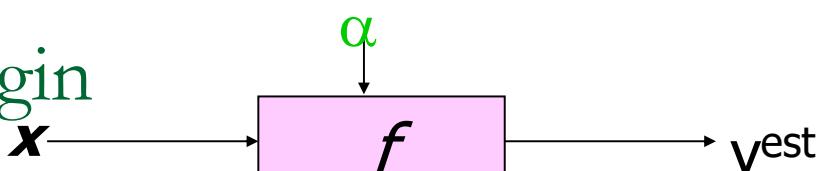
- denotes +1
- denotes -1



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

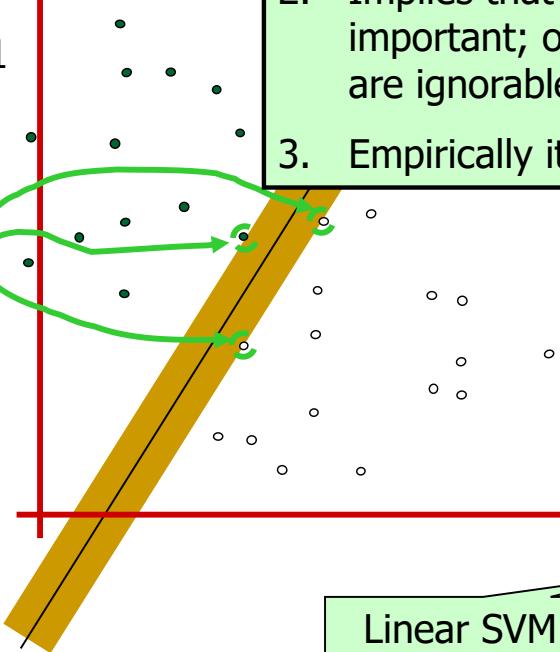
Maximum Margin



- denotes +1
- denotes -1

1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Support Vectors are those datapoints that the margin pushes up against



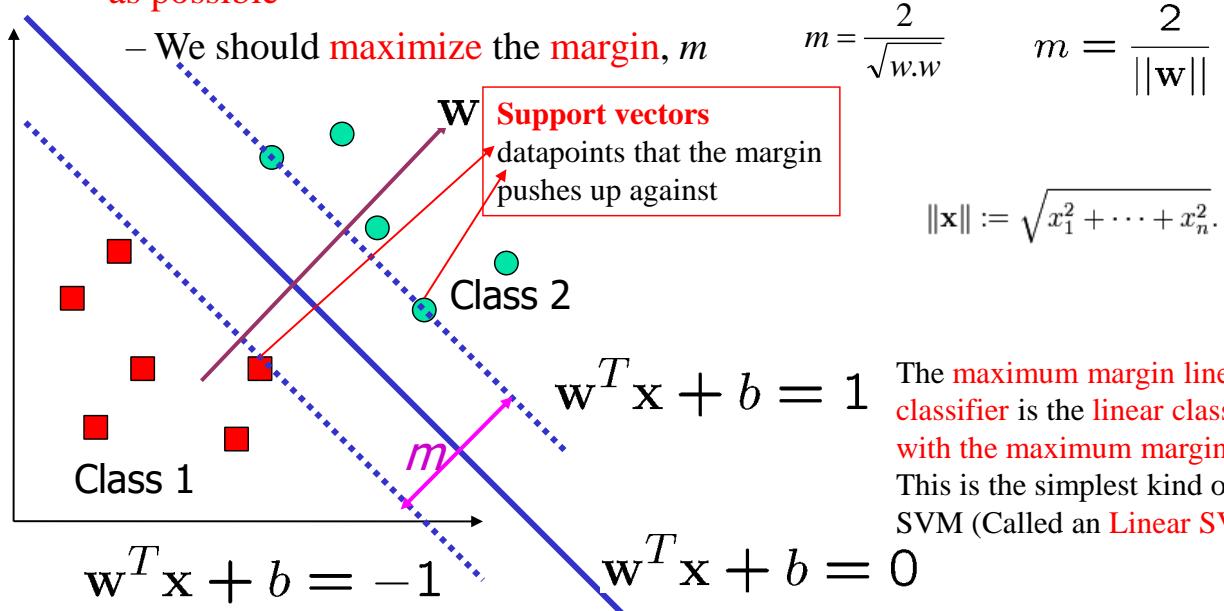
linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Good Decision Boundary: Margin Should Be Large

The decision boundary should be as far away from the data of both classes as possible



The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an Linear SVM)

The SVMs optimization problem

- If $F=1$ it will not affect the result of the optimization problem and $M = \frac{2F}{\|w\|}$.
- $$\underset{w,b}{\text{maximize}} \quad M$$
- $$\text{subject to } f_i \geq F, i = 1, \dots, m$$

- The problem becomes:

$$\underset{w,b}{\text{maximize}} \quad \frac{2}{\|w\|}.$$

$$\text{subject to } f_i \geq 1, i = 1, \dots, m$$

This maximization problem is equivalent to the following minimization problem:

$$\underset{w,b}{\text{minimize}} \quad \|w\|.$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1, i = 1, \dots, m$$

The original optimization problem is difficult to solve. Instead,

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2} \|w\|^2.$$

$$\text{subject to } y_i(w \cdot x_i + b) - 1 \geq 0, i = 1, \dots, m$$

It is convex quadratic optimization problem

Throughout the slides, $x^T y = x \cdot y = \text{the 'dot' product of vectors } x \text{ and } y$.

Solving the Optimization Problem

- The method of Lagrange multipliers
- Given an optimization problem:

$$\begin{aligned} & \underset{x}{\text{minimize}} \quad f(x) \\ & \text{subject to } g(x) = 0 \end{aligned}$$

The minimum of f is found when its gradient point in the same direction as the gradient of g :

$$\nabla f(x) = \alpha \nabla g(x)$$

So if we want to find the minimum of f under the constraint g , we just need to solve for:

$$\nabla f(x) - \alpha \nabla g(x) = 0$$

The constant α is called a Lagrange multiplier.

The Lagrange multiplier method can be summarized by these three steps:

1. Construct the Lagrangian function L by introducing one multiplier per constraint
2. Get the gradient ∇L of the Lagrangian
3. Solve for $\nabla L(x, \alpha)=0$

The SVM Lagrangian problem

- SVM optimization problem:

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2} \|w\|^2.$$

$$\text{subject to } y_i(w \cdot x_i + b) - 1 \geq 0, i = 1, \dots, m$$

- Then $f(w) = \frac{1}{2} \|w\|^2$, m constraint functions $g_i(w, b) = y_i(w \cdot x_i + b) - 1, i = 1, \dots, m$

- Lagrangian function:

$$\begin{aligned} L(w, b, \alpha) &= f(w) - \sum_{i=1}^m \alpha_i g_i(w, b) \\ L(w, b, \alpha) &= \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y_i(w \cdot x_i + b) - 1] \end{aligned}$$

Note that we introduced one Lagrange multiplier α_i for each constraint function.

To get the solution of the primal problem, we need to solve the following Lagrangian problem.

$$\begin{aligned} & \min_{w,b} \max_{\alpha} L(w, b, \alpha). \\ & \text{subject to } \alpha_i \geq 0, \quad i = 1, \dots, m \end{aligned}$$

To sum up, if a solution satisfies the KKT conditions, we are guaranteed that it is the optimal solution.

The Karush-Kuhn-Tucker conditions are:

- Stationarity condition:

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{i=1}^m \alpha_i y_i = 0$$

- Primal feasibility condition:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \text{for all } i = 1, \dots, m$$

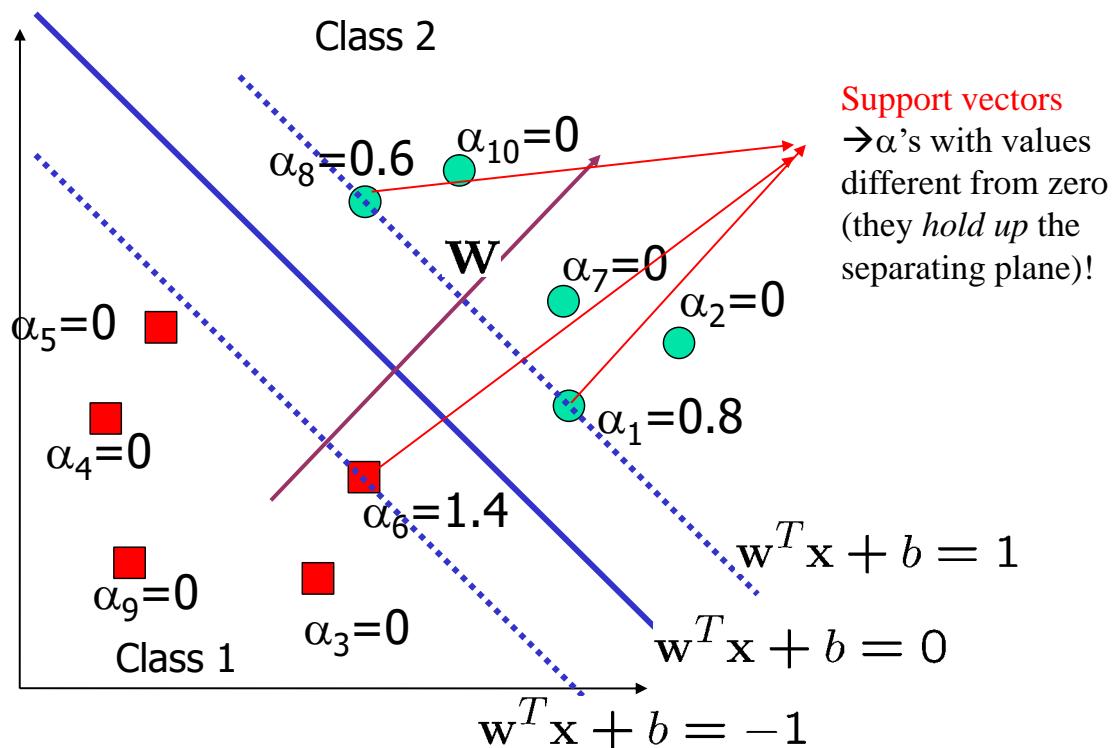
- Dual feasibility condition:

$$\alpha_i \geq 0 \quad \text{for all } i = 1, \dots, m$$

- Complementary slackness condition:

$$\alpha_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0 \quad \text{for all } i = 1, \dots, m$$

A Geometrical Interpretation



Lagrange Dual Function

- **Definition:** the (Lagrange) dual function associated to the constraint optimization problem is defined by

$$\begin{aligned}\forall \alpha \geq 0, F(\alpha) &= \inf_{\mathbf{x} \in X} L(\mathbf{x}, \alpha) \\ &= \inf_{\mathbf{x} \in X} f(\mathbf{x}) + \sum_{i=1}^m \alpha_i g_i(\mathbf{x}).\end{aligned}$$

- F is always concave: Lagrangian is linear with respect to α and \inf preserves concavity.
- $\forall \alpha \geq 0, F(\alpha) \leq p^*$: for a feasible \mathbf{x} ,

$$f(\mathbf{x}) + \sum_{i=1}^m \alpha_i g_i(\mathbf{x}) \leq f(\mathbf{x}).$$

Slide: [Mehryar Mohri - Introduction to Machine Learning](#)

Dual Optimization Problem

- **Definition:** the dual (optimization) problem associated to the constraint optimization is

$$\max_{\alpha} F(\alpha)$$

subject to: $\alpha \geq 0$.

- always a convex optimization problem.
- optimal value d^* .

Weak and Strong Duality

- Weak duality: $d^* \leq p^*$.
 - always holds (clear from previous observations).
- Strong duality: $d^* = p^*$.
 - does not hold in general.
 - holds for convex problems with constraint qualifications.

Slater's constraint qualification condition for convex programming states that strong duality holds if there exists an x that is strictly feasible (i.e. all constraints are satisfied and the nonlinear constraints are satisfied with strict inequalities).

Saddle Point - Sufficient Condition

(Lagrange, 1797)

- **Theorem:** Let P be a constrained optimization problem over $X = \mathbb{R}^N$. If (x^*, α^*) is a saddle point, that is $\forall x \in \mathbb{R}^N, \forall \alpha \geq 0, L(x^*, \alpha) \leq L(x^*, \alpha^*) \leq L(x, \alpha^*)$, then it is a solution of P .

The Wolfe dual problem

- The Lagrangian problem has m inequality constraints (where m is the number of training examples) and is typically solved using its dual form.
- According to duality principle the maximum of the dual problem will always be less than or equal to the minimum of the primal problem (it provides a lower bound to the solution of the primal problem)
- Lagrangian function

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y_i(w \cdot x_i + b) - 1]$$

Solving the minimization problem involves taking the partial derivatives of L with respect to w and b :

$$\begin{aligned}\nabla_w L &= w - \sum_{i=1}^m \alpha_i y_i x_i = 0 & \rightarrow & w = \sum_{i=1}^m \alpha_i y_i x_i \\ \nabla_b L &= -\sum_{i=1}^m \alpha_i y_i = 0 & \rightarrow & \sum_{i=1}^m \alpha_i y_i = 0\end{aligned}$$

Let us substitute w by this value L and we get:

$$W(a, b) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

This is the **Wolfe dual Lagrangian function**

Optional

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y_i(w \cdot x_i + b) - 1] \quad w = \sum_{i=1}^m \alpha_i y_i x_i$$

Let us substitute w by this value into \mathcal{L} :

$$\begin{aligned}W(\alpha, b) &= \frac{1}{2} \left(\sum_{i=1}^m \alpha_i y_i x_i \right) \cdot \left(\sum_{j=1}^m \alpha_j y_j x_j \right) - \sum_{i=1}^m \alpha_i \left[y_i \left(\left(\sum_{j=1}^m \alpha_j y_j x_j \right) \cdot x_i + b \right) - 1 \right] \\ &= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i=1}^m \alpha_i y_i \left(\left(\sum_{j=1}^m \alpha_j y_j x_j \right) \cdot x_i + b \right) + \sum_{i=1}^m \alpha_i \\ &= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - b \sum_{i=1}^m \alpha_i y_i + \sum_{i=1}^m \alpha_i \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - b \sum_{i=1}^m \alpha_i y_i\end{aligned}$$

Wolfe dual problem

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ & \text{subject to } \alpha_i \geq 0, \quad i = 1, \dots, m \\ & \quad \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned}$$

The main advantage of the Wolfe dual problem over the Lagrangian problem is that the objective function W now depends only on the Lagrange multipliers.

When we solve the Wolfe dual problem, we get a vector α containing all Lagrange multipliers and then we will compute w and b .

Compute w :

$$w = \sum_{i=1}^m \alpha_i y_i x_i$$

Compute b :

$$\begin{aligned} w \cdot x_i + b &= y_i \\ b &= y_i - w \cdot x_i \end{aligned}$$

Hypothesis function

The SVMs use the same hypothesis function as the perceptron. The class of an example

$$h(x_i) = \text{sign}(w \cdot x_i + b)$$

When using the dual formulation, it is computed using only the support vectors:

$$h(x_i) = \text{sign} \left(\sum_{j=1}^S \alpha_j y_j (x_j \cdot x_i) + b \right)$$

This formulation of the SVM is called the **hard margin SVM**. It cannot work when the data is not linearly separable. Soft margin SVM applied non-linearly separable data.

The Wolfe Dual Optimization Problem

We can transform the problem to its dual

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

Dot product of X

α 's → New variables
(Lagrangian multipliers)

This is a convex quadratic programming (QP) problem

- Global maximum of α_i can always be found
- well established tools for solving this optimization problem (e.g. cplex)

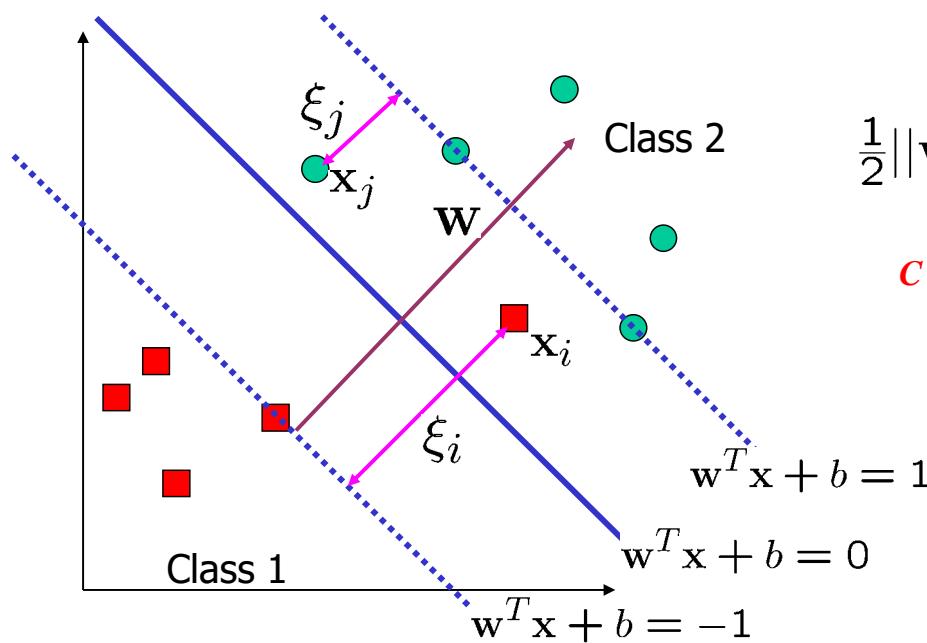
Note:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

Non-linearly Separable Problems

We allow “error” ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T \mathbf{x} + b$

ξ_i approximates the number of misclassified samples



New objective function:

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

C : tradeoff parameter between error and margin;
chosen by the user;
large C means a higher penalty to errors

Soft margin SVM

Slack variables

Given a training set: $D = \{(x_i, y_i) \mid x_i \in R^n, y_i \in \{-1, 1\}\}_{i=1}^m$. In 1995, Vapnik and Cortes introduced a modified version of the original SVM that allows the classifier to make some mistakes. The goal is now not to make zero classification mistakes, but to make as few mistakes as possible. To do so, they modified the constraints of the optimization problem by adding a variable ξ .

$$\begin{aligned} & \underset{w, b, \xi}{\text{minimize}} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ & \text{subject to } y_i(w \cdot x_i + b) \geq 1 - \xi_i, \\ & \quad \xi_i \geq 0 \quad i = 1, \dots, m \end{aligned}$$

where ξ_i -slack variables, C is the regularization parameter.

Wolfe dual problem

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ & \text{subject to} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \\ & \quad \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned}$$

The optimization problem is also called **1-norm soft margin** because we are minimizing the 1-norm of the slack vector ξ .

The Wolfe dual problem

$$\begin{aligned} & \max. \quad W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ & \text{subject to} \quad \boxed{C \geq \alpha_i \geq 0}, \quad \sum_{i=1}^n \alpha_i y_i = 0 \\ & \quad \mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j} \end{aligned}$$

\mathbf{w} is also recovered as

The only difference with the linear separable case is that there is an upper bound C on α_i

Once again, a QP solver can be used to find α_i efficiently!!!

Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points \mathbf{x}_i are support vectors with non-zero Lagrangian multipliers α_i .
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Principal Component Analysis

PCA: Motivations

- ▶ Dimension Reduction: Given a large number of features d , we want to end up with a smaller number of features p , where $p \ll d$.
- ▶ The reduced set of features should have as much of information as possible from the original set of features.
- ▶ If the set of features are correlated, we should extract a set of uncorrelated features.

PCA: Motivations

- ▶ The reduced set of features should have as much of *information* as possible from the original set of features.
- ▶ Measure of information : Feature Variance.

PCA: Motivations

- ▶ Measure of information : Feature Variance.

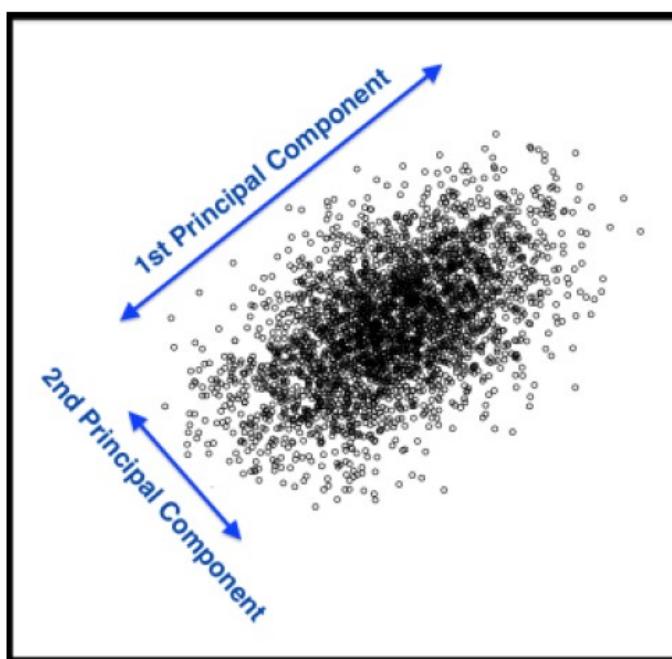


Image Source: <https://www.kdnuggets.com/2016/06/nutrition-principal-component-analysis-tutorial.html>

A Derivation of PCA: How to represent a set of multivariate observations by using a single point?

Define a squared loss function: $J_0(x_0) = \sum_{k=1}^n \|x_0 - x_k\|^2$ with $m = \frac{1}{n} \sum_{k=1}^n x_k$

$$\begin{aligned} \text{Clearly, } J_0(x_0) &= \sum_{k=1}^n \|(x_0 - m) - (x_k - m)\|^2 \\ &= \sum_{k=1}^n \|x_0 - m\|^2 - 2 \sum_{k=1}^n (x_0 - m)^T (x_k - m) + \sum_{k=1}^n \|x_k - m\|^2 \\ &= \sum_{k=1}^n \|x_0 - m\|^2 - 2(x_0 - m)^T \sum_{k=1}^n (x_k - m) + \sum_{k=1}^n \|x_k - m\|^2 \end{aligned}$$

$$= \sum_{k=1}^n \|x_0 - m\|^2 + \sum_{k=1}^n \|x_k - m\|^2$$

Thus, sample mean provides a zero-dimensional representation

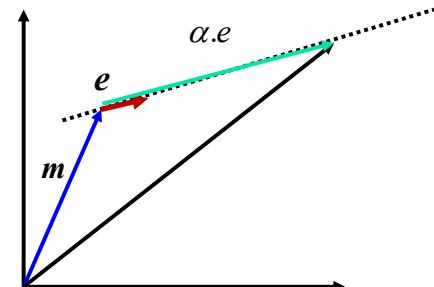
implies x_0 has to be m

independent of m

Instead of a point, can we represent the set of multivariate points (data objects) using a line passing through m ?

$$x = m + \alpha \cdot e \Leftrightarrow x_k = m + \alpha_k \cdot e$$

$$\begin{aligned} J_1(\alpha_1, \alpha_2, \dots, \alpha_n, e) &= \sum_{k=1}^n \|(m + \alpha_k e) - x_k\|^2 \\ &= \sum_{k=1}^n \|\alpha_k e - (x_k - m)\|^2 \\ &= \sum_{k=1}^n \alpha_k^2 \|e\|^2 - 2 \sum_{k=1}^n \alpha_k e^T (x_k - m) + \sum_{k=1}^n \|x_k - m\|^2 \end{aligned}$$



Letting $\|e\| = 1$ differentiating w.r.t. α_k and setting it to zero, $\alpha_k = e^T (x_k - m)$

Now can we find the best direction for the line e ?

Let's define the scatter matrix: $S = \sum_{k=1}^n (x_k - m)(x_k - m)^T$

Now by plugging in $\alpha_k = e^T(x_k - m)$

$$\begin{aligned} J_1(e) &= \sum_{k=1}^n \alpha_k^2 - 2 \sum_{k=1}^n \alpha_k^2 + \sum_{k=1}^n \|x_k - m\|^2 = - \sum_{k=1}^n [e^T(x_k - m)]^2 + \sum_{k=1}^n \|x_k - m\|^2 \\ &= - \sum_{k=1}^n e^T(x_k - m)(x_k - m)^T e + \sum_{k=1}^n \|x_k - m\|^2 \\ &= -e^T S e + \sum_{k=1}^n \|x_k - m\|^2 \end{aligned}$$

Clearly, e that minimizes J_1 also maximizes $e^T S e$

Let's form the Lagrangian as: $u = e^T S e - \lambda(e^T e - 1)$

By differentiating, $\frac{\partial u}{\partial e} = 2Se - 2\lambda e = 0 \Rightarrow Se = \lambda e$.

As, $e^T S e = \lambda e^T e = \lambda$ maximizing this term is equivalent to finding the largest eigenvalue of S

Thus, we project along the eigenvector corresponding to the largest eigenvalue of matrix S .

Can we generalize to a d' dimensional space instead of a line (which is 1D)?

Yes, just choose: $x = m + \sum_{i=1}^{d'} \alpha_i e_i$

and minimize: $J_{d'} = \sum_{k=1}^n \left\| \left(m + \sum_{i=1}^{d'} \alpha_i e_i \right) - x_k \right\|^2$

The minimization corresponds to the largest d' eigenvalues of the scatter matrix S.

Principal Component Analysis

- **Principal component analysis**

- It is a way of identifying the underlying patterns in data
- It can extract information in a large data set with many variables and approximate this data set with fewer factors
- In other words, it can reduce the number of variables to a more manageable set

- **Steps of the principal component analysis**

- Step 1: Get some data
- Step 2: Subtract the mean
- Step 3: Calculate the covariance matrix
- Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix
- Step 5: Deriving the transformed data set
- Step 6: Getting the original data back

Step 1:

x^*	y^*
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9
1.81	1.91

Step 2:

x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01
0	0

demeaned
⇒

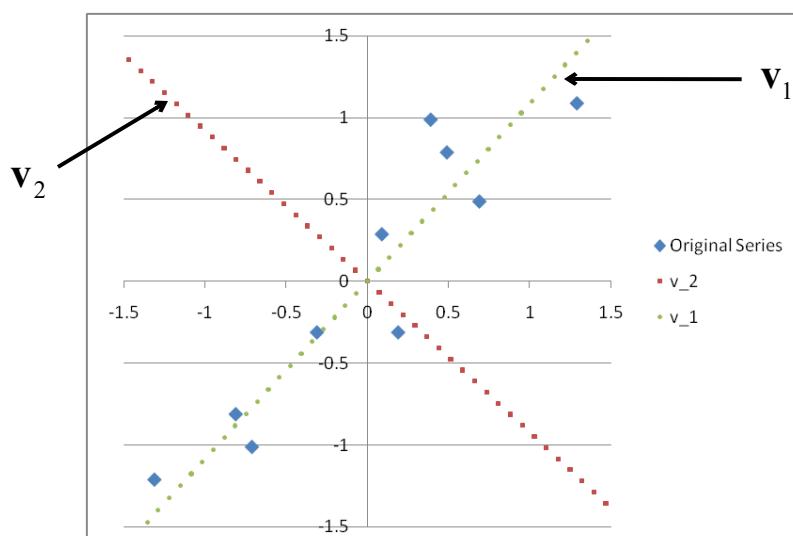
$$\equiv X^T = [x \ y]$$

Step 3:

$$\begin{aligned} \text{var}(X^T) &= E[XX^T] = E\left[\begin{bmatrix} x^T \\ y^T \end{bmatrix} \begin{bmatrix} x & y \end{bmatrix}\right] = E\begin{bmatrix} x^T x & x^T y \\ y^T x & y^T y \end{bmatrix} \\ &= \begin{bmatrix} \text{var}(x) & \text{cov}(x, y) \\ \text{cov}(x, y) & \text{var}(y) \end{bmatrix} = \begin{pmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{pmatrix} \equiv A \end{aligned}$$

- Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix A

$$\lambda_1 = 1.284028, \mathbf{v}_1 = \begin{pmatrix} -0.67787 \\ -0.73518 \end{pmatrix} \quad \lambda_2 = 0.049083, \mathbf{v}_2 = \begin{pmatrix} -0.73518 \\ 0.67787 \end{pmatrix}$$



1. The two eigenvectors are perpendicular (orthogonal) to each other according to Thm. 9 (In fact, they are orthonormal here)
2. \mathbf{v}_1 eigenvector (corresponding to the largest eigenvalue) is just like a best-fit regression line
3. \mathbf{v}_2 seems less important to explain the data since the projection of each node on the \mathbf{v}_2 axis is very close to zero
4. The interpretation of \mathbf{v}_1 is the new axis which retains as much as possible the **interpoint distance information** (or said the **variance information**) that was contained in the original two dimensions

※ The intuition behind the principal component analysis:

(1) Total variance of series of x and y = variance of x + variance of y

= sum of the main diagonal entries in the covariance matrix A

= $0.616556+0.716556 = 1.33311$ (The series x , which are the coordinate values on the x -axis, explains $0.616556/1.33311 = 46.25\%$ of total variance)

(2) Consider $P = [\mathbf{v}_1 \ \mathbf{v}_2]$ and $X = PX'$. (According to the Principal Axes Theorem on Slide 7.65, it is equivalent to transform x - and y -axes to be \mathbf{v}_1 - and \mathbf{v}_2 -axes, i.e., the data is the same but with different coordinate values X' on the $\mathbf{v}_1\mathbf{v}_2$ -plane.)

$$\Rightarrow X' = P^{-1}X = P^T X \Rightarrow (X')^T = X^T P, \text{ where } (X')^T = [x' \ y'] \text{ and } X^T = [x \ y]$$

$$\Rightarrow \text{var}((X')^T) = \text{var}(X^T P) = P^T \text{var}(X^T)P = P^T AP = D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad (\text{It also implies that the new series of } x' \text{ and } y' \text{ are independent})$$

(3) Total variance of transformed series of x' and y' (called **principal components**) in X'

= variance of x' + variance of y'

= sum of the main diagonal entries in the covariance matrix $\text{var}((X')^T) = \lambda_1 + \lambda_2$

(4) A property for eigenvalues: $\text{Trace}(A) = \sum \lambda_i$, which means that after the transformation, the total variance remains the same. In this case, $\lambda_1 + \lambda_2 = 1.284028 + 0.049083 = 1.33311$.

(5) The new series x' , which are the coordinate values on the \mathbf{v}_1 -axis, explains $\lambda_1 / (\lambda_1 + \lambda_2)$
 $= 1.284028 / (1.284028 + 0.049083) = 96.32\%$ of total variance

• Step 5: Deriving the transformed data set: $(X')^T = X^T P$

Case 1: $P = [\mathbf{v}_1 \ \mathbf{v}_2] = \begin{bmatrix} v_{11} = -0.67787 & v_{21} = -0.73518 \\ v_{12} = -0.73518 & v_{22} = 0.67787 \end{bmatrix}$

$$(X')^T = [x' \ y'] = [x \ y] \begin{bmatrix} v_{11} = -0.67787 & v_{21} = -0.73518 \\ v_{12} = -0.73518 & v_{22} = 0.67787 \end{bmatrix} \\ = [v_{11}x + v_{12}y \ v_{21}x + v_{22}y] \\ = [-0.67787x - 0.73518y \ -0.73518x + 0.67787y]$$

x'	y'
-0.82797	-0.17512
1.77758	0.14286
-0.99220	0.38437
-0.27421	0.13042
-1.67580	-0.20950
-0.91295	0.17528
0.09911	-0.34982
1.14457	0.04642
0.43805	0.01776
1.22382	-0.16268
0	0

$$(X')^T =$$

$$\Rightarrow \text{var}((X')^T) = \begin{bmatrix} 1.284028 & 0 \\ 0 & 0.049083 \end{bmatrix}$$

Case 2: Set $y' = 0$ on purpose

$$(X')^T = [x' \ y'] \\ = [v_{11}x + v_{12}y \ 0] \\ = [-0.67787x - 0.73518y \ 0]$$

$$(X')^T =$$

x'	y'
-0.82797	0
1.77758	0
-0.99220	0
-0.27421	0
-1.67580	0
-0.91295	0
0.09911	0
1.14457	0
0.43805	0
1.22382	0
0	0

$$\Rightarrow \text{var}((X')^T) = \begin{bmatrix} 1.284028 & 0 \\ 0 & 0 \end{bmatrix}$$

- Step 6: Getting the original data back:

$$X^T = (X')^T P^{-1} (= (X')^T P^T) + \text{original mean, where } P = [\mathbf{v}_1 \ \mathbf{v}_2]$$

$$\begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix} \begin{bmatrix} v_{11} = -0.67787 & v_{12} = -0.73518 \\ v_{21} = -0.73518 & v_{22} = 0.67787 \end{bmatrix} = \begin{bmatrix} v_{11}x' + v_{21}y' & v_{12}x' + v_{22}y' \\ -0.67787x' - 0.73518y' & -0.73518x' + 0.67787y' \end{bmatrix}$$

$$= [-0.67787x' - 0.73518y' \quad -0.73518x' + 0.67787y']$$

case 1

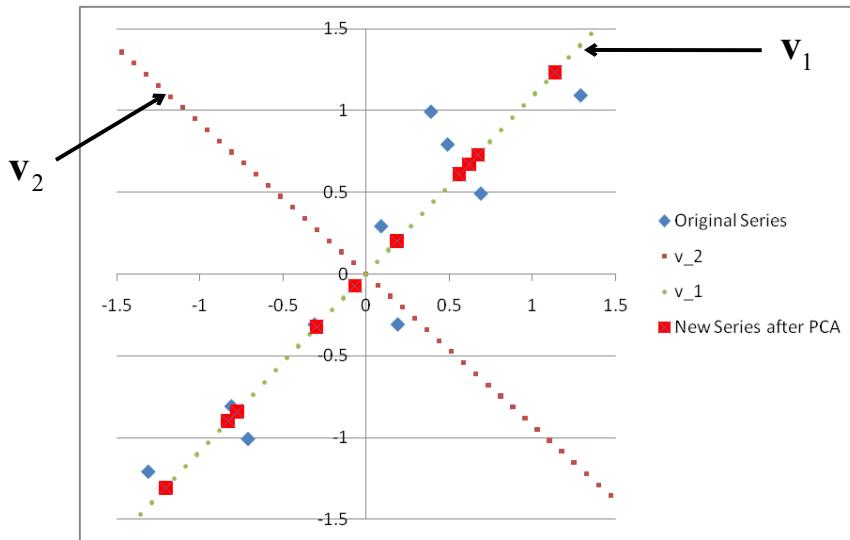
x^*	y^*
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9
1.81	1.91

case 2

x^*	y^*
2.37	2.52
0.61	0.60
2.48	2.64
2.00	2.11
2.95	3.14
2.43	2.58
1.74	1.84
1.03	1.07
1.51	1.59
0.98	1.01
1.81	1.91

※ We can derive the original data set if we take both \mathbf{v}_1 and \mathbf{v}_2 and thus x' and y' into account when deriving the transformed data

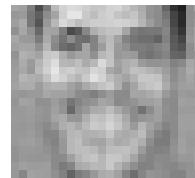
※ Although when we derive the transformed data, only \mathbf{v}_1 and thus only x' are considered, the data gotten back is still similar to the original data. That means, x' can be a common factor almost able to explain both series x and y



- ※ If only the principal component x' is considered in the Principal Component Analysis, it is equivalent to project all points onto the \mathbf{v}_1 vector
- ※ It can be observed in the above figure that the projection onto \mathbf{v}_1 vector can retain as much as possible the interpoint distance information (variance) that was contained in the original series of (x, y)

Starting idea of the eigenface algorithm:

- Treat pixels as a vector



X

- Recognize face by nearest neighbor



y₁ ... y_n

$$k = \operatorname{argmin}_k \|y_k^T - x\|$$

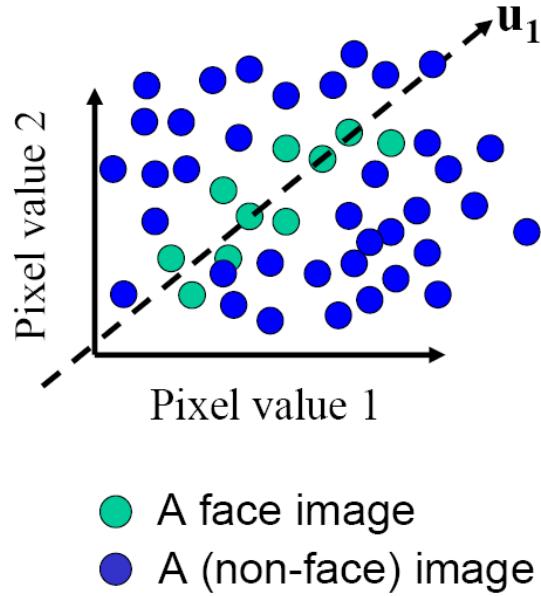
The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
 - 100x100 image = 10,000 dimensions
 - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images



The space of all face images

- Eigenface idea: construct a low-dimensional linear subspace that best explains the variation in the set of face images



Principal Component Analysis (PCA)

- Given: N data points $\mathbf{x}_1, \dots, \mathbf{x}_N$ in \mathbb{R}^d
- We want to find a new set of features that are linear combinations of original ones:
$$u(\mathbf{x}_i) = \mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})$$
($\boldsymbol{\mu}$: mean of data points)
- Choose unit vector \mathbf{u} in \mathbb{R}^d that captures the most data variance

Principal Component Analysis

- Direction that maximizes the variance of the projected data:

$$\text{Maximize} \quad \frac{1}{N} \sum_{i=1}^N \underbrace{\mathbf{u}^T(\mathbf{x}_i - \mu)(\mathbf{u}^T(\mathbf{x}_i - \mu))^T}_{\text{Projection of data point}} \quad \text{subject to } \|\mathbf{u}\|=1$$

$$\begin{aligned} &= \mathbf{u}^T \left[\underbrace{\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T}_{\text{Covariance matrix of data}} \right] \mathbf{u} \\ &= \mathbf{u}^T \Sigma \mathbf{u} \end{aligned}$$

The direction that maximizes the variance is the eigenvector associated with the largest eigenvalue of Σ

Implementation issue

- Covariance matrix is huge (N^2 for N pixels)
- But typically # examples $\ll N$
- Simple trick
 - \mathbf{X} is matrix of normalized training data
 - Solve for eigenvectors \mathbf{u} of $\mathbf{X}\mathbf{X}^T$ instead of $\mathbf{X}^T\mathbf{X}$
 - Then $\mathbf{X}^T\mathbf{u}$ is eigenvector of covariance $\mathbf{X}^T\mathbf{X}$
 - May need to normalize (to get unit length vector)

Eigenfaces (PCA on face images)

1. Compute covariance matrix of face images
2. Compute the principal components (“eigenfaces”)
 - K eigenvectors with largest eigenvalues
3. Represent all face images in the dataset as linear combinations of eigenfaces
 - Perform nearest neighbor on these coefficients

M. Turk and A. Pentland, [Face Recognition using Eigenfaces](#), CVPR 1991

Eigenfaces example

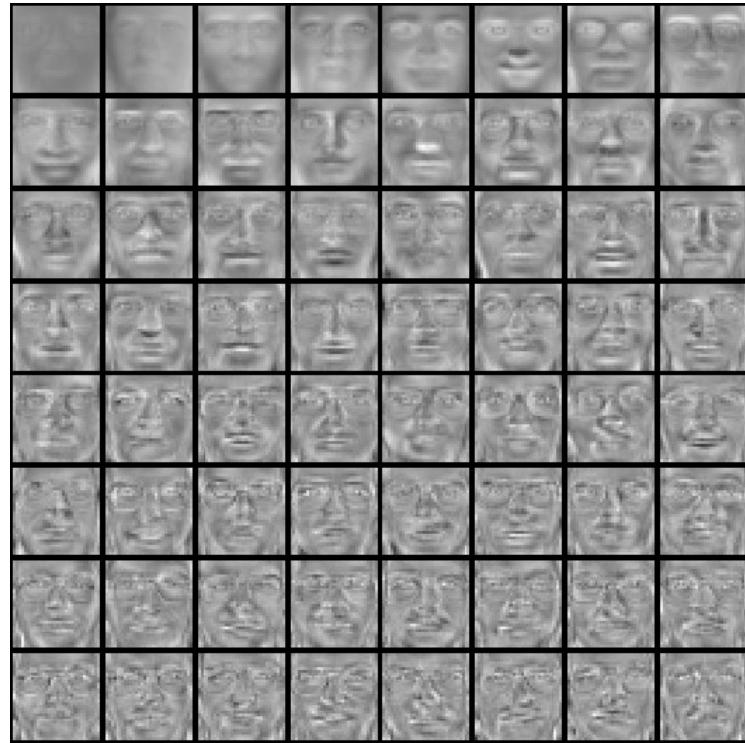
- Training images
- $\mathbf{X}_1, \dots, \mathbf{X}_N$



Eigenfaces example

Top eigenvectors: u_1, \dots, u_k

Mean: μ



Visualization of eigenfaces

Principal component (eigenvector) u_k



$\mu + 3\sigma_k u_k$

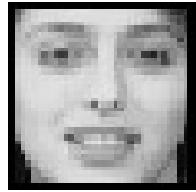


$\mu - 3\sigma_k u_k$



Representation and reconstruction

- Face \mathbf{x} in “face space” coordinates:



$$\begin{aligned}\mathbf{x} &\rightarrow [\mathbf{u}_1^T(\mathbf{x} - \mu), \dots, \mathbf{u}_k^T(\mathbf{x} - \mu)] \\ &= w_1, \dots, w_k\end{aligned}$$

Representation and reconstruction

- Face \mathbf{x} in “face space” coordinates:



$$\begin{aligned}\mathbf{x} &\rightarrow [\mathbf{u}_1^T(\mathbf{x} - \mu), \dots, \mathbf{u}_k^T(\mathbf{x} - \mu)] \\ &= w_1, \dots, w_k\end{aligned}$$

- Reconstruction:

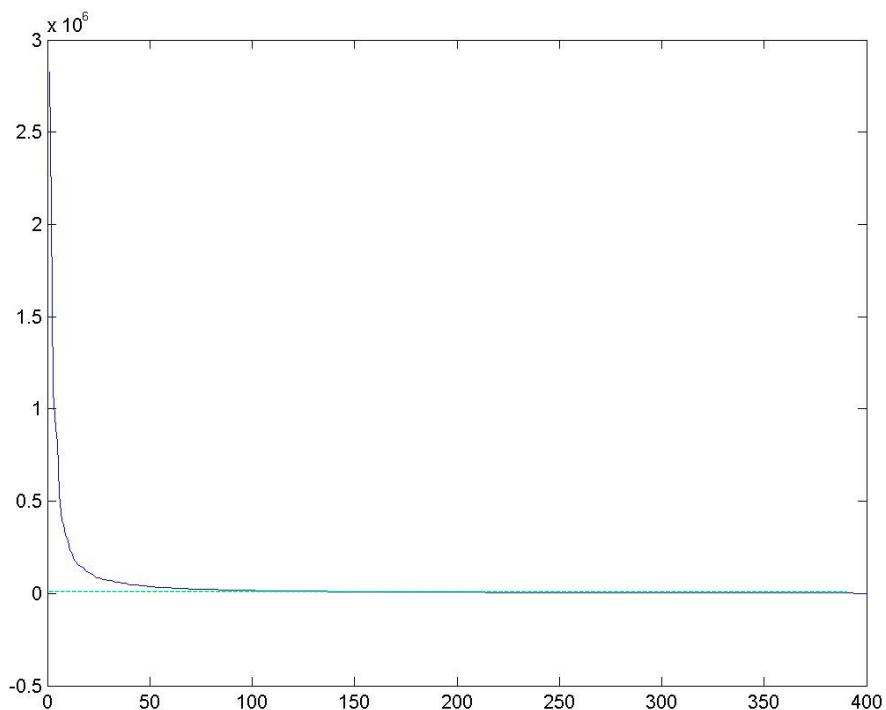
$$\begin{aligned}\hat{\mathbf{x}} &= \mathbf{\mu} + \begin{matrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 & \mathbf{u}_5 & \mathbf{u}_6 & \mathbf{u}_7 \end{matrix} \\ \hat{\mathbf{x}} &= \mathbf{\mu} + w_1 \mathbf{u}_1 + w_2 \mathbf{u}_2 + w_3 \mathbf{u}_3 + w_4 \mathbf{u}_4 + \dots\end{aligned}$$

Reconstruction



After computing eigenfaces using 400 face images from ORL face database

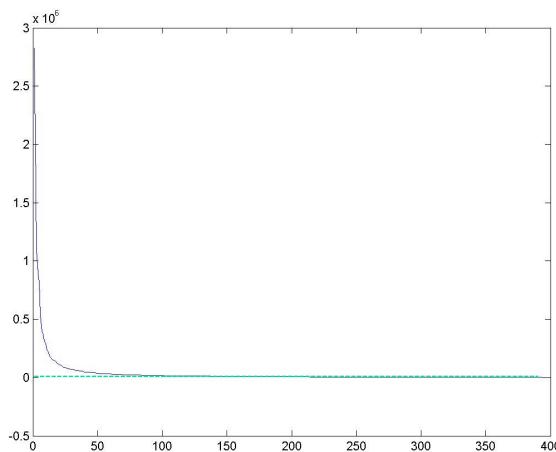
Eigenvalues (variance along eigenvectors)



Note

Preserving variance (minimizing MSE) does not necessarily lead to qualitatively good reconstruction.

$P = 200$



Recognition with eigenfaces

Process labeled training images

- Find mean μ and covariance matrix Σ
- Find k principal components (eigenvectors of Σ) u_1, \dots, u_k
- Project each training image x_i onto subspace spanned by principal components:
 $(w_{i1}, \dots, w_{ik}) = (u_1^T(x_i - \mu), \dots, u_k^T(x_i - \mu))$

Given novel image x

- Project onto subspace:
 $(w_1, \dots, w_k) = (u_1^T(x - \mu), \dots, u_k^T(x - \mu))$
- Optional: check reconstruction error $\|x - \hat{x}\|$ to determine whether image is really a face
- Classify as closest training face in k -dimensional subspace

PCA

- General dimensionality reduction technique
- Preserves most of variance with a much more compact representation
 - Lower storage requirements (eigenvectors + a few numbers per face)
 - Faster matching
- What are the problems for face recognition?

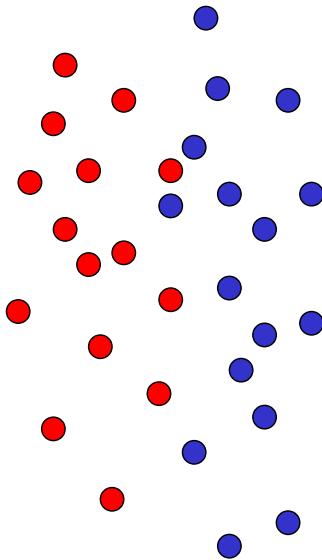
Limitations

Global appearance method: not robust to misalignment, background variation



Limitations

- The direction of maximum variance is not always good for classification



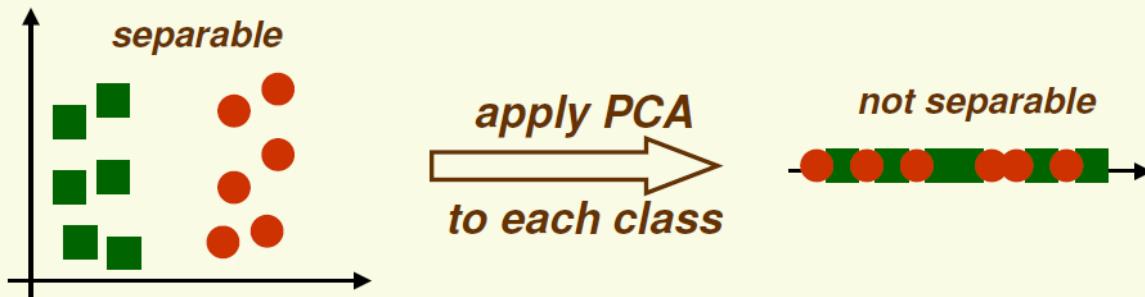
A more discriminative subspace: FLD

- Fisher Linear Discriminants → “Fisher Faces”
- PCA preserves maximum variance
- FLD preserves discrimination
 - Find projection that maximizes scatter between classes and minimizes scatter within classes

Reference: [Eigenfaces vs. Fisherfaces, Belhumeur et al., PAMI 1997](#)

Data Representation vs. Data Classification

- PCA finds the most accurate *data representation* in a lower dimensional space
 - Project data in the directions of maximum variance
- However the directions of maximum variance may be useless for classification



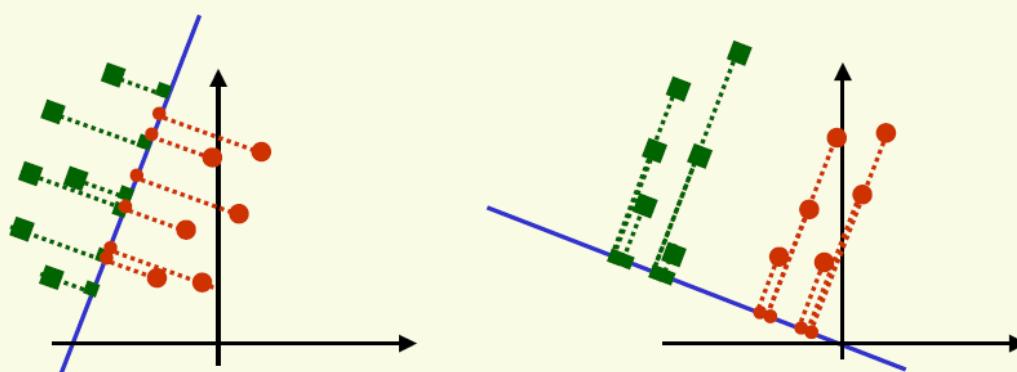
- Fisher Linear Discriminant project to a line which preserves direction useful for *data classification*

LDA Slides: Prof. Olga Veksler, Univ. of Waterloo

Fisher Linear Discriminant

- Main idea: find projection to a line s.t. samples from different classes are well separated

Example in 2D

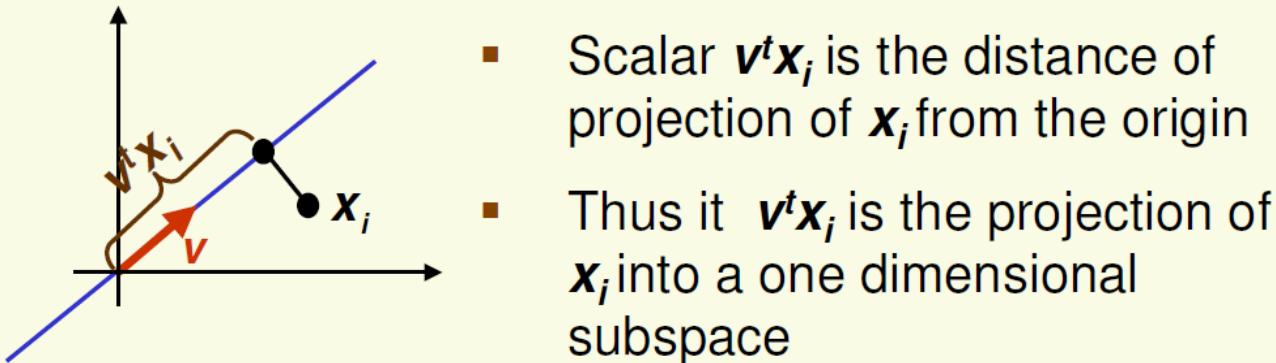


*bad line to project to,
classes are mixed up*

*good line to project to,
classes are well separated*

Fisher Linear Discriminant

- Suppose we have 2 classes and d -dimensional samples $\mathbf{x}_1, \dots, \mathbf{x}_n$ where
 - n_1 samples come from the first class
 - n_2 samples come from the second class
- consider projection on a line
- Let the line direction be given by unit vector \mathbf{v}



Fisher Linear Discriminant

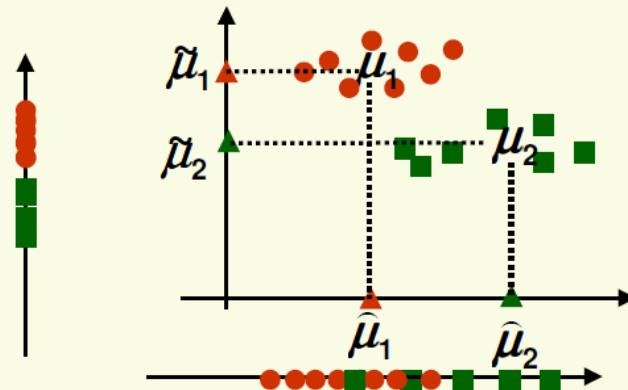
- Thus the projection of sample \mathbf{x}_i onto a line in direction \mathbf{v} is given by $\mathbf{v}^t \mathbf{x}_i$
- How to measure separation between projections of different classes?
- Let $\tilde{\mu}_1$ and $\tilde{\mu}_2$ be the means of projections of classes 1 and 2
- Let μ_1 and μ_2 be the means of classes 1 and 2
- $|\tilde{\mu}_1 - \tilde{\mu}_2|$ seems like a good measure

$$\tilde{\mu}_1 = \frac{1}{n_1} \sum_{x_i \in C1} \mathbf{v}^t \mathbf{x}_i = \mathbf{v}^t \left(\frac{1}{n_1} \sum_{x_i \in C1} \mathbf{x}_i \right) = \mathbf{v}^t \mu_1$$

$$similarly, \quad \tilde{\mu}_2 = \mathbf{v}^t \mu_2$$

Fisher Linear Discriminant

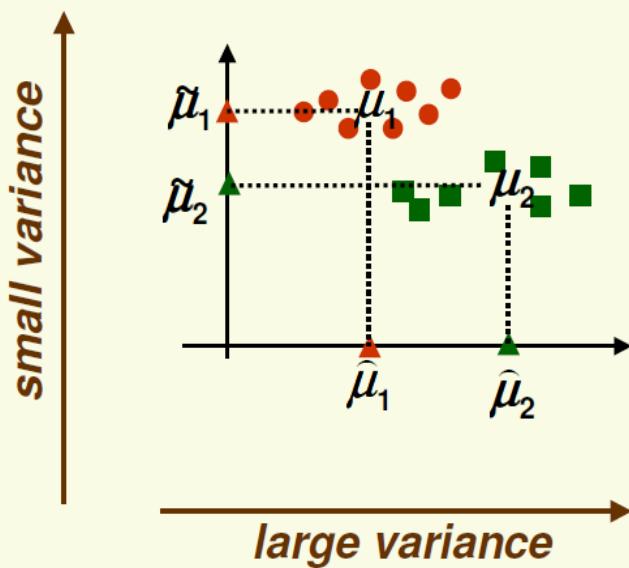
- How good is $|\tilde{\mu}_1 - \tilde{\mu}_2|$ as a measure of separation?
 - The larger $|\tilde{\mu}_1 - \tilde{\mu}_2|$, the better is the expected separation



- the vertical axes is a better line than the horizontal axes to project to for class separability
- however $|\hat{\mu}_1 - \hat{\mu}_2| > |\tilde{\mu}_1 - \tilde{\mu}_2|$

Fisher Linear Discriminant

- The problem with $|\tilde{\mu}_1 - \tilde{\mu}_2|$ is that it does not consider the variance of the classes



Fisher Linear Discriminant

- We need to normalize $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by a factor which is proportional to variance
- Have samples $\mathbf{z}_1, \dots, \mathbf{z}_n$. Sample mean is $\mu_z = \frac{1}{n} \sum_{i=1}^n \mathbf{z}_i$
- Define their **scatter** as
$$s = \sum_{i=1}^n (\mathbf{z}_i - \mu_z)^2$$
- Thus scatter is just sample variance multiplied by n
 - scatter measures the same thing as variance, the spread of data around the mean
 - scatter is just on different scale than variance

larger scatter:



smaller scatter:



Fisher Linear Discriminant

- Fisher Solution: normalize $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by scatter
- Let $\mathbf{y}_i = \mathbf{v}^t \mathbf{x}_i$, i.e. \mathbf{y}_i 's are the projected samples
- Scatter for projected samples of class 1 is
$$\tilde{s}_1^2 = \sum_{\mathbf{y}_i \in \text{Class 1}} (\mathbf{y}_i - \tilde{\mu}_1)^2$$
- Scatter for projected samples of class 2 is
$$\tilde{s}_2^2 = \sum_{\mathbf{y}_i \in \text{Class 2}} (\mathbf{y}_i - \tilde{\mu}_2)^2$$

Fisher Linear Discriminant

- We need to normalize by both scatter of class 1 and scatter of class 2
- Thus Fisher linear discriminant is to project on line in the direction \mathbf{v} which maximizes

want projected means are far from each other

$$J(\mathbf{v}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

want scatter in class 1 is as small as possible, i.e. samples of class 1 cluster around the projected mean $\tilde{\mu}_1$

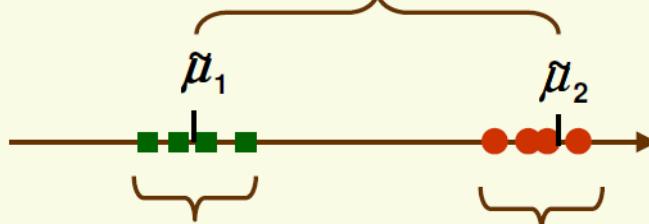
want scatter in class 2 is as small as possible, i.e. samples of class 2 cluster around the projected mean $\tilde{\mu}_2$

Fisher Linear Discriminant

$$J(\mathbf{v}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

- If we find \mathbf{v} which makes $J(\mathbf{v})$ large, we are guaranteed that the classes are well separated

projected means are far from each other



small \tilde{s}_1 implies that projected samples of class 1 are clustered around projected mean

small \tilde{s}_2 implies that projected samples of class 2 are clustered around projected mean

Fisher Linear Discriminant Derivation

$$J(\mathbf{v}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

- All we need to do now is to express J explicitly as a function of \mathbf{v} and maximize it
 - straightforward but need linear algebra and Calculus
- Define the separate class scatter matrices \mathbf{S}_1 and \mathbf{S}_2 for classes 1 and 2. These measure the scatter of original samples \mathbf{x}_i (before projection)

$$\mathbf{S}_1 = \sum_{x_i \in \text{Class 1}} (\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^t$$

$$\mathbf{S}_2 = \sum_{x_i \in \text{Class 2}} (\mathbf{x}_i - \boldsymbol{\mu}_2)(\mathbf{x}_i - \boldsymbol{\mu}_2)^t$$

Fisher Linear Discriminant Derivation

- Now define the **within** the class scatter matrix

$$\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2$$

- Recall that $\tilde{s}_1^2 = \sum_{y_i \in \text{Class 1}} (\mathbf{y}_i - \tilde{\boldsymbol{\mu}}_1)^2$

- Using $\mathbf{y}_i = \mathbf{v}^t \mathbf{x}_i$ and $\tilde{\boldsymbol{\mu}}_1 = \mathbf{v}^t \boldsymbol{\mu}_1$

$$\begin{aligned}\tilde{s}_1^2 &= \sum_{y_i \in \text{Class 1}} (\mathbf{v}^t \mathbf{x}_i - \mathbf{v}^t \boldsymbol{\mu}_1)^2 \\ &= \sum_{y_i \in \text{Class 1}} (\mathbf{v}^t (\mathbf{x}_i - \boldsymbol{\mu}_1))^t (\mathbf{v}^t (\mathbf{x}_i - \boldsymbol{\mu}_1)) \\ &= \sum_{y_i \in \text{Class 1}} ((\mathbf{x}_i - \boldsymbol{\mu}_1)^t \mathbf{v})^t ((\mathbf{x}_i - \boldsymbol{\mu}_1)^t \mathbf{v}) \\ &= \sum_{y_i \in \text{Class 1}} \mathbf{v}^t (\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^t \mathbf{v} = \mathbf{v}^t \mathbf{S}_1 \mathbf{v}\end{aligned}$$

Fisher Linear Discriminant Derivation

- Similarly $\tilde{\mathbf{s}}_2^2 = \mathbf{v}^t \mathbf{S}_2 \mathbf{v}$
- Therefore $\tilde{\mathbf{s}}_1^2 + \tilde{\mathbf{s}}_2^2 = \mathbf{v}^t \mathbf{S}_1 \mathbf{v} + \mathbf{v}^t \mathbf{S}_2 \mathbf{v} = \mathbf{v}^t \mathbf{S}_W \mathbf{v}$
- Define between class scatter matrix
$$\mathbf{S}_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t$$
- \mathbf{S}_B measures separation between the means of two classes (before projection)
- Let's rewrite the separations of the projected means
$$\begin{aligned}(\tilde{\mu}_1 - \tilde{\mu}_2)^2 &= (\mathbf{v}^t \mu_1 - \mathbf{v}^t \mu_2)^2 \\&= \mathbf{v}^t (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t \mathbf{v} \\&= \mathbf{v}^t \mathbf{S}_B \mathbf{v}\end{aligned}$$

Fisher Linear Discriminant Derivation

- Thus our objective function can be written:

$$J(\mathbf{v}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{\mathbf{s}}_1^2 + \tilde{\mathbf{s}}_2^2} = \frac{\mathbf{v}^t \mathbf{S}_B \mathbf{v}}{\mathbf{v}^t \mathbf{S}_W \mathbf{v}}$$

- Minimize $J(\mathbf{v})$ by taking the derivative w.r.t. \mathbf{v} and setting it to 0

$$\begin{aligned}\frac{d}{d\mathbf{v}} J(\mathbf{v}) &= \frac{\left(\frac{d}{d\mathbf{v}} \mathbf{v}^t \mathbf{S}_B \mathbf{v} \right) \mathbf{v}^t \mathbf{S}_W \mathbf{v} - \left(\frac{d}{d\mathbf{v}} \mathbf{v}^t \mathbf{S}_W \mathbf{v} \right) \mathbf{v}^t \mathbf{S}_B \mathbf{v}}{(\mathbf{v}^t \mathbf{S}_W \mathbf{v})^2} \\&= \frac{(2\mathbf{S}_B \mathbf{v})\mathbf{v}^t \mathbf{S}_W \mathbf{v} - (2\mathbf{S}_W \mathbf{v})\mathbf{v}^t \mathbf{S}_B \mathbf{v}}{(\mathbf{v}^t \mathbf{S}_W \mathbf{v})^2} = 0\end{aligned}$$

Fisher Linear Discriminant Derivation

- Need to solve $\mathbf{v}^t \mathbf{S}_W \mathbf{v} (\mathbf{S}_B \mathbf{v}) - \mathbf{v}^t \mathbf{S}_B \mathbf{v} (\mathbf{S}_W \mathbf{v}) = 0$

$$\Rightarrow \frac{\mathbf{v}^t \mathbf{S}_W \mathbf{v} (\mathbf{S}_B \mathbf{v})}{\mathbf{v}^t \mathbf{S}_W \mathbf{v}} - \frac{\mathbf{v}^t \mathbf{S}_B \mathbf{v} (\mathbf{S}_W \mathbf{v})}{\mathbf{v}^t \mathbf{S}_W \mathbf{v}} = 0$$

$$\Rightarrow \mathbf{S}_B \mathbf{v} - \frac{\mathbf{v}^t \mathbf{S}_B \mathbf{v} (\mathbf{S}_W \mathbf{v})}{\mathbf{v}^t \mathbf{S}_W \mathbf{v}} = \lambda$$

$$\Rightarrow \underbrace{\mathbf{S}_B \mathbf{v}}_{\lambda \mathbf{S}_W \mathbf{v}} = \lambda \mathbf{S}_W \mathbf{v}$$

generalized eigenvalue problem

Fisher Linear Discriminant Derivation

$$\mathbf{S}_B \mathbf{v} = \lambda \mathbf{S}_W \mathbf{v}$$

- If \mathbf{S}_W has full rank (the inverse exists), can convert this to a standard eigenvalue problem

$$\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{v} = \lambda \mathbf{v}$$

- But $\mathbf{S}_B \mathbf{x}$ for any vector \mathbf{x} , points in the same direction as $\mu_1 - \mu_2$

$$\mathbf{S}_B \mathbf{x} = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t \mathbf{x} = (\mu_1 - \mu_2) \underbrace{(\mu_1 - \mu_2)^t \mathbf{x}}_{\alpha} = \alpha(\mu_1 - \mu_2)$$

- Thus can solve the eigenvalue problem immediately

$$\boxed{\mathbf{v} = \mathbf{S}_W^{-1}(\mu_1 - \mu_2)}$$

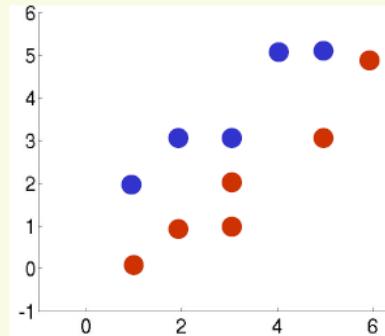
$$\mathbf{S}_W^{-1} \mathbf{S}_B \underbrace{[\mathbf{S}_W^{-1}(\mu_1 - \mu_2)]}_{\mathbf{v}} = \mathbf{S}_W^{-1} [\alpha(\mu_1 - \mu_2)] = \underbrace{\alpha [\mathbf{S}_W^{-1}(\mu_1 - \mu_2)]}_{\lambda \mathbf{v}}$$

Fisher Linear Discriminant Example

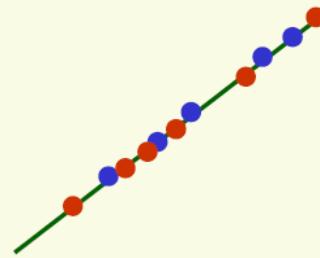
- Data
 - Class 1 has 5 samples $\mathbf{c}_1 = [(1,2), (2,3), (3,3), (4,5), (5,5)]$
 - Class 2 has 6 samples $\mathbf{c}_2 = [(1,0), (2,1), (3,1), (3,2), (5,3), (6,5)]$

- Arrange data in 2 separate matrices

$$\mathbf{c}_1 = \begin{bmatrix} 1 & 2 \\ \vdots & \vdots \\ 5 & 5 \end{bmatrix} \quad \mathbf{c}_2 = \begin{bmatrix} 1 & 0 \\ \vdots & \vdots \\ 6 & 5 \end{bmatrix}$$



- Notice that PCA performs very poorly on this data because the direction of largest variance is not helpful for classification



Fisher Linear Discriminant Example

- First compute the mean for each class

$$\mu_1 = \text{mean}(\mathbf{c}_1) = [3 \ 3.6] \quad \mu_2 = \text{mean}(\mathbf{c}_2) = [3.3 \ 2]$$

- Compute scatter matrices \mathbf{S}_1 and \mathbf{S}_2 for each class

$$\mathbf{S}_1 = 4 * \text{cov}(\mathbf{c}_1) = \begin{bmatrix} 10 & 8.0 \\ 8.0 & 7.2 \end{bmatrix} \quad \mathbf{S}_2 = 5 * \text{cov}(\mathbf{c}_2) = \begin{bmatrix} 17.3 & 16 \\ 16 & 16 \end{bmatrix}$$

- Within the class scatter:

$$\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2 = \begin{bmatrix} 27.3 & 24 \\ 24 & 23.2 \end{bmatrix}$$

- it has full rank, don't have to solve for eigenvalues

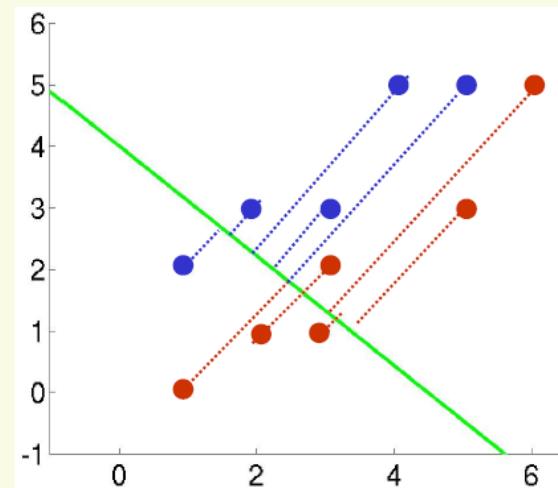
- The inverse of \mathbf{S}_w is $\mathbf{S}_w^{-1} = \text{inv}(\mathbf{S}_w) = \begin{bmatrix} 0.39 & -0.41 \\ -0.41 & 0.47 \end{bmatrix}$

- Finally, the optimal line direction \mathbf{v}

$$\mathbf{v} = \mathbf{S}_w^{-1}(\mu_1 - \mu_2) = \begin{bmatrix} -0.79 \\ 0.89 \end{bmatrix}$$

Fisher Linear Discriminant Example

- Notice, as long as the line has the right direction, its exact position does not matter
- Last step is to compute the actual **1D** vector \mathbf{y} . Let's do it separately for each class

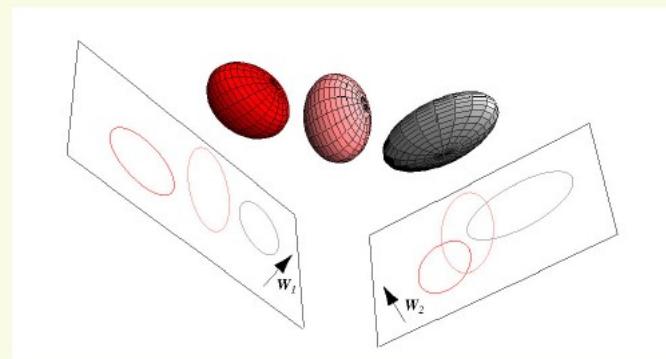


$$Y_1 = \mathbf{v}^t \mathbf{c}_1^t = [-0.65 \quad 0.73] \begin{bmatrix} 1 & \dots & 5 \\ 2 & \dots & 5 \end{bmatrix} = [0.81 \dots 0.4]$$

$$Y_2 = \mathbf{v}^t \mathbf{c}_2^t = [-0.65 \quad 0.73] \begin{bmatrix} 1 & \dots & 6 \\ 0 & \dots & 5 \end{bmatrix} = [-0.65 \dots -0.25]$$

Multiple Discriminant Analysis (MDA)

- Can generalize FLD to multiple classes
- In case of c classes, can reduce dimensionality to 1, 2, 3, ..., $c-1$ dimensions
- Project sample \mathbf{x}_i to a linear subspace $\mathbf{y}_i = \mathbf{V}^t \mathbf{x}_i$
 - \mathbf{V} is called projection matrix



Multiple Discriminant Analysis (MDA)

- Let
 - n_i by the number of samples of class i
 - and μ_i be the sample mean of class i
 - μ be the total mean of all samples

$$\mu_i = \frac{1}{n_i} \sum_{x \in \text{class } i} x \quad \mu = \frac{1}{n} \sum_{x_i} x_i$$

- Objective function: $J(V) = \frac{\det(V^t S_B V)}{\det(V^t S_W V)}$

- within the class scatter matrix S_W is

$$S_W = \sum_{i=1}^c S_i = \sum_{i=1}^c \sum_{x_k \in \text{class } i} (x_k - \mu_i)(x_k - \mu_i)^t$$

- between the class scatter matrix S_B is

$$S_B = \sum_{i=1}^c n_i (\mu_i - \mu)(\mu_i - \mu)^t$$

maximum rank is $c-1$

Multiple Discriminant Analysis (MDA)

$$J(V) = \frac{\det(V^t S_B V)}{\det(V^t S_W V)}$$

- First solve the **generalized eigenvalue** problem:

$$S_B v = \lambda S_W v$$

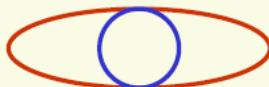
- At most $c-1$ distinct solution eigenvalues
- Let v_1, v_2, \dots, v_{c-1} be the corresponding eigenvectors
- The optimal projection matrix V to a subspace of dimension k is given by the eigenvectors corresponding to the largest k eigenvalues
- Thus can project to a subspace of dimension at most $c-1$

FDA and MDA Drawbacks

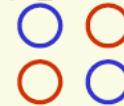
- Reduces dimension only to $k = c - 1$ (unlike PCA)
 - For complex data, projection to even the best line may result in unseparable projected samples

- Will fail:

1. $J(v)$ is always 0: happens if $\mu_1 = \mu_2$



PCA performs reasonably well here:



PCA also fails:



2. If $J(v)$ is always large: classes have large overlap when projected to any line (PCA will also fail)



Recognition with FLD

- Similar to “eigenfaces”
- Compute within-class and between-class scatter matrices
- Solve generalized eigenvector problem

$$S_i = \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T \quad \text{Project to LDA subspace and classify by nearest neighbor}$$

$$S_W = \sum_{i=1}^c S_i \quad S_B = \sum_{i=1}^c N_i (\mu_i - \bar{\mu})(\mu_i - \bar{\mu})^T$$

$$W_{opt} = \arg \max_w \frac{|W^T S_B W|}{|W^T S_W W|} \quad S_B w_i = \lambda_i S_W w_i \quad i = 1, \dots, m$$

$$\hat{x} = {W_{opt}}^T x$$

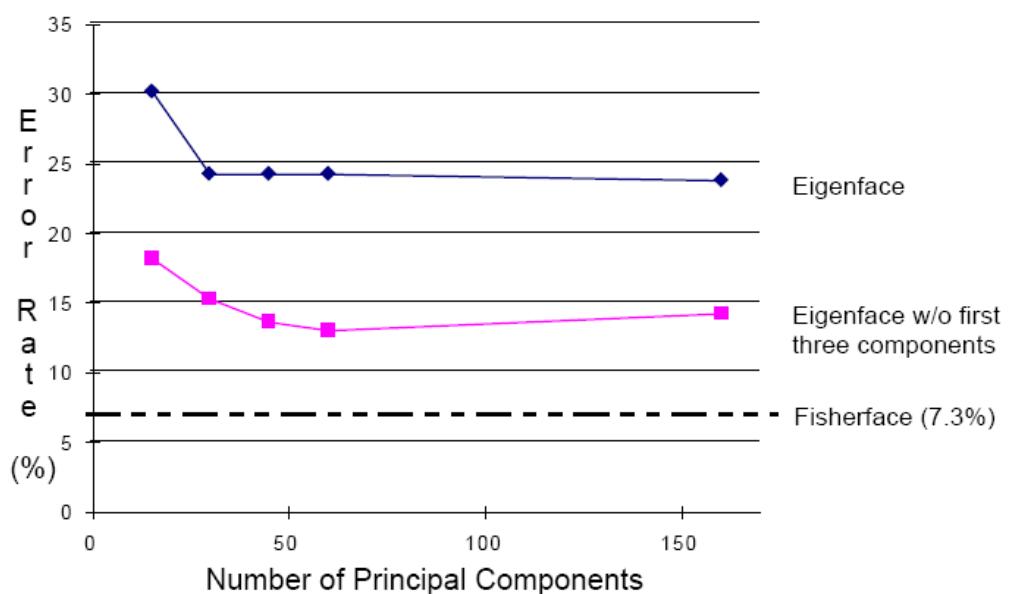
Results: Eigenface vs. Fisherface

- Input: 160 images of 16 people
- Train: 159 images
- Test: 1 image
- Variation in Facial Expression, Eyewear, and Lighting



Reference: [Eigenfaces vs. Fisherfaces, Belhumeur et al., PAMI 1997](#)

Eigenfaces vs. Fisherfaces



Reference: [Eigenfaces vs. Fisherfaces, Belhumeur et al., PAMI 1997](#)

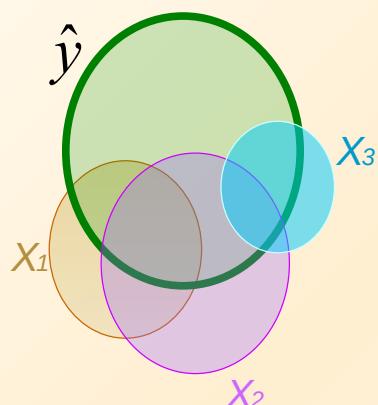
Things to remember

- PCA is a generally useful dimensionality reduction technique
 - But not ideal for discrimination
- FLD better for discrimination, though only ideal under Gaussian data assumptions
- Computer face recognition works very well under controlled environments – still room for improvement in general conditions

Linear Regression Analysis

Regression Analysis

$$\hat{y} = a + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_kx_k + \varepsilon$$



Simple and Multiple Regression Analysis

What does regression analysis do?

- ❖ Examines whether changes/differences in values of one variable (**dependent variable Y**) are linked to changes/differences in values of one or more other variables (**independent variables X_1 , X_2 , etc.**), **while controlling** for the changes in values of all other Xs.
 - ❖ E.g., Relationship between salary and gender for people who have the same levels of education, work experience, position level, seniority, etc.
- ❖ The DV (Y) must be metric.
- ❖ The IVs (Xs) must be either metric or dummy var.
- ❖ **Central Question Addressed:**
 - ❖ Is Y a function of X_1 , X_2 , etc.? How ?
 - ❖ Is there a relationship between Y and X_1 , X_2 , etc., (in each case, after controlling for the effects of all other Xs)? In what way?
 - ❖ What is the relative impact of each X on Y, holding all other Xs constant (that is, all other Xs being equal)?



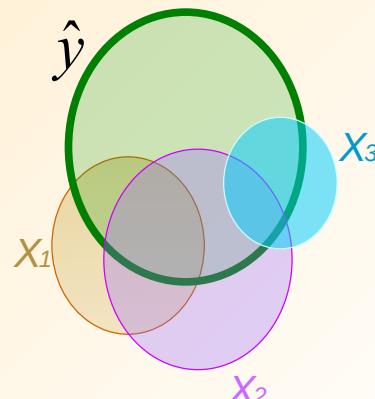
Simple and Multiple Regression Analysis

More specifically,

- ❖ Do values of Y tend to increase/decrease as values of X_1 , X_2 , etc. increase/decrease?

If so,

- ❖ By how much?
And
- ❖ How strong is the connection/relationship between Xs and Y?
 - what % of differences/variations in Y values (e.g., income) among study subjects can be explained by (or attributed to) differences in X values (e.g. years of education, years of experience, etc.)?



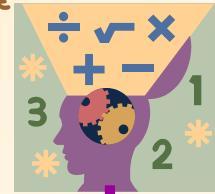
Simple and Multiple Regression Analysis

- ❖ NOTE: Once we can determine how values of Y change as a function of values of X_1, X_2 , etc., we will also be able to **predict/estimate** the value of Y from specific values of X_1, X_2 , etc.

$$Y = a + b_1 x_1 + b_2 x_2 + b_3 x_3 + \dots + b_k x_k + \epsilon$$

- ❖ Therefore, regression analysis, in a sense, is about ESTIMATING values of Y, using information about values of Xs:

- ❖ Estimation, by definition, involves?
 - ❖ To minimize error in estimation.
 - ❖ Or, to compute estimates that are as close to the true/actual values as possible.
- ❖ The objective?
 - ❖ To minimize error in estimation.
 - ❖ Or, to compute estimates that are as close to the true/actual values as possible.



Simple and Multiple Regression Analysis

QUESTION: What is the simplest way to obtain an estimate for some population characteristic (e.g., number of credit cards per U.S. household)?



ANSWER:

1. Select a representative sample from the population and
2. Compute the mean for that sample (e.g., compute the average number of CCs for the sample households).



Regression analysis can be viewed as a technique that often significantly improves the accuracy of estimation results relative to using the mean value.

So, suppose we were to estimate the **number of credit cards** for U.S. households, based on information from a random sample of, say, $n = 8$ families.

Simple and Multiple Regression Analysis

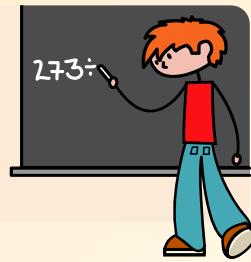
Estimating Number of Credit Cards*

i Family Number	y_i Actual # of Credit Cards
1	4
2	6
3	6
4	7
5	8
6	7
7	8
8	10

$$\sum Y_i = 56$$

\hat{y} = Estimate?

$$\hat{y} = \bar{y} = \frac{56}{8} = 7$$



QUESTION: Can we determine how much error in estimation we are committing by using as our estimate, for each of these households?

$$\bar{Y} = 7$$

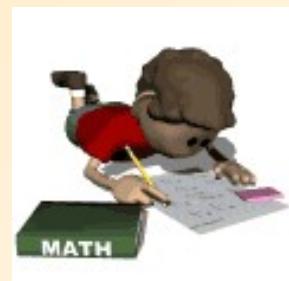
* This example was adopted from Hair, Black, Babin, Anderson, & Tatham, (2006). *Multivariate Data Analysis*, 6th ed., Prentice Hall.

Simple and Multiple Regression Analysis

Estimating Number of Credit Cards

i Family Number	y_i Actual # of Credit Cards	$\hat{y} = \bar{y}$ Estimate for # of Credit Cards	Error in Estimation
1	4	7	?
2	6	7	?
3	6	7	?
4	7	7	?
5	8	7	?
6	7	7	?
7	8	7	?
8	10	7	?

$$\sum y_i = 56 \quad \hat{y} = \bar{y} = \frac{56}{8} = 7$$



Simple and Multiple Regression Analysis

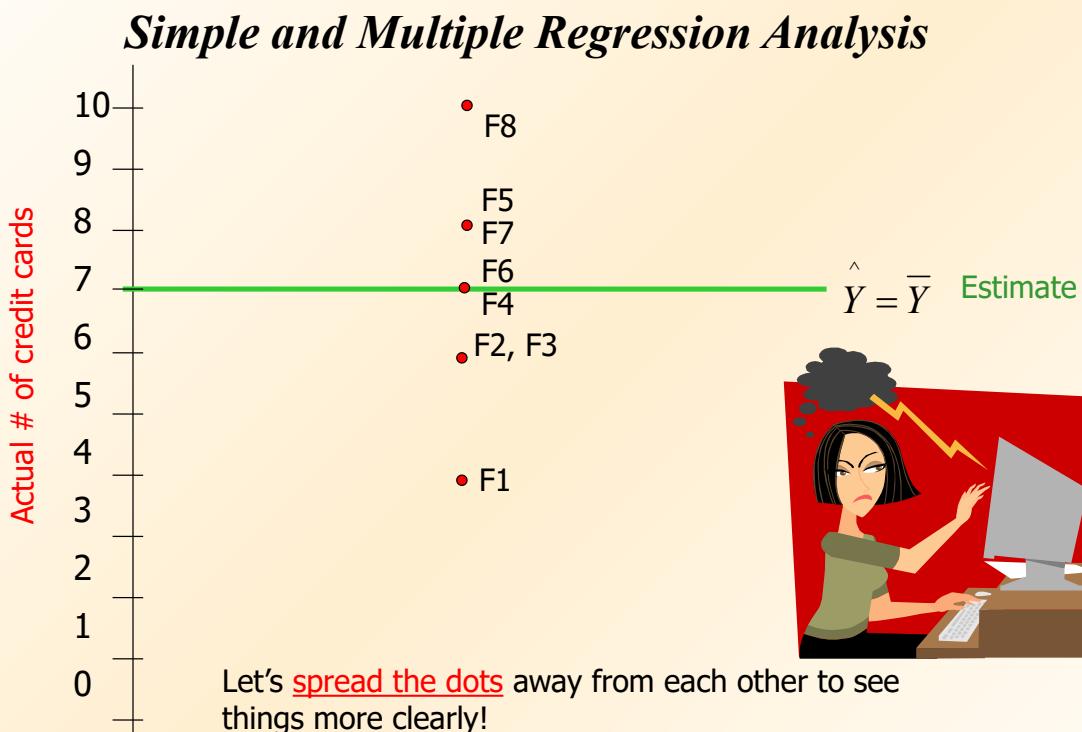
Estimating Number of Credit Cards

i Family Number	y_i Actual # of Credit Cards	$\hat{y} = \bar{y}$ Estimate for # of Credit Cards	$y_i - \bar{y}$ Error in Estimation
1	4	7	-3
2	6	7	-1
3	6	7	-1
4	7	7	0
5	8	7	+1
6	7	7	0
7	8	7	+1
8	10	7	+3

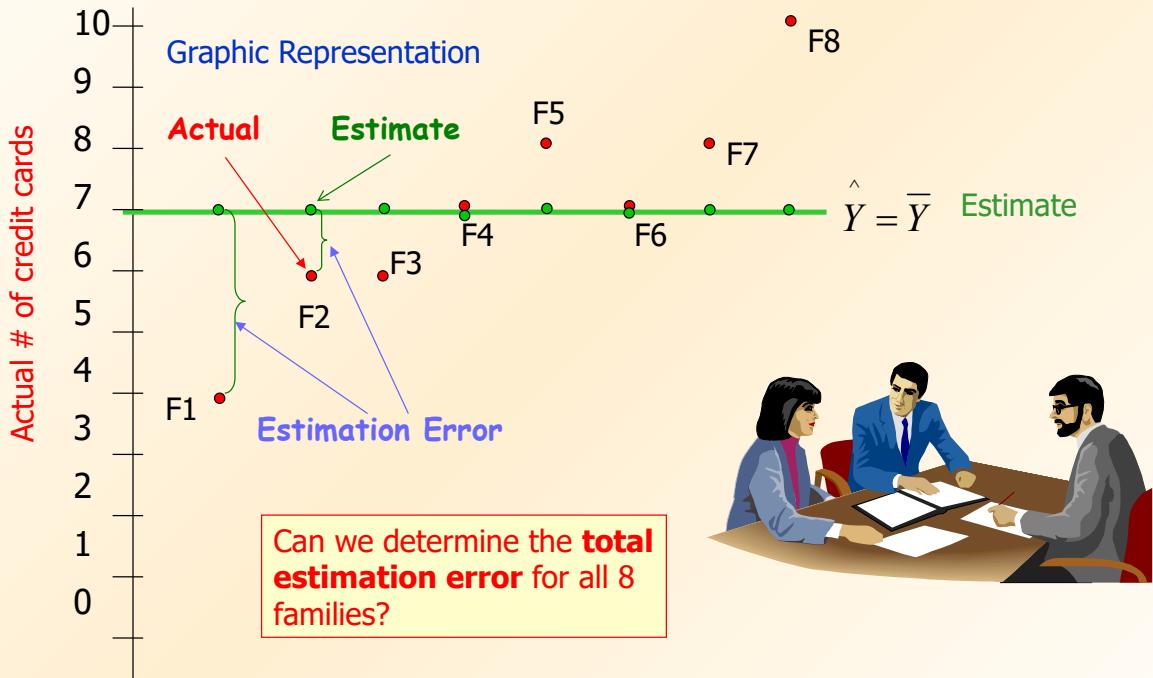


Lets now see all
this graphically

$$\sum y_i = 56 \quad \hat{y} = \bar{y} = \frac{56}{8} = 7$$



Simple and Multiple Regression Analysis



Simple and Multiple Regression Analysis

i Family Number	y_i Actual # of Credit Cards	$\hat{y} = \bar{y}$ Estimate for # of Credit Cards	$y_i - \bar{y}$ Error in Estimation
1	4	7	-3
2	6	7	-1
3	6	7	-1
4	7	7	0
5	8	7	+1
6	7	7	0
7	8	7	+1
8	10	7	+3

$\sum y_i = 56$ $\hat{y} = \bar{y} = \frac{56}{8} = 7$ $\sum (y_i - \bar{y}) = 0$



What would be the total estimation error for all 8 families combined?

Solution?

Simple and Multiple Regression Analysis

Estimating Number of Credit Cards



i Family Number	y_i Actual # of Credit Cards	$\hat{y} = \bar{y}$ Estimate for # of Credit Cards	$y_i - \bar{y}$ Error in Estimation	$(y_i - \bar{y})^2$ Errors Squared
1	4	7	-3	9
2	6	7	-1	1
3	6	7	-1	1
4	7	7	0	0
5	8	7	+1	1
6	7	7	0	0
7	8	7	+1	1
8	10	7	+3	9

$$\sum y_i = 56$$

$$\hat{y} = \bar{y} = \frac{56}{8} = 7$$

$$\sum (y_i - \bar{y}) = 0$$

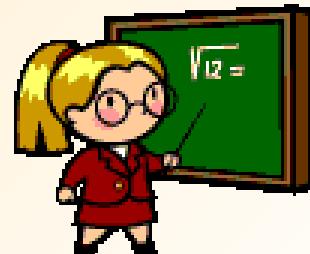
$$\sum (y_i - \bar{y})^2 = 22$$

SST = Sum of Squares Total

Simple and Multiple Regression Analysis

22 = SST = Index for total (combined) amount of estimation error for all families (observations) in the sample when using the mean as the estimate.

- ✓ SST is also the sum of squared deviations from the mean.
 - Remember the formula for computing **Variance**?
- **Objective in Estimation?**
Minimize error, maximize precision.
- **Can we cut down the amount of estimation error (SST)? How?**
Yes, we can, by using information about other variables suspected to be strong predictors (strongly related to) # of credit cards possessed by families (e.g., family size, family income, etc.)..



Simple and Multiple Regression Analysis

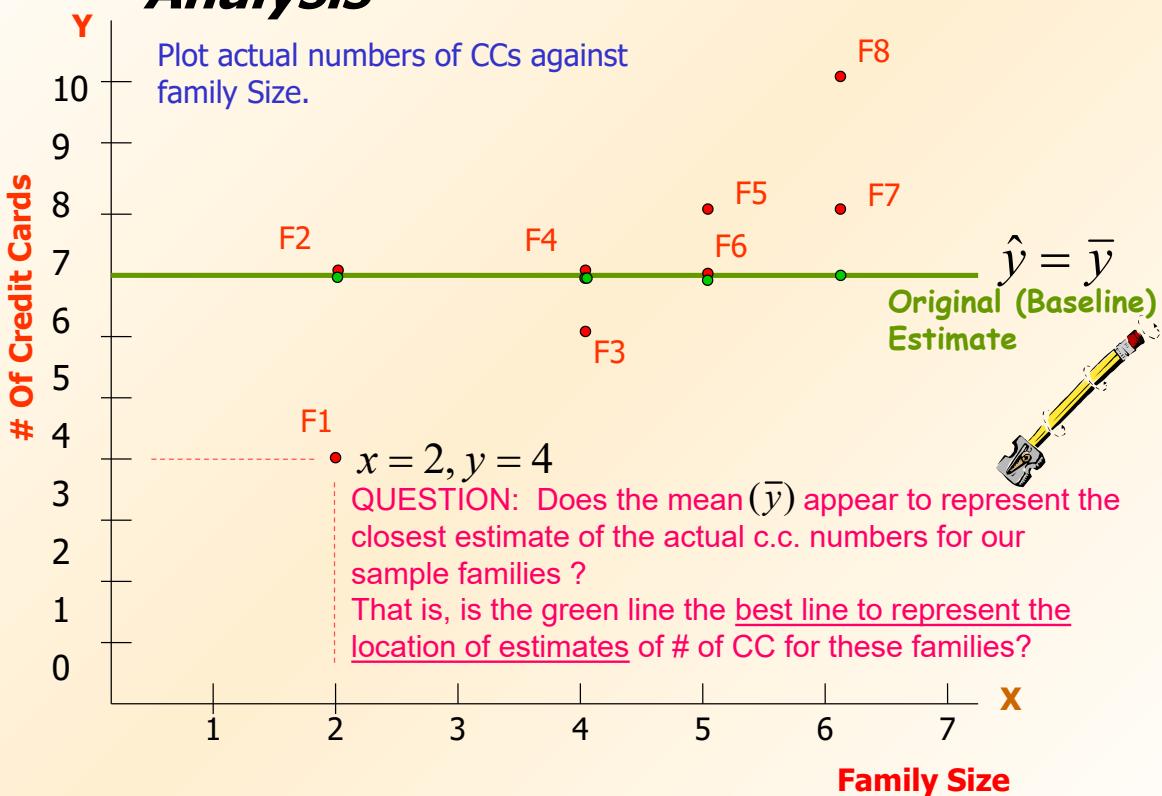
i Family Number	y Actual # of Credit Cards	x Family Size
1	4	2
2	6	2
3	6	4
4	7	4
5	8	5
6	7	5
7	8	6
8	10	6



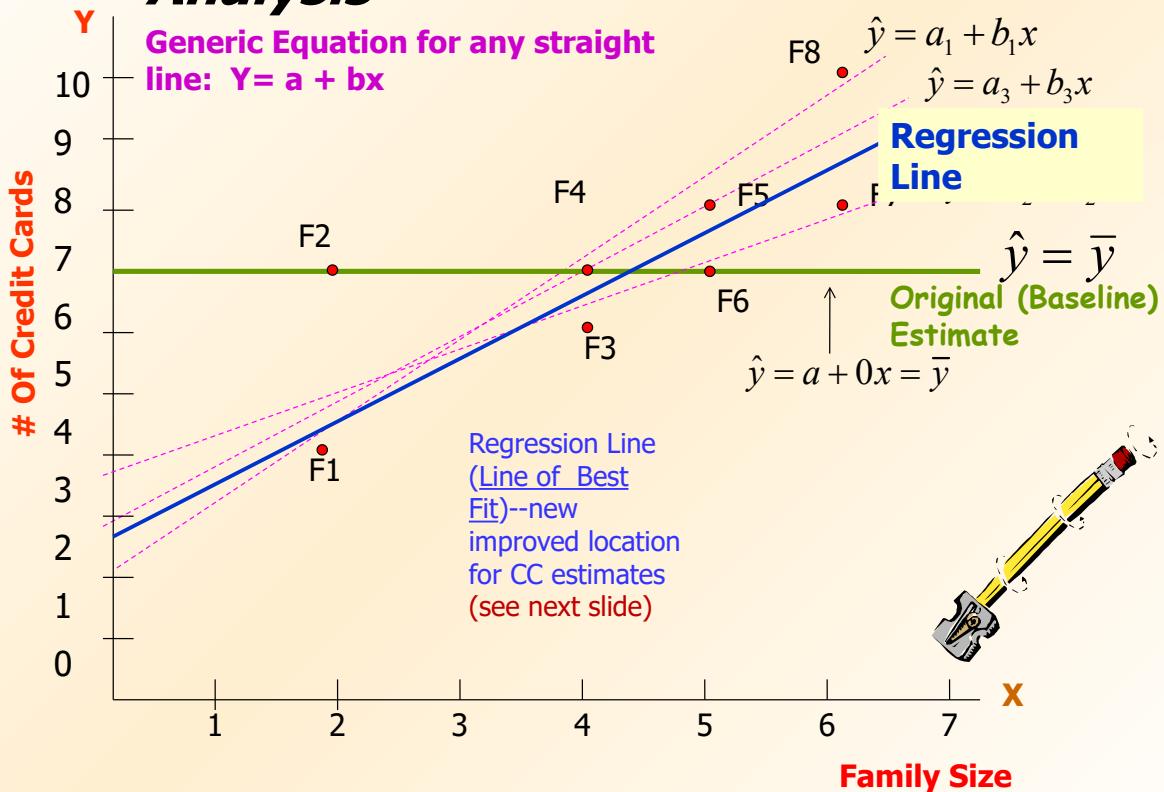
We now can attempt to estimate # of credit cards from the information on family size, rather than from its own mean.

Let's first see this graphically!

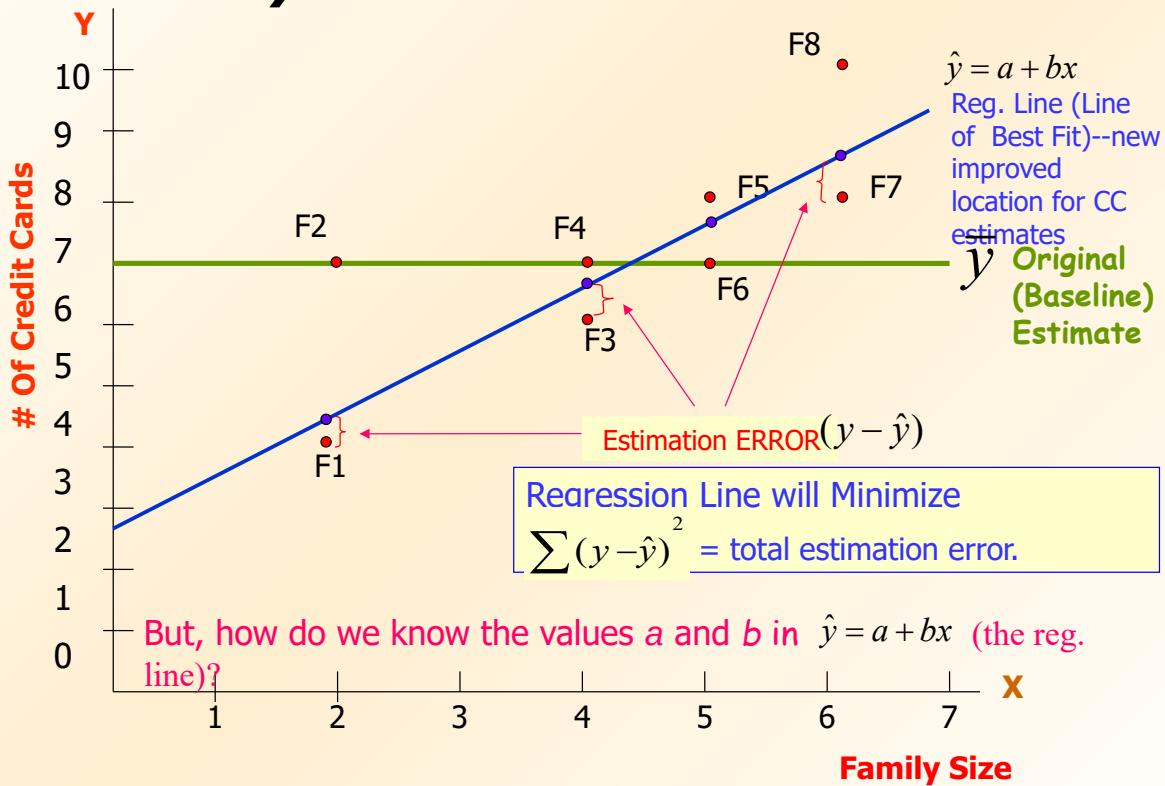
Simple and Multiple Regression Analysis



Simple and Multiple Regression Analysis



Simple and Multiple Regression Analysis



Actual # of credit cards

EQUATION FOR REGRESSION LINE (LINE OF BEST FIT)--

Values of a and b for the regression line:

$$\hat{y} = a + bx \quad \left\{ \begin{array}{l} b = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2} \\ a = \bar{y} - b\bar{x} \end{array} \right.$$



Let's use above formulas to compute the values of "a" and "b" for the regression line in our example.

We will need: \bar{y} , \bar{x} , $\sum(x - \bar{x})(y - \bar{y})$, and $\sum(x - \bar{x})^2$

Simple and Multiple Regression Analysis

We need: \bar{y} , \bar{x} , $\sum(x - \bar{x})(y - \bar{y})$, and $\sum(x - \bar{x})^2$

i Family Number	y Actual # of Credit Cards	x Family Size	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
1	4	2	?	?	?	?
2	6	2	?	?	?	?
3	6	4	?	?	?	?
4	7	4	?	?	?	?
5	8	5	?	?	?	?
6	7	5	?	?	?	?
7	8	6	?	?	?	?
8	10	6	?	?	?	?

$$\bar{Y} = \frac{56}{8} = 7 \quad \bar{x} = \frac{34}{8} = 4.25$$

$$\sum(x - \bar{x})(y - \bar{y}) = ? \quad \sum(x - \bar{x})^2 = ?$$

Simple and Multiple Regression Analysis

We need: \bar{y} , \bar{x} , $\sum(x - \bar{x})(y - \bar{y})$, and $\sum(x - \bar{x})^2$

i Family Number	y Actual # of Credit Cards	x Family Size	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
1	4	2	-2.25	-3	6.75	5.0625
2	6	2	-2.25	-1	2.25	5.0625
3	6	4	-.25	-1	.25	.0625
4	7	4	-.25	0	0	.0625
5	8	5	.75	1	.75	.5625
6	7	5	.75	0	0	.5625
7	8	6	1.75	1	1.75	3.0625
8	10	6	1.75	3	5.25	3.0625

$$\bar{Y} = \frac{56}{8} = 7 \quad \bar{x} = \frac{34}{8} = 4.25$$

$$\sum(x - \bar{x})(y - \bar{y}) = 17 \quad \sum(x - \bar{x})^2 = 17.5$$

Simple and Multiple Regression Analysis

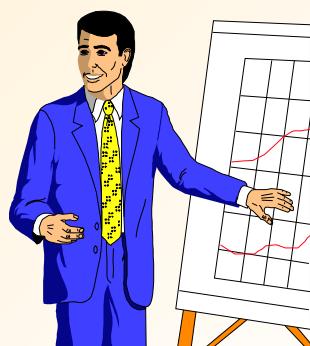
REGRESSION LINE (LINE OF BEST FIT):

$$\hat{y} = a + bx \left\{ \begin{array}{l} b = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2} = \frac{17}{17.5} = .971 \\ a = \bar{y} - b\bar{x} = 7 - .971(4.25) = 2.87 \end{array} \right.$$

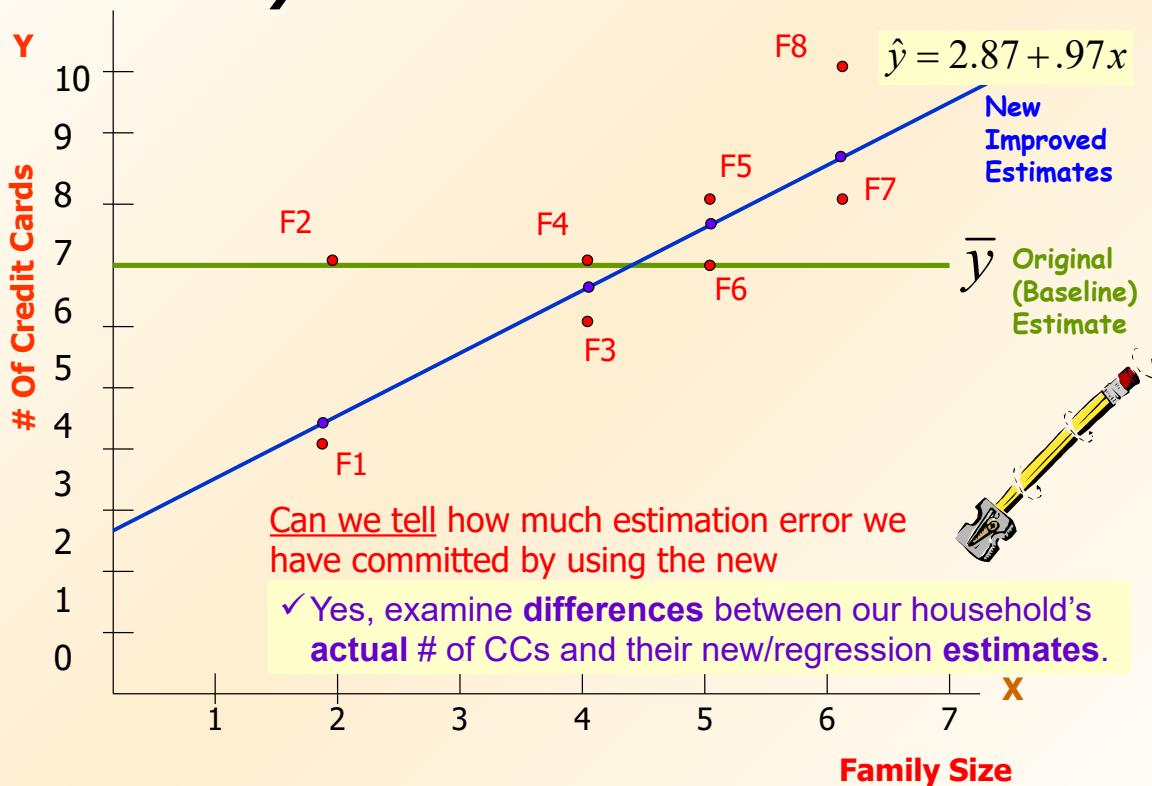
$$a = 2.87 \quad b = .97$$

$$\hat{y} = 2.87 + .97x$$

↑ ?
 Y-Intercept ↑ ?
 Regression Coefficient



Simple and Multiple Regression Analysis



Simple and Multiple Regression Analysis

i Family Number	y Actual # of Credit Cards	x Family Size	\hat{y} Regression Estimate	$y - \hat{y}$ Error (Residual)	$(y - \hat{y})^2$ Errors Squared
1	4	2	?	?	?
2	6	2	?	?	?
3	6	4	?	?	?
4	7	4	?	?	?
5	8	5	?	?	?
6	7	5	?	?	?
7	8	6	?	?	?
8	10	6	?	?	?

$\sum (y - \hat{y})^2$

Simple and Multiple Regression Analysis

$$\hat{y} = 2.87 + .97x \quad \hat{y} = 2.87 + .97(2) = 4.81$$

i Family Number	y Actual # of Credit Cards	x Family Size	\hat{y} Regression Estimate	$y - \hat{y}$ Error (Residual)	$(y - \hat{y})^2$ Errors Squared
1	4	2	4.81	-.81	.66
2	6	2	4.81	1.19	1.42
3	6	4	6.76	-.76	.58
4	7	4	6.76	.24	.06
5	8	5	7.73	.27	.07
6	7	5	7.73	-.73	.53
7	8	6	8.7	-.7	.49
8	10	6	8.7	1.3	1.69

$$5.486 = \sum (y - \hat{y})^2$$

SSE = Sum of Squares Error (SS Residual)

Simple and Multiple Regression Analysis

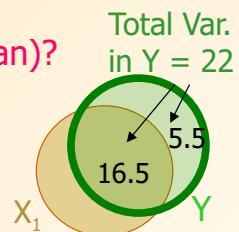
Total Baseline Error using the mean (**SS Total**) 22.0

New or Remaining Error (**SS Error** or **SS Residual**) 5.486 ~ 5.5

QUESTION: How much of the original estimation error have we explained away (eliminated) by using the regression model (instead of the mean)?

$$22 - 5.486 = 16.514 \text{ (SS Regression or SS Explained)}$$

QUESTION: What % of estimation error have we explained (eliminated by using the regression model)?



$$R^2 = 16.514 / 22 = .751 \text{ or } 75\% \text{ What is this called?}$$

that is % of differences in # of CCs among households explained by differences in their family size.

What does the remaining 25% represent?

Percent of variation (differences) in number of credit cards owned by families that can be accounted for by: (a) all other potential predictors not included in the model, beyond family size, and (b) unexplainable random/chance variations.

Simple and Multiple Regression Analysis

$$R^2 = \text{SS Regression} / \text{SS Total} = 16.5/22 = 75\%$$

R^2 is a measure of our success regarding accuracy of our estimation effort.

- ✓ R^2 = % of estimation error that we have been able to explain away by using the regression model, instead of using the mean.
- ✓ R^2 indicates how much better we can predict Y from information about Xs, rather than from using its own mean.
- ✓ R^2 = % of differences (variations) in Y values that is explained by (attributable to) differences in X values.

Note: When dealing with only two variables (a single X and Y):

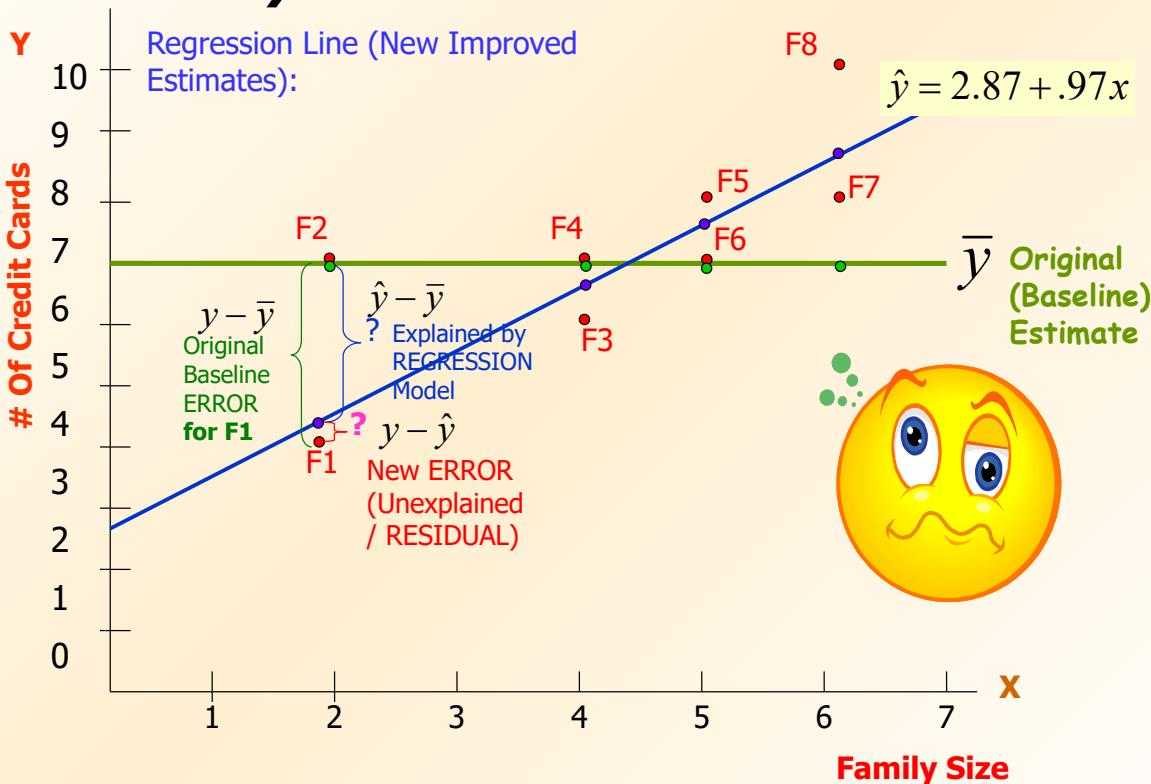
$$r = \sqrt{R^2} = \sqrt{\frac{16.5}{22}} = \sqrt{.75} = .866$$

Pearson Correlation
of Y with X_1
(NOT controlling for any
other var.)



Let's now examine all this graphically!

Simple and Multiple Regression Analysis



Simple and Multiple Regression Analysis

5.5 = SSE = The amount of estimation error for the 8 sample families when using simple regression (i.e., a regression model that includes only information about family size).

Can we reduce the amount of estimation error (SSE) to an even lower level and, thus, improving the estimation process? How?

Yes, by adding information on a second variables suspected strongly related to # of credit cards (e.g., family income-- X_2).



Simple and Multiple Regression Analysis

i Family Number	y_i Actual # of Credit Cards	x_1 Family Size	x_2 Family Income
1	4	2	14
2	6	2	16
3	6	4	14
4	7	4	17
5	8	5	18
6	7	5	21
7	8	6	17
8	10	6	25

We now can attempt to estimate # of CCs from our information on family size and family income!

Our regression model will now be a linear plane, rather than a straight line!

Generic Equation for a linear plane:

$$\hat{y} = a + b_1 x_1 + b_2 x_2$$

Let's examine the regression plane for our example graphically.



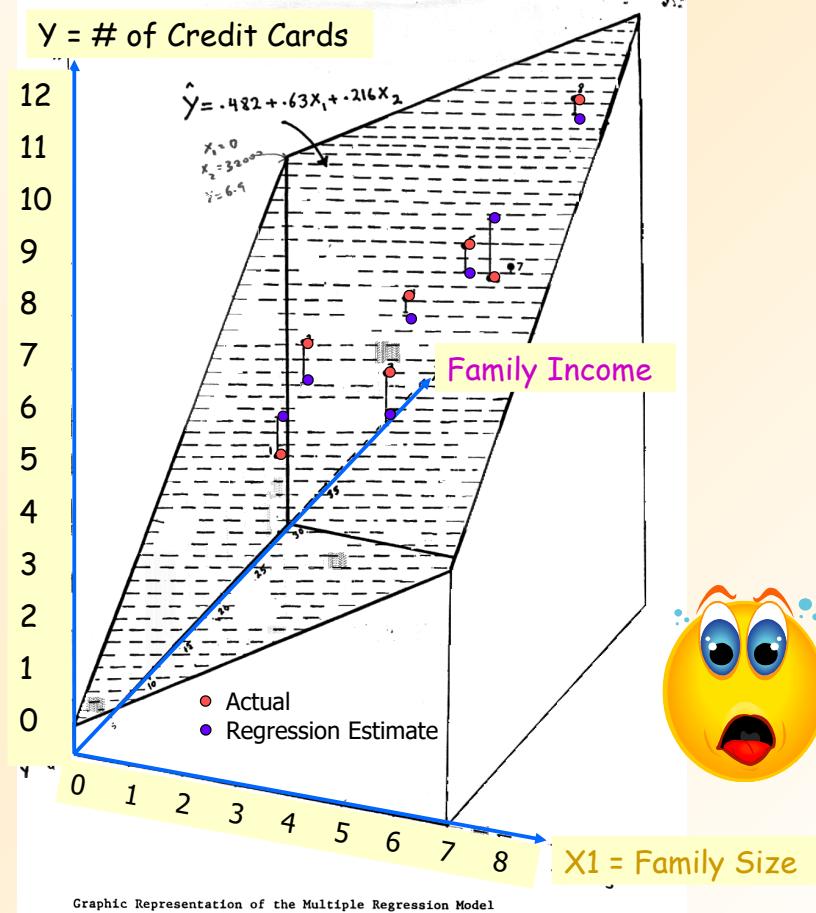
$$\hat{y} = a + b_1 x_1 + b_2 x_2$$

Formulas are available for computing values of a , b_1 and b_2

MULTIPLE REGRESSION MODEL FOR OUR EXAMPLE:

$$\hat{y} = .482 + .63x_1 + .216x_2$$

Let's now see how much error in estimation we are committing by using this multiple regression model.



Simple and Multiple Regression Analysis

$$\hat{y} = .482 + .63x_1 + .216x_2$$

i Family Number	y Actual # of Credit Cards	x_1 Family Size	x_2 Family Income (\$000)	\hat{Y} Regression Estimate	$y - \hat{y}$ Error (Residual)	$(y - \hat{y})^2$ Errors Squared
1	4	2	14	?	?	?
2	6	2	16	?	?	?
3	6	4	14	?	?	?
4	7	4	17	?	?	?
5	8	5	18	?	?	?
6	7	5	21	?	?	?
7	8	6	17	?	?	?
8	10	6	25	?	?	?

$$\sum (y - \hat{y})^2$$

Simple and Multiple Regression Analysis

$$\hat{y} = .482 + .63x_1 + .216x_2 \quad \hat{y} = .482 + .63(2) + .216(14) = 4.77$$

i Family Number	y Actual # of Credit Cards	x_1 Family Size	x_2 Family Income (\$000)	\hat{Y} Regression Estimate	$y - \hat{y}$ Error (Residual)	$(y - \hat{y})^2$ Errors Squared
1	4	2	14	4.77	-.77	.59
2	6	2	16	5.20	.80	.64
3	6	4	14	6.03	-.03	.00
4	7	4	17	6.68	.32	.10
5	8	5	18	7.53	.47	.22
6	7	5	21	8.18	-1.18	1.39
7	8	6	17	7.95	.05	.00
8	10	6	25	9.67	.33	.11

SSE = Sum of Squares Error
(Residual)

$$3.05 = \sum (y - \hat{y})^2$$

Unique (additional) contribution of X_2 (family income) beyond X_1 = ? $5.5 - 3.05 = 2.45$

Simple and Multiple Regression Analysis

The MULTIPLE REGRESSION MODEL FOR OUR EXAMPLE:

$$\hat{y} = .482 + .63x_1 + .216x_2$$



Y-Intercept, "a"

(NOTE: Only when all Xs can meaningfully take on value of zero, the intercept will have a meaningful/direct/ practical interpretation. Otherwise, it is simply an aid in increasing accuracy of estimation.

b_1 and b_2 = Regression Coefficients

0.63: Among families of the same income, an increase in family size by one person would, on average, result in .63 more credit cards.

0.21: Among families of the same size, an income increase of \$1,000, results in an average increase of 0.2 credit cards .

"b's represent effect of each X on Y when all other Xs are controlled for/held constant/taken into account

- i.e., after impacts of all other variables are accounted for (remember the high blood pressure-hearing problem connection?)

Simple and Multiple Regression Analysis

The MULTIPLE REGRESSION MODEL FOR OUR EXAMPLE:

$$\hat{y} = .482 + .63x_1 + .216x_2$$

$$SST = 22 \quad SSE = 3.05$$



What is our new R²?

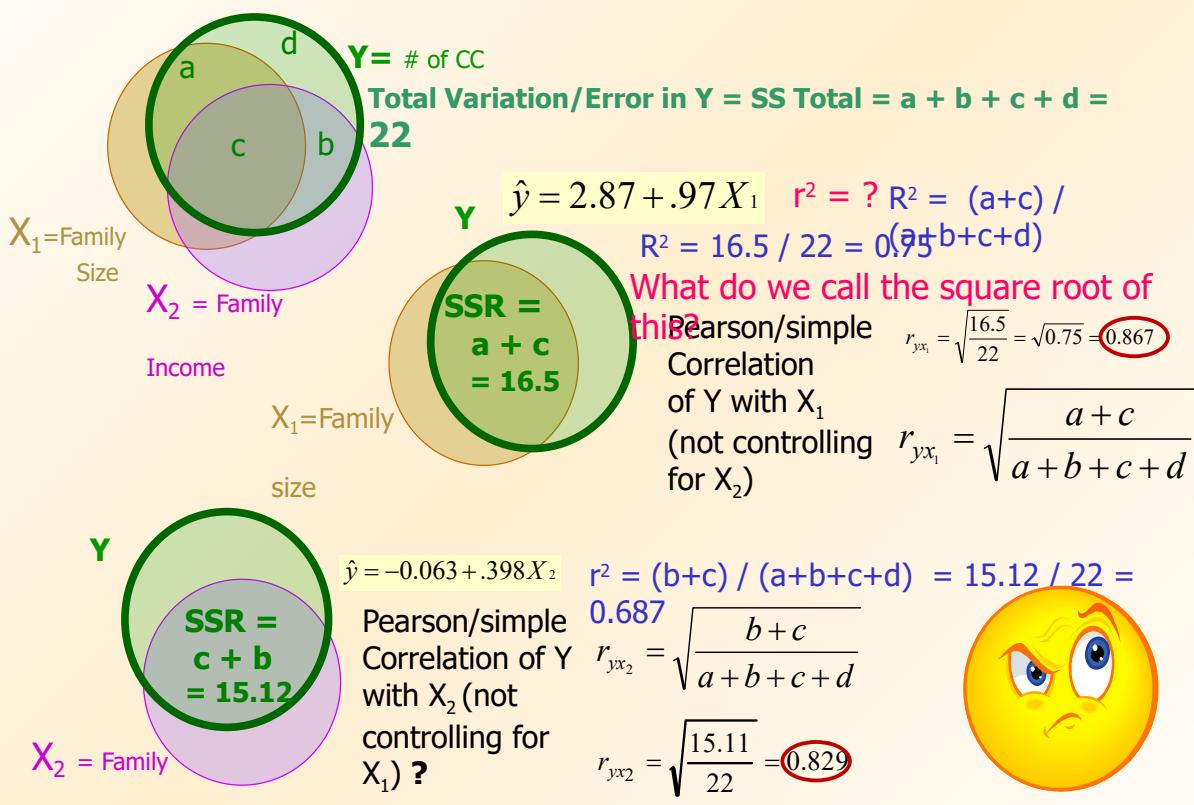
$$SS \text{ Regression} = 22 - 3.05 = 18.95$$

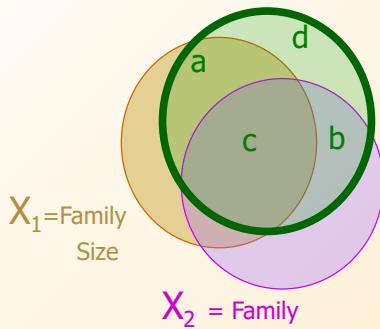
$$R^2 = 18.95 / 22 = .861 \text{ or } 86\%$$

Percent of differences in households' number of CCs that is explained by differences in family size and family income.

The Remaining 14%?
(3.05 / 22 = .14)

Percent of variation in number of credit cards that can be accounted for by (a) all other relevant factors not included in the model, beyond family size and income, and (b) unexplainable random/chance variations.





$$\hat{y} = .482 + .63x_1 + .216x_2$$

R^2 Graphically = ?

NOTE: **c** is explained by both X_1 and X_2



$$SSR = a + b + c = 18.95$$

$$SST = a + b + c + d = 22$$

$$R^2 = SSR / SST = (a + b + c) / (a + b + c + d) = 18.95 / 22 \\ = 86\%$$

SSE = ?

$$SSE = d = 22 - 18.95 = 3.05$$



Simple and Multiple Regression Analysis

$$\hat{y} = .482 + .63x_1 + .216x_2 \quad \hat{y} = .482 + .63(2) + .216(14) = 4.77$$

i Family Number	y Actual # of Credit Cards	x_1 Family Size	x_2 Family Income (\$000)	\hat{Y} Regression Estimate	$y - \hat{y}$ Error (Residual)	$(y - \hat{y})^2$ Errors Squared
1	4	2	14	4.77	-.77	.59
2	6	2	16	5.20	.80	.64
3	6	4	14	6.03	-.03	.00
4	7	4	17	6.68	.32	.10
5	8	5	18	7.53	.47	.22
6	7	5	21	8.18	-1.18	1.39
7	8	6	17	7.95	.05	.00
8	10	6	25	9.67	.33	.11

SSE = Sum of Squares Error (Residual)

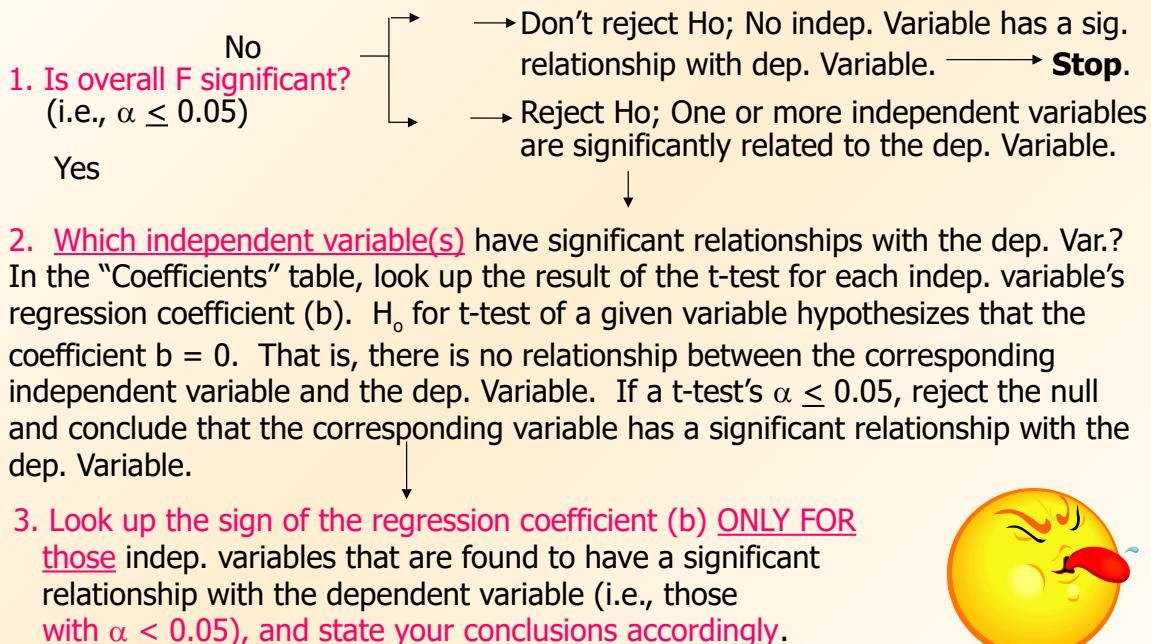
$$3.05 = \sum (y - \hat{y})^2$$

Remember:

Unique (additional) contribution of X_2 = $5.5 - 3.05 = 2.45$

Interpreting Regression Results

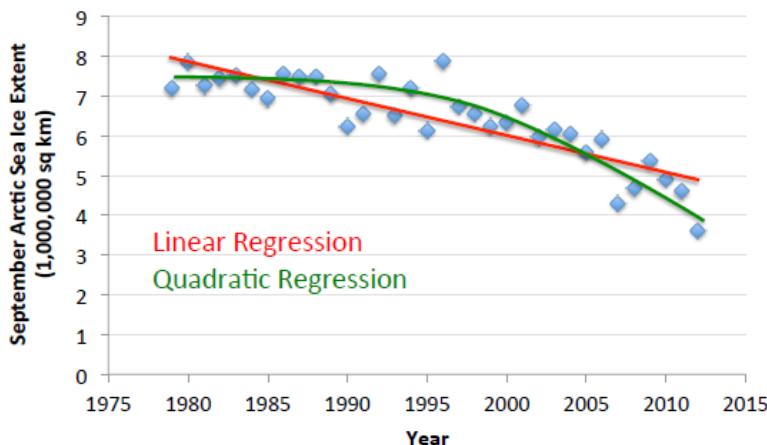
$H_0: R^2 = 0$. That is, There is **NO RELATIONSHIP** between the DV and **ANY OF the IVs included in the regression model.**



Regression

Given:

- Data $X = \{x^{(1)}, \dots, x^{(n)}\}$ where $x^{(i)} \in \mathbb{R}^d$
- Corresponding labels $y = \{y^{(1)}, \dots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}$



Prostate Cancer Dataset

- 97 samples, partitioned into 67 train / 30 test
- Eight predictors (features):
 - 6 continuous (4 log transforms), 1 binary, 1 ordinal
- Continuous outcome variable:
 - Ipsa : $\log(\text{prostate specific antigen level})$

TABLE 3.2. Linear model fit to the prostate cancer data. The Z score is the coefficient divided by its standard error (3.12). Roughly a Z score larger than two in absolute value is significantly nonzero at the $p = 0.05$ level.

Term	Coefficient	Std. Error	Z Score
Intercept	2.46	0.09	27.60
lcavol	0.68	0.13	5.37
lweight	0.26	0.10	2.75
age	-0.14	0.10	-1.40
lbph	0.21	0.10	2.06
svi	0.31	0.12	2.47
lcp	-0.29	0.15	-1.87
gleason	-0.02	0.15	-0.15
pgg45	0.27	0.15	1.74

Based on slide by Jeff Howbert

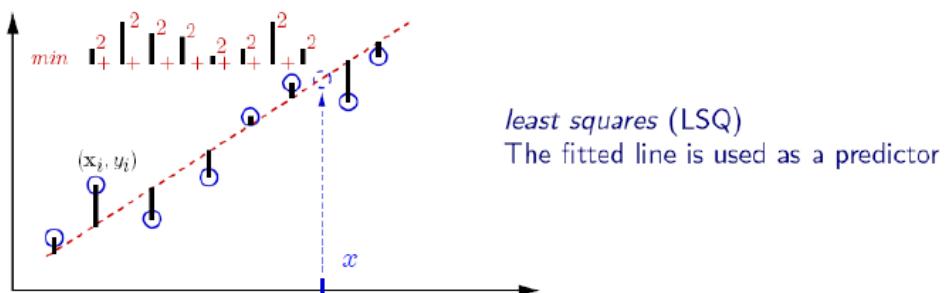
Linear Regression

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j$$

Assume $x_0 = 1$

- Fit model by minimizing sum of squared errors

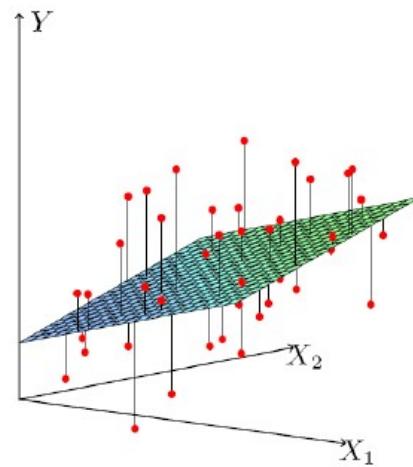
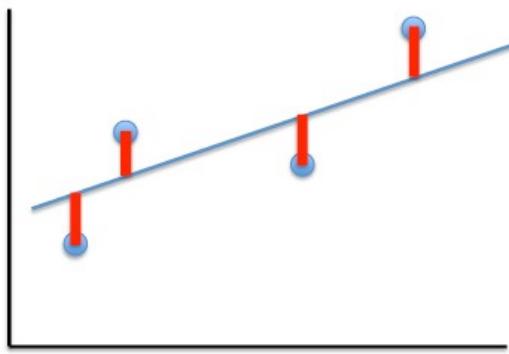


Least Squares Linear Regression

- Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta} \left(x^{(i)} \right) - y^{(i)} \right)^2$$

- Fit by solving $\min_{\theta} J(\theta)$

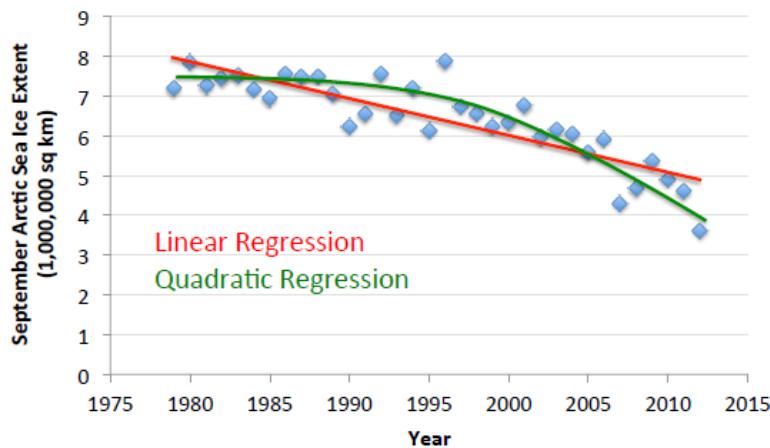


Regression Analysis: Part 2

Regression

Given:

- Data $X = \{x^{(1)}, \dots, x^{(n)}\}$ where $x^{(i)} \in \mathbb{R}^d$
- Corresponding labels $y = \{y^{(1)}, \dots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}$



Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013)

Prostate Cancer Dataset

- 97 samples, partitioned into 67 train / 30 test
- Eight predictors (features):
 - 6 continuous (4 log transforms), 1 binary, 1 ordinal
- Continuous outcome variable:
 - Ipsa : $\log(\text{prostate specific antigen level})$

TABLE 3.2. Linear model fit to the prostate cancer data. The Z score is the coefficient divided by its standard error (3.12). Roughly a Z score larger than two in absolute value is significantly nonzero at the $p = 0.05$ level.

Term	Coefficient	Std. Error	Z Score
Intercept	2.46	0.09	27.60
lcavol	0.68	0.13	5.37
lweight	0.26	0.10	2.75
age	-0.14	0.10	-1.40
lbph	0.21	0.10	2.06
svi	0.31	0.12	2.47
lcp	-0.29	0.15	-1.87
gleason	-0.02	0.15	-0.15
pgg45	0.27	0.15	1.74

Based on slide by Jeff Howbert

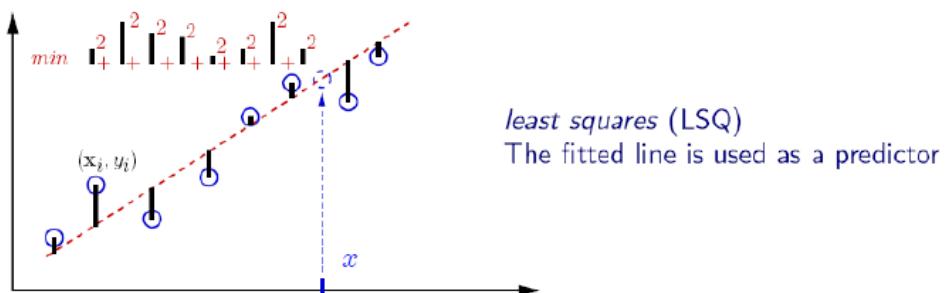
Linear Regression

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j$$

Assume $x_0 = 1$

- Fit model by minimizing sum of squared errors

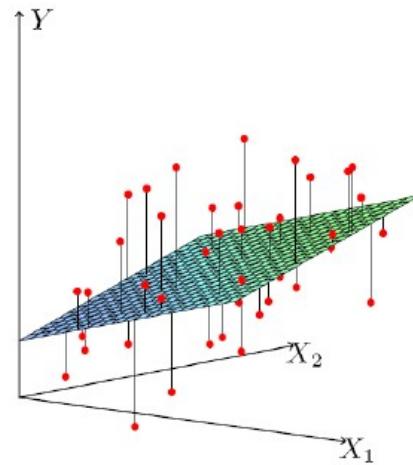
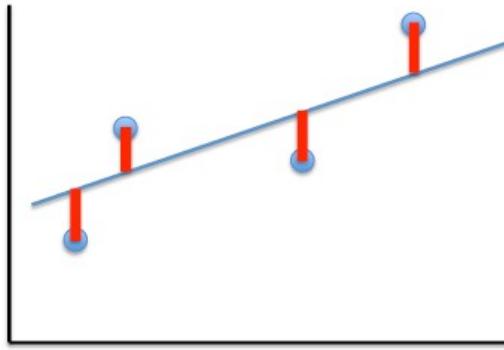


Least Squares Linear Regression

- Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

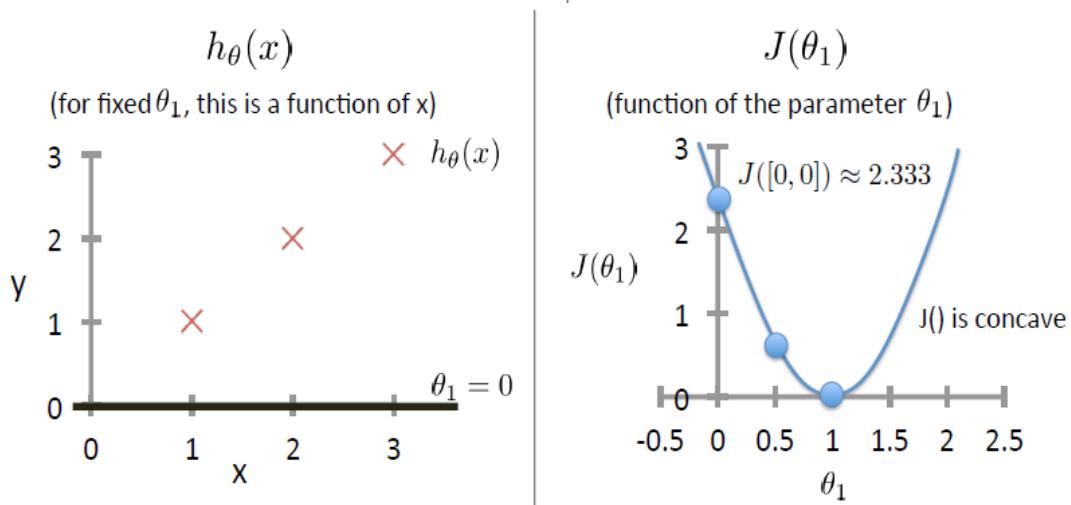
- Fit by solving $\min_{\theta} J(\theta)$



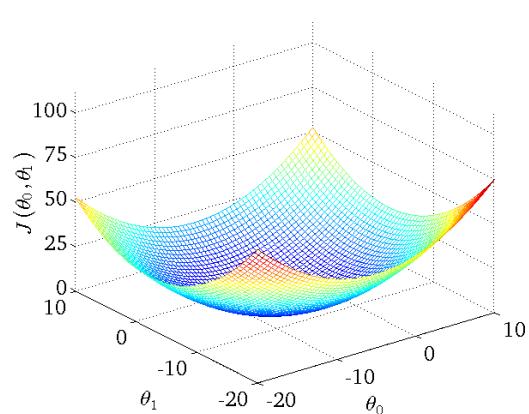
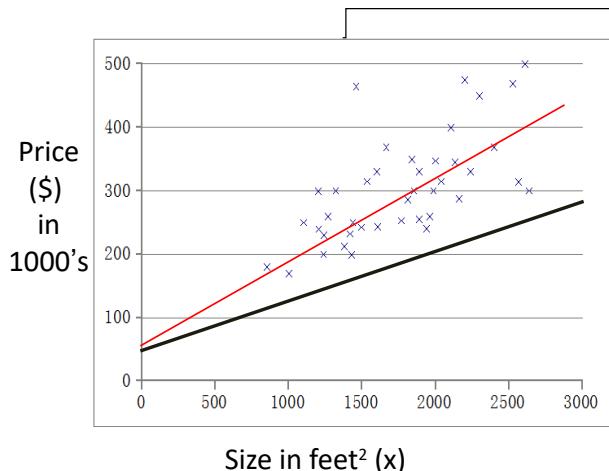
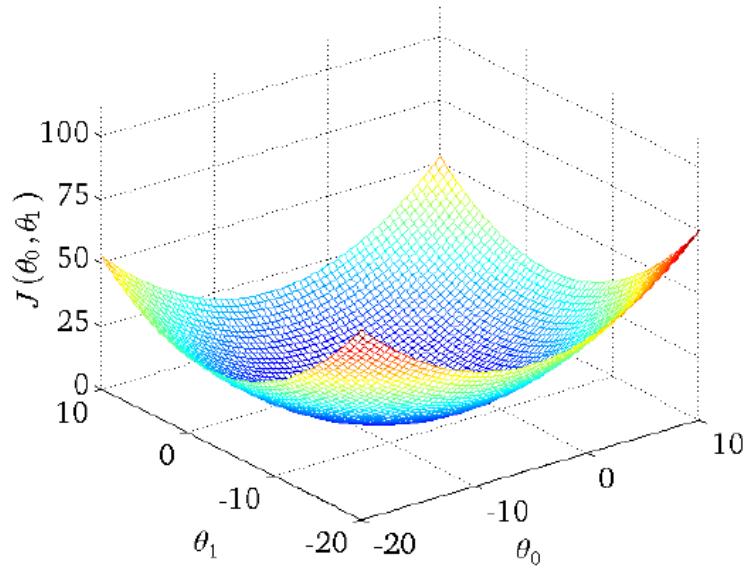
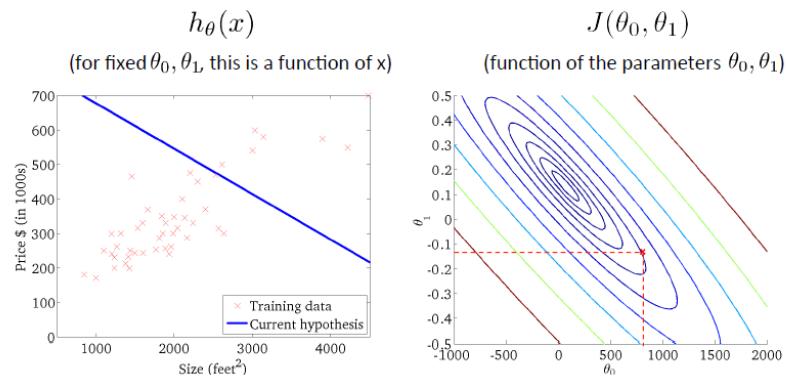
Intuition Behind Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$



Intuition Behind Cost Function



$$h_{\theta}(x) = 50 + 0.06x \quad J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$



$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

Question: How to minimize J ?

Gradient Descent

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$
until we hopefully end up at a minimum

Gradient descent algorithm

repeat until convergence {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (for $j = 0$ and $j = 1$)

}

Correct: Simultaneous update

```
temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ 
temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ 
 $\theta_0 := \text{temp0}$ 
 $\theta_1 := \text{temp1}$ 
```

Incorrect:

```
temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ 
 $\theta_0 := \text{temp0}$ 
temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ 
 $\theta_1 := \text{temp1}$ 
```

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{simultaneously update } j = 0 \text{ and } j = 1)$$

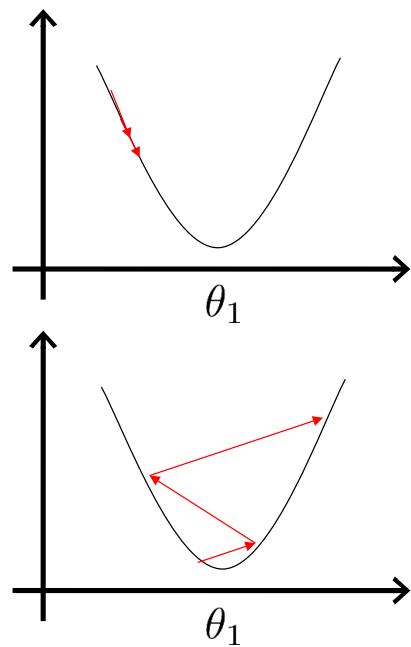
}

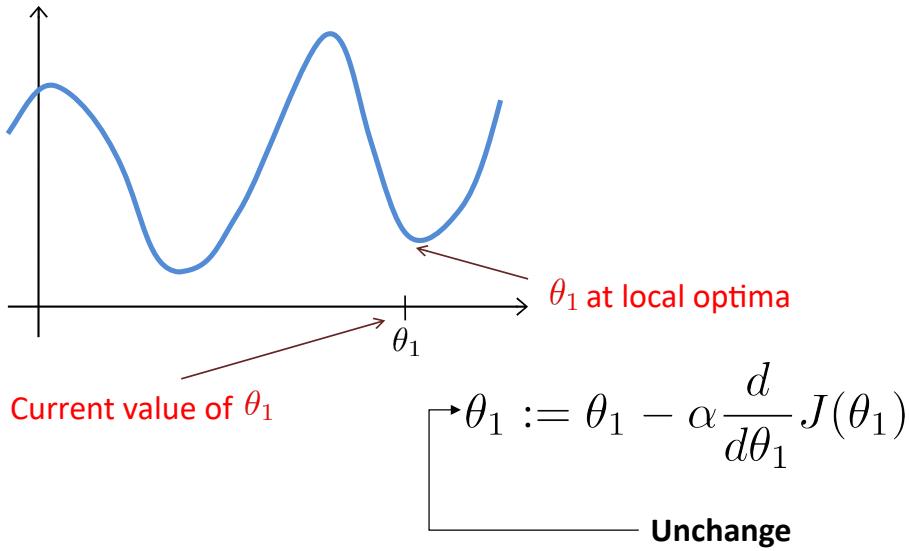
Notice : α is the learning rate.

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.





Gradient descent can converge to a local minimum, even with the learning rate α fixed.

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

Gradient Descent for Linear Regression

Gradient descent algorithm

```

repeat until convergence {
     $θ_j := θ_j - α \frac{\partial}{\partial θ_j} J(θ_0, θ_1)$ 
    (for  $j = 1$  and  $j = 0$ )
}

```

Linear Regression Model

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

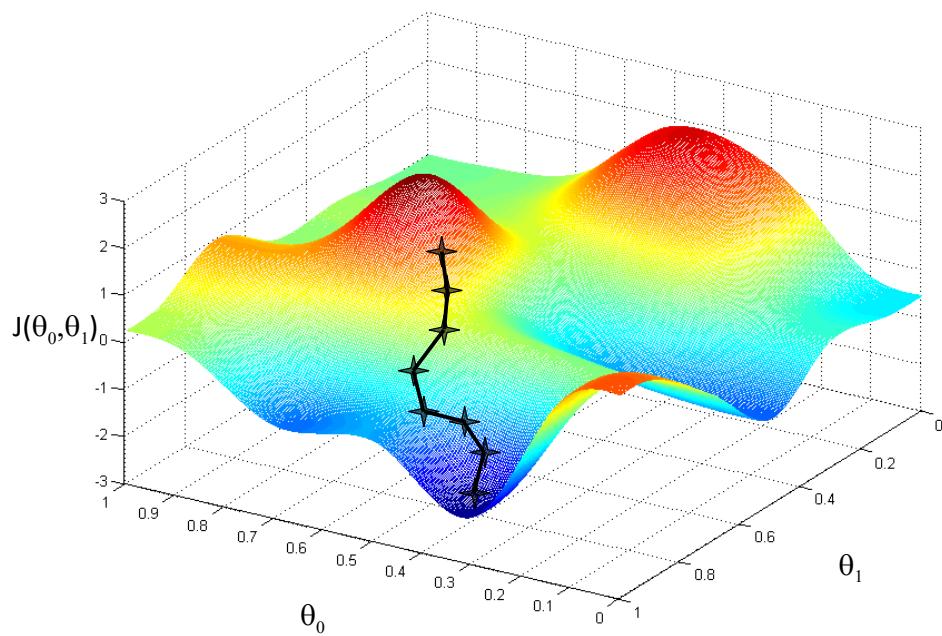
Gradient descent algorithm

repeat until convergence {

$$\left. \begin{aligned} \theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \end{aligned} \right\}$$

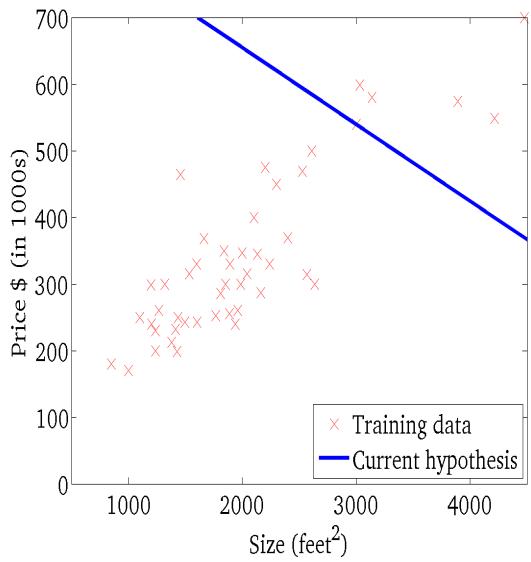
update
 θ_0 and θ_1
simultaneously

}



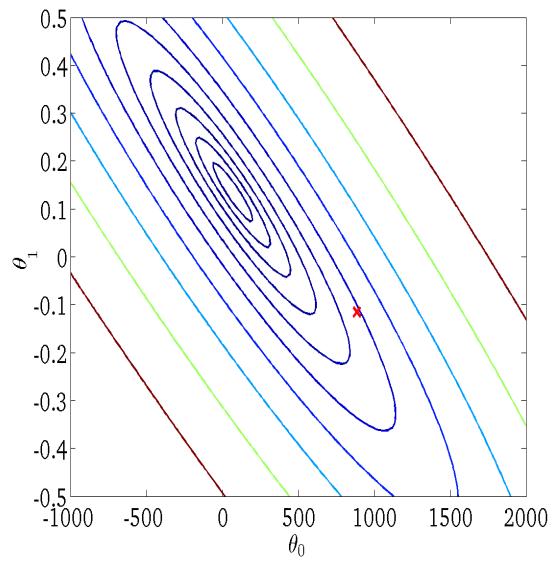
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



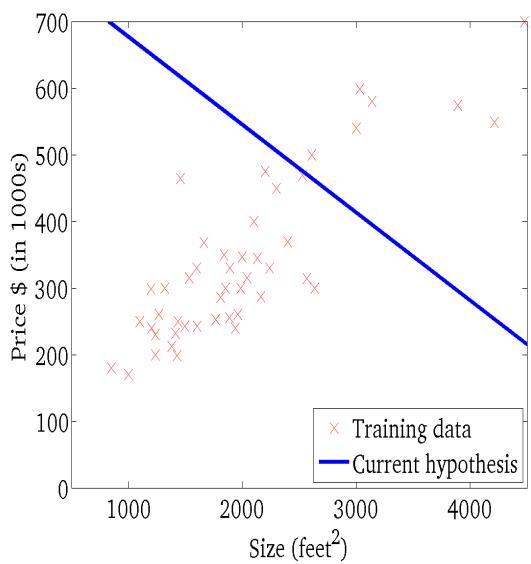
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



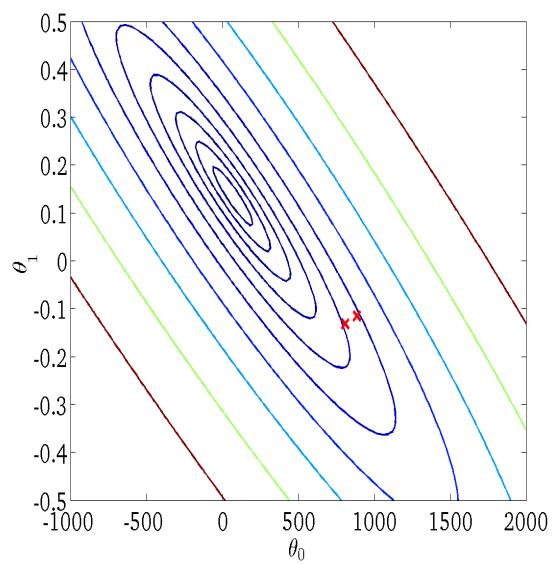
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



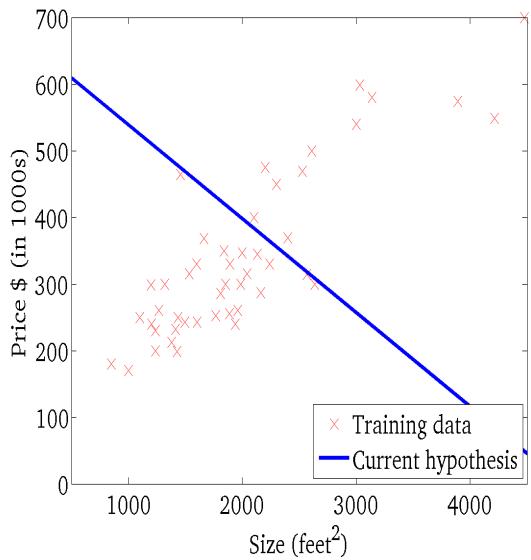
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



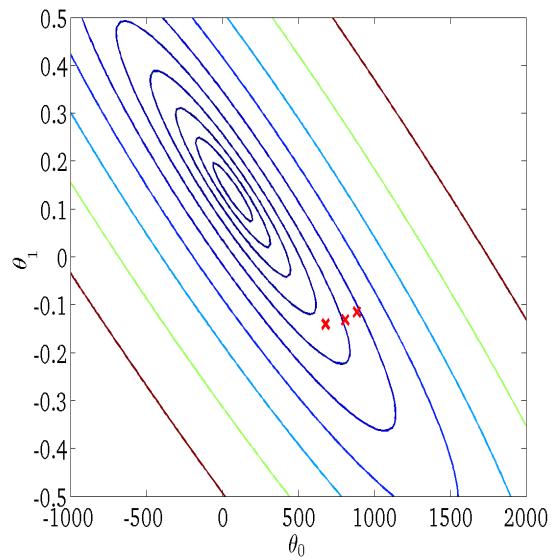
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



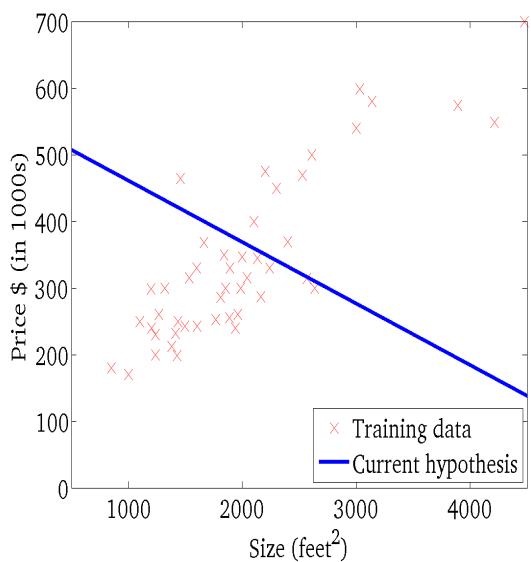
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



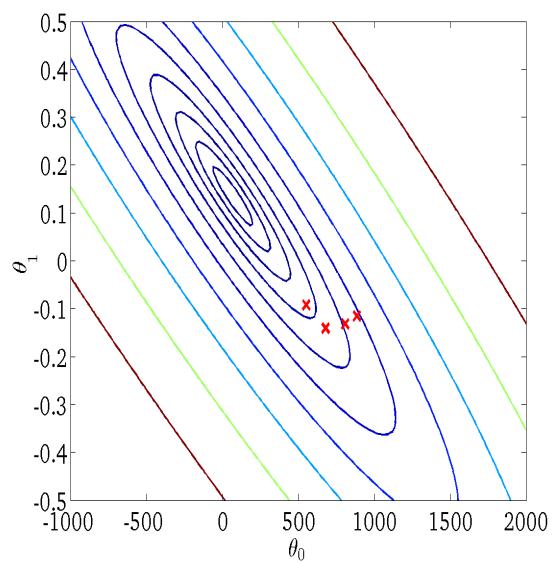
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



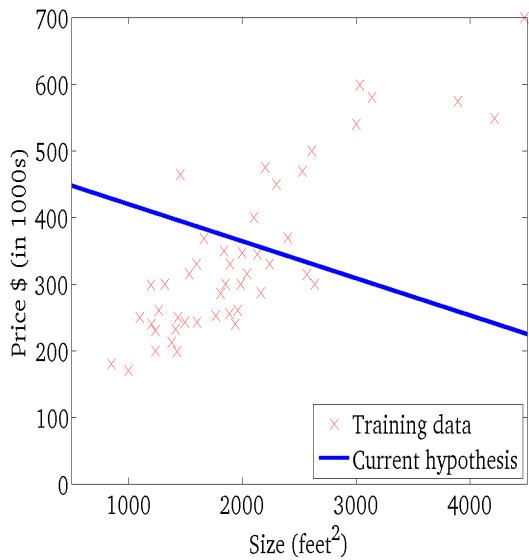
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



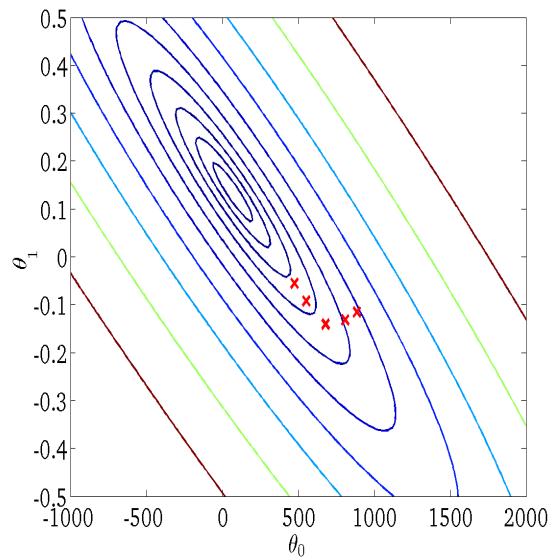
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



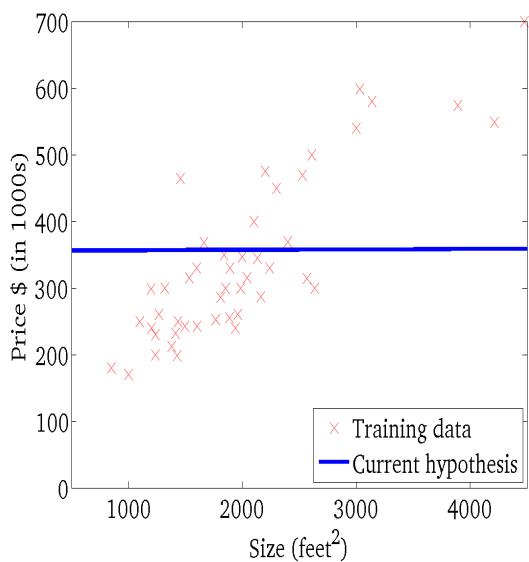
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



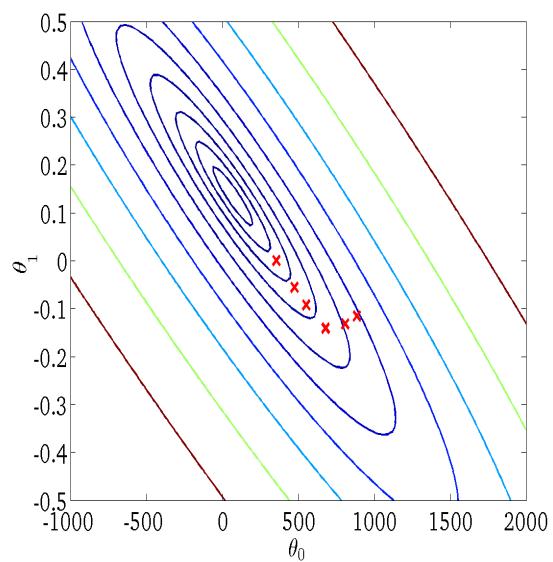
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



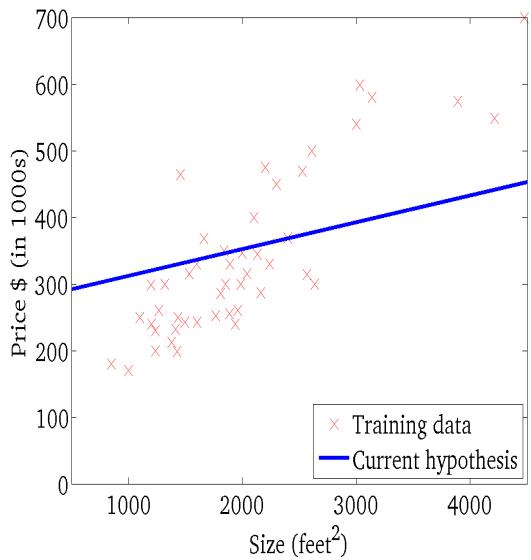
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



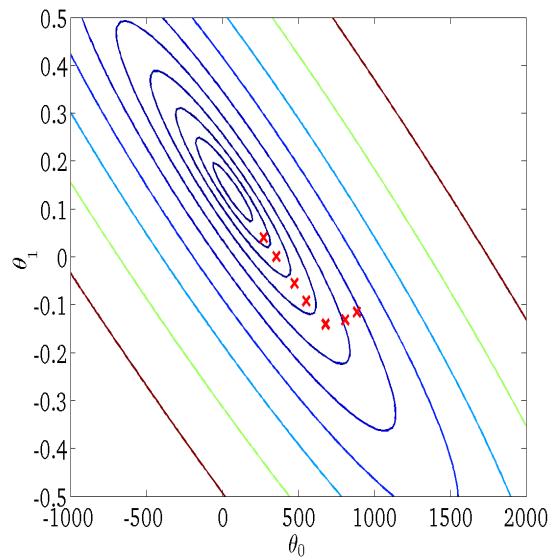
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



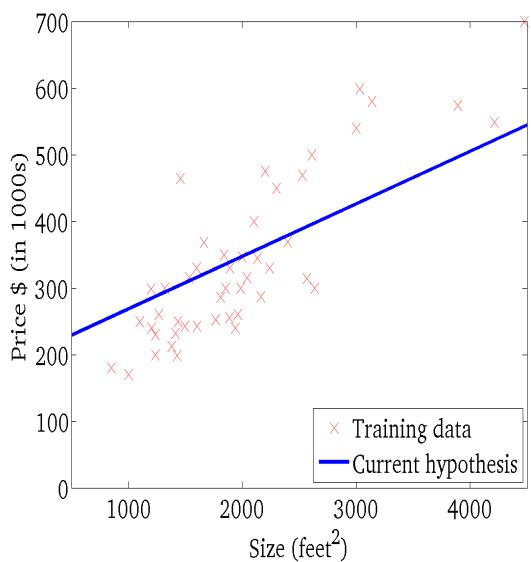
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



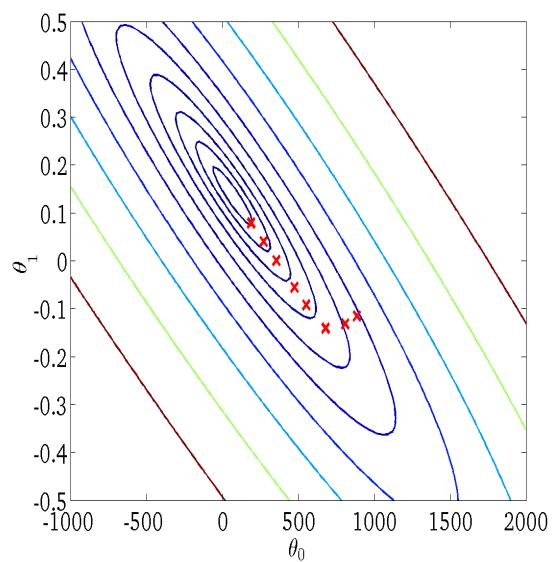
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

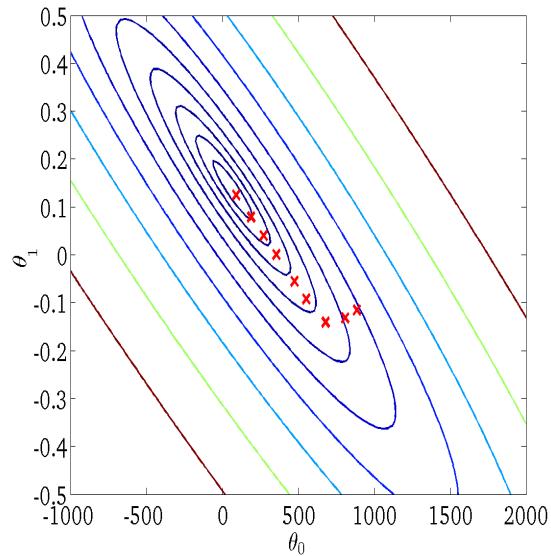
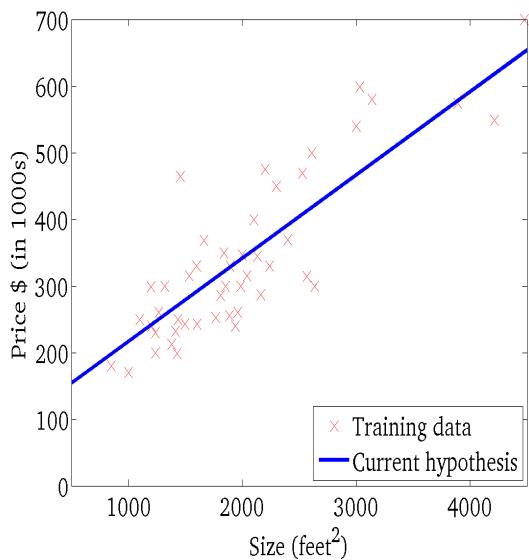


$$h_{\theta}(x)$$

$$J(\theta_0, \theta_1)$$

(for fixed θ_0, θ_1 , this is a function of x)

(function of the parameters θ_0, θ_1)



Linear Regression with multiple variables

Hypothesis: $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \dots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n)$$

} (simultaneously update for every $j = 0, \dots, n$)

Gradient Descent

Previously (n=1):

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m \underbrace{(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)} x_j^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

(simultaneously update θ_0, θ_1)

}

New algorithm ($n \geq 1$):

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for
 $j = 0, \dots, n$)

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

...

Examples: $m = 4$.

x_0	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

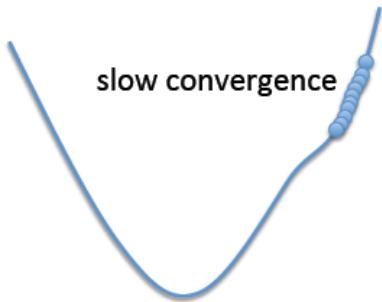
$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T y \quad \text{simultaneously update}$$

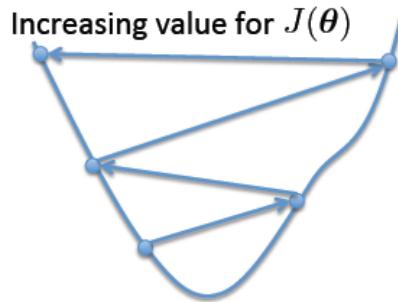
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Choosing α

α too small



α too large



- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

Closed Form Solution

- Can obtain θ by simply plugging X and y into

$$\theta = (X^T X)^{-1} X^T y$$

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

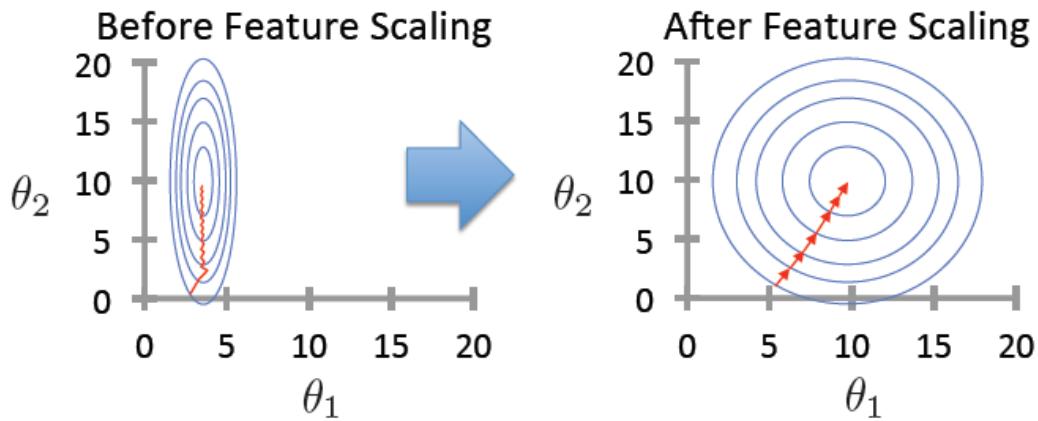
- If $X^T X$ is not invertible (i.e., singular), may need to:
 - Use pseudo-inverse instead of the inverse
 - In python, `numpy.linalg.pinv(a)`
 - Remove redundant (not linearly independent) features
 - Remove extra features to ensure that $d \leq n$

Gradient Descent vs Closed Form

Gradient Descent	Closed Form Solution
<ul style="list-style-type: none">Requires multiple iterationsNeed to choose αWorks well when n is largeCan support incremental learning	<ul style="list-style-type: none">Non-iterativeNo need for αSlow if n is large<ul style="list-style-type: none">– Computing $(X^T X)^{-1}$ is roughly $O(n^3)$

Improving Learning: Feature Scaling

- Idea:** Ensure that feature have similar scales

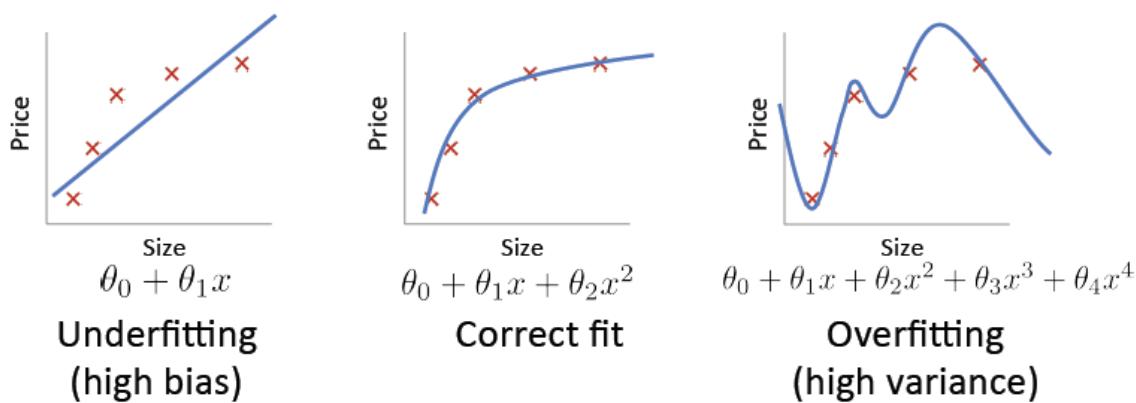


- Makes gradient descent converge *much* faster

Feature Standardization

- Rescales features to have zero mean and unit variance
 - Let μ_j be the mean of feature j : $\mu_j = \frac{1}{n} \sum_{i=1}^n x_j^{(i)}$
 - Replace each value with:
$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j} \quad \begin{matrix} \text{for } j = 1 \dots d \\ (\text{not } x_0!) \end{matrix}$$
 - s_j is the standard deviation of feature j
 - Could also use the range of feature j ($\max_j - \min_j$) for s_j
- Must apply the same transformation to instances for both training and prediction
- Outliers can cause problems

Quality of Fit



Overfitting:

- The learned hypothesis may fit the training set very well ($J(\theta) \approx 0$)
- ...but fails to generalize to new examples

Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea:** penalize for large values of θ_j
 - Can incorporate into the cost function
 - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

Regularization

- Linear regression objective function
- $$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d \theta_j^2$$
- model fit to data regularization
- λ is the regularization parameter ($\lambda \geq 0$)
 - No regularization on θ_0 !

Understanding Regularization

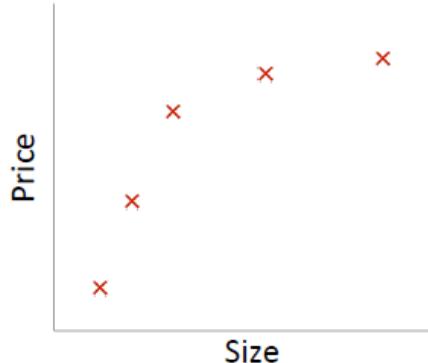
$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

- Note that $\sum_{j=1}^d \theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$
 - This is the magnitude of the feature coefficient vector!
- We can also think of this as:
$$\sum_{j=1}^d (\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{0}\|_2^2$$
- L₂ regularization pulls coefficients toward 0

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?

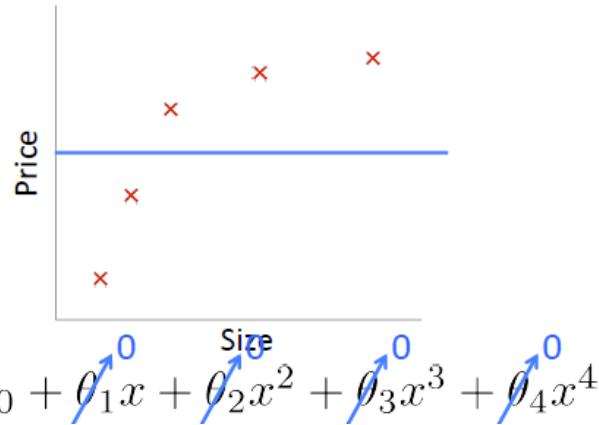


$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Understanding Regularization

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



Regularized Linear Regression

- Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

- Fit by solving $\min_{\theta} J(\theta)$

- Gradient update:

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\theta) & \quad \theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) \\ \frac{\partial}{\partial \theta_j} J(\theta) & \quad \theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \frac{\lambda}{n} \theta_j \end{aligned}$$

regularization

Regularized Linear Regression

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

$$\begin{aligned}\theta_0 &\leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) \\ \theta_j &\leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \frac{\lambda}{n} \theta_j\end{aligned}$$

- We can rewrite the gradient step as:

$$\theta_j \leftarrow \theta_j \left(1 - \alpha \frac{\lambda}{n} \right) - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Regularized Linear Regression

- To incorporate regularization into the closed form solution:

$$\theta = \left(\mathbf{X}^T \mathbf{X} + \lambda \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \right)^{-1} \mathbf{X}^T \mathbf{y}$$

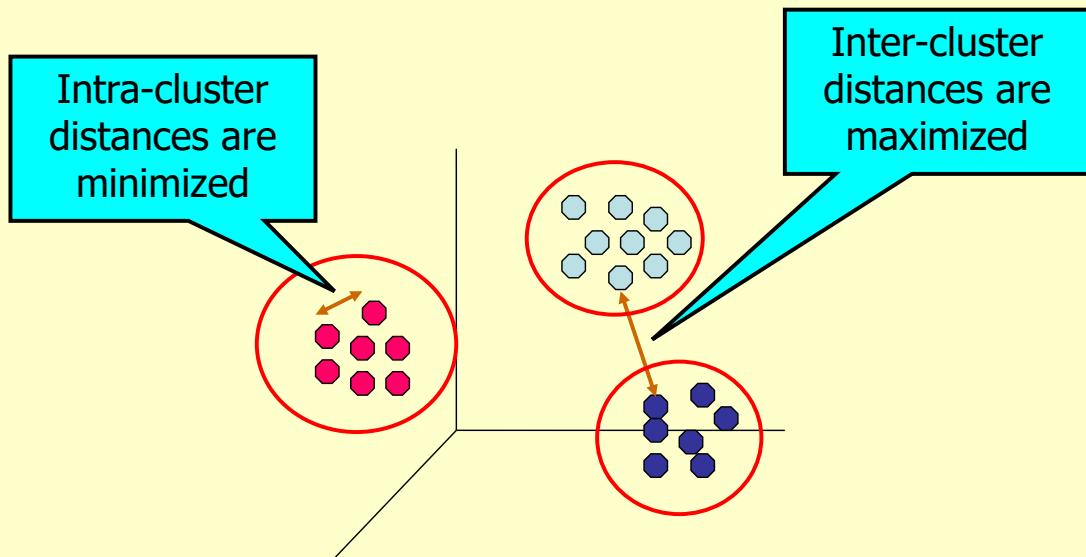
- Can derive this the same way, by solving $\frac{\partial}{\partial \theta} J(\theta) = 0$
- Can prove that for $\lambda > 0$, inverse exists in the equation above

Pattern Clustering: Introduction

Some slides from: Dr. Carla Brodley, Tufts University

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Why cluster?

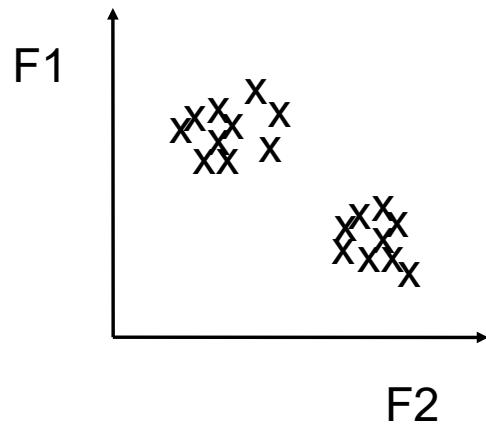
- Labeling is expensive
- Gain insight into the structure of the data
- Find prototypes in the data

Goal of Clustering

- Given a set of data points, each described by a set of attributes, find clusters such that:

- Inter-cluster similarity is maximized

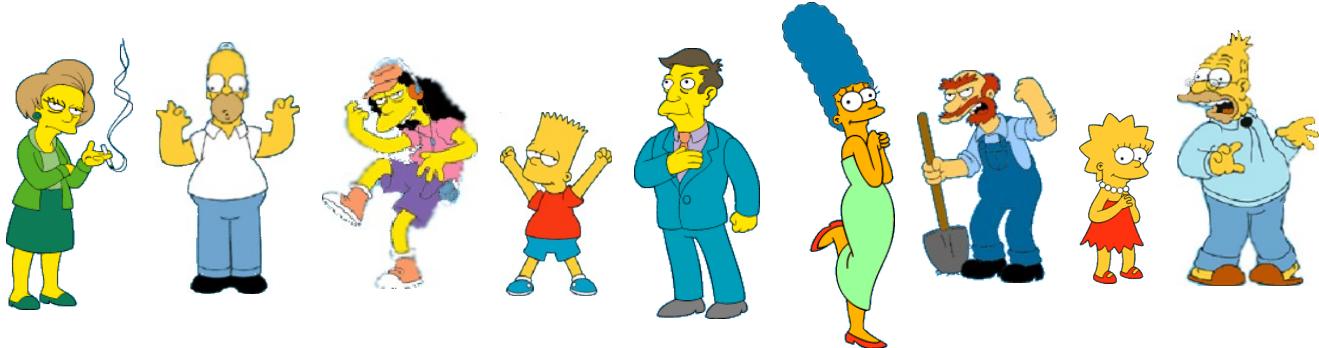
- Intra-cluster similarity is minimized



- Requires the definition of a similarity measure

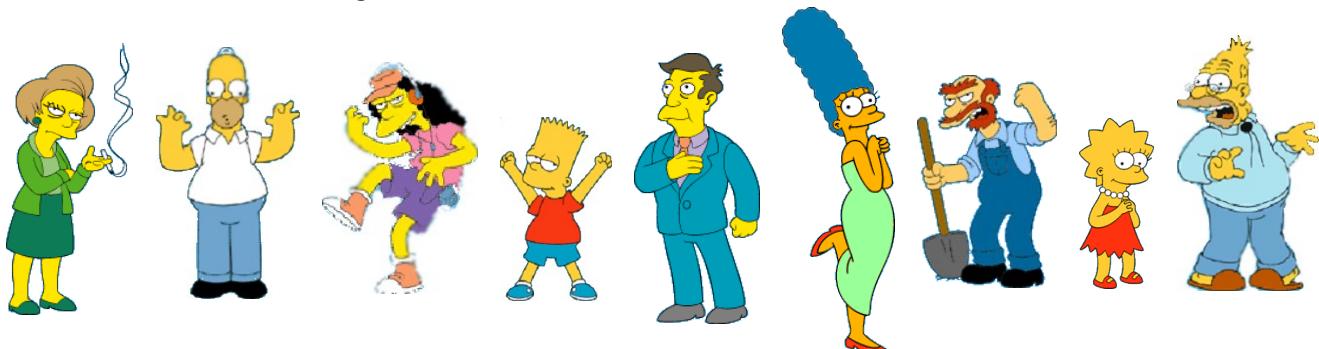
What is a natural grouping of these objects?

Slide from Eamonn Keogh

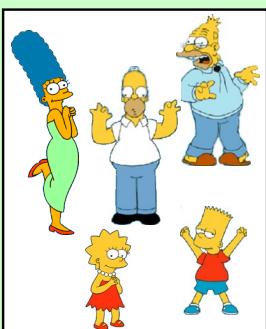


What is a natural grouping of these objects?

Slide from Eamonn Keogh



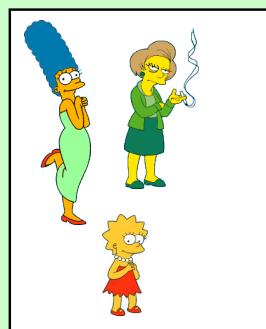
Clustering is subjective



Simpson's Family



School Employees



Females



Males

What is Similarity?

Slide based on one by Eamonn Keogh



Similarity is hard to define, but...
"We know it when we see it"

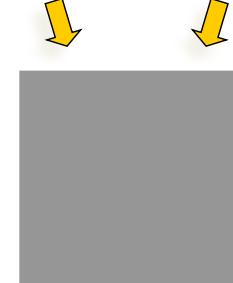
Defining Distance Measures

Slide from Eamonn Keogh

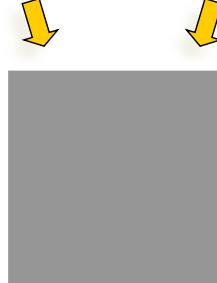
Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1, O_2)$



Peter Piotr



0.23



3



342.7

What properties should a distance measure have?

- $D(A,B) = D(B,A)$ *Symmetry*
- $D(A,A) = 0$ *Constancy of Self-Similarity*
- $D(A,B) = 0 \text{ iif } A = B$ *Positivity (Separation)*
- $D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*

Slide based on one by Eamonn Keogh

Intuitions behind desirable distance measure properties

$$D(A, B) = D(B, A)$$

Otherwise you could claim “Alex looks like Bob, but Bob looks nothing like Alex.”

$$D(A, A) = 0$$

Otherwise you could claim “Alex looks more like Bob, than Bob does.”

Intuitions behind desirable distance measure properties (continued)

$$D(A, B) = 0 \text{ IIf } A=B$$

Otherwise there are objects in your world that are different, but you cannot tell apart.

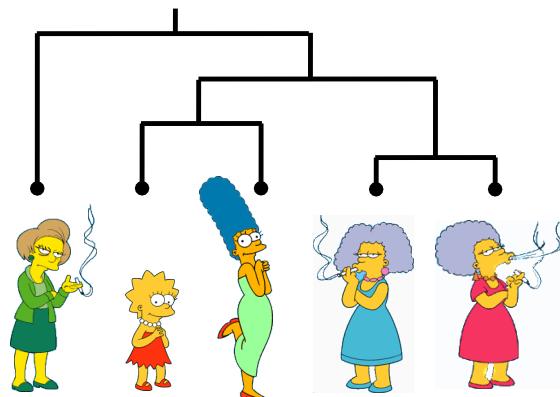
$$D(A, B) \leq D(A, C) + D(B, C)$$

Otherwise you could claim “Alex is very like Bob, and Alex is very like Carl, but Bob is very unlike Carl.”

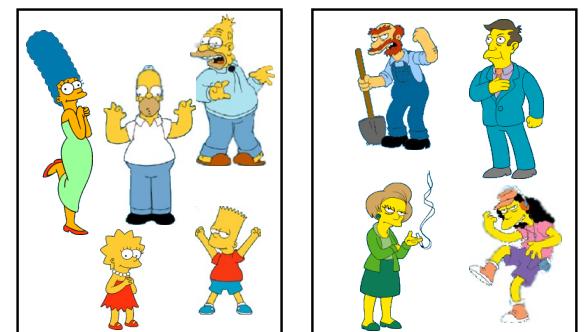
Two Types of Clustering

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion

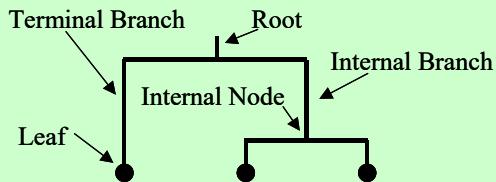
Hierarchical



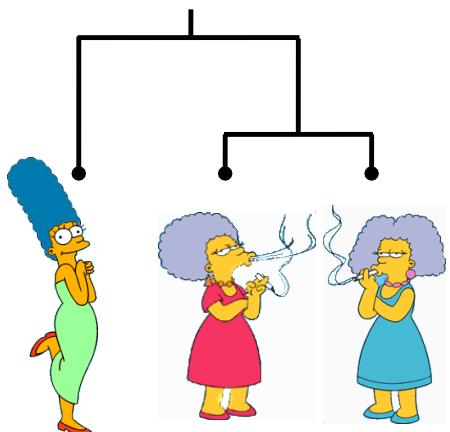
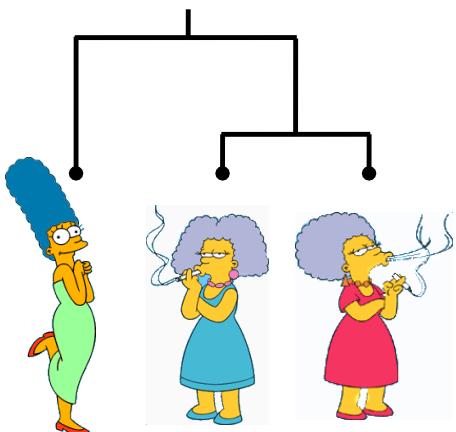
Partitional



Dendrogram: A Useful Tool for Summarizing Similarity Measurements

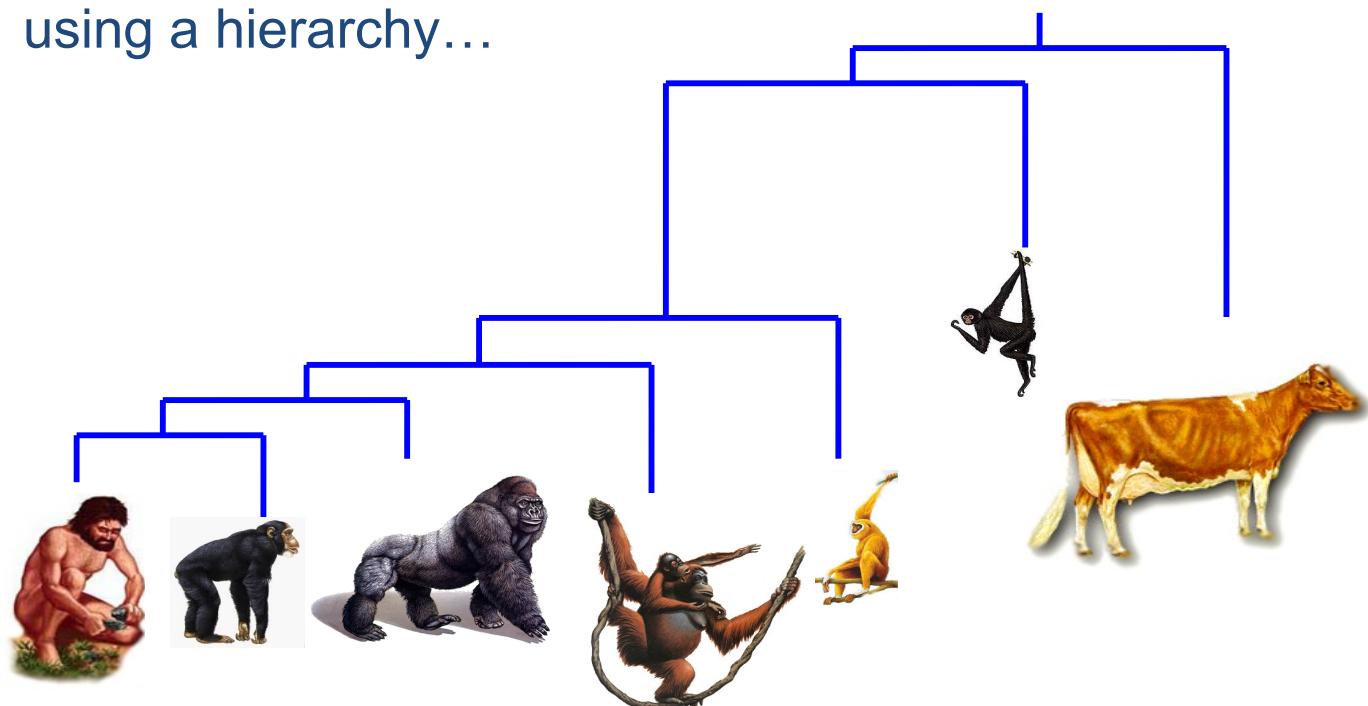


The similarity between two objects in a dendrogram is represented as the height of the lowest internal node they share.



Slide based on one by Eamonn Keogh

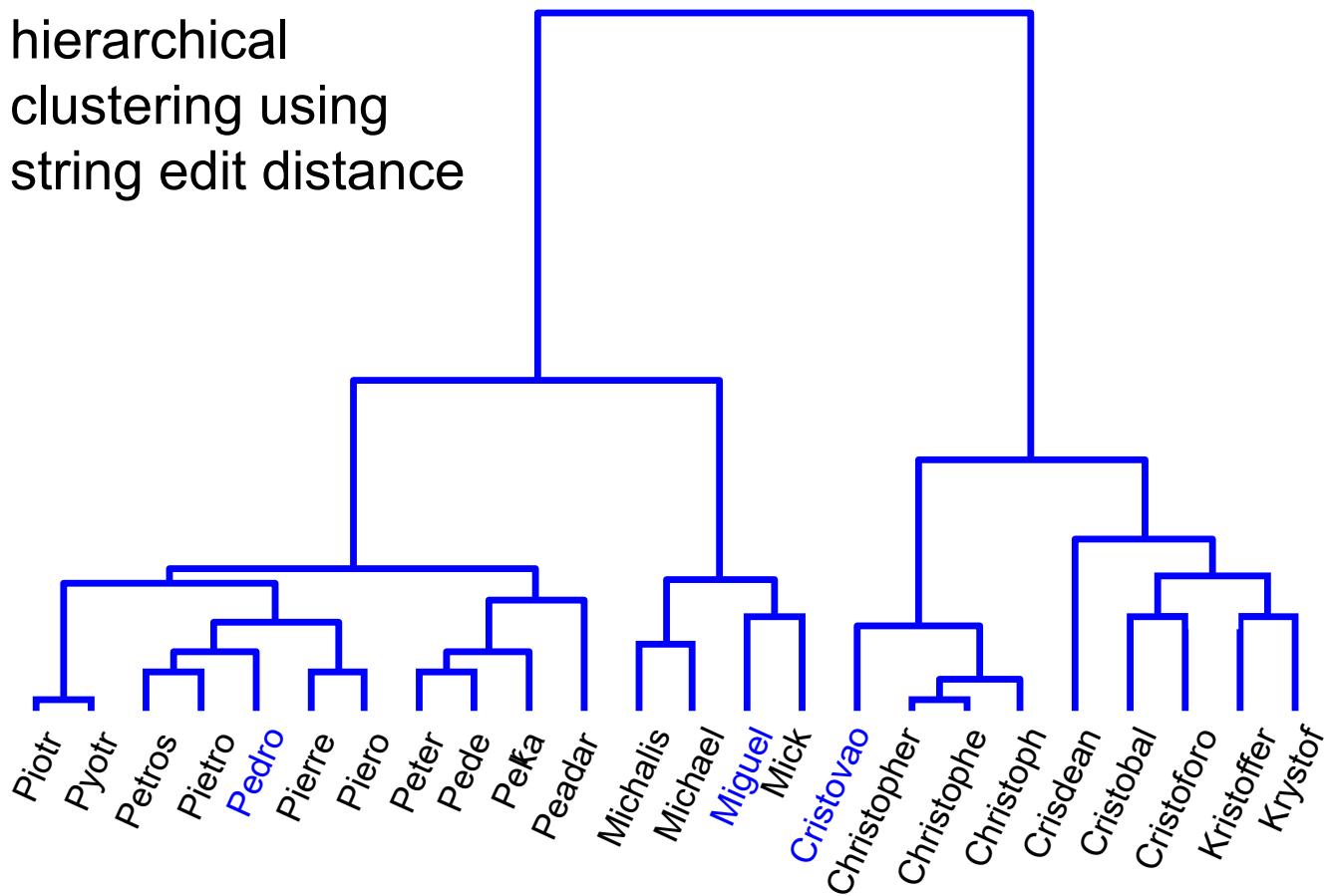
There is only one dataset that can be perfectly clustered using a hierarchy...



(Bovine:0.69395, (Spider Monkey 0.390, (Gibbon:0.36079,(Orang:0.33636,(Gorilla:0.17147,(Chimp:0.19268,
Human:0.11927):0.08386):0.06124):0.15057):0.54939);

A demonstration of hierarchical clustering using string edit distance

Slide based on one by Eamonn Keogh

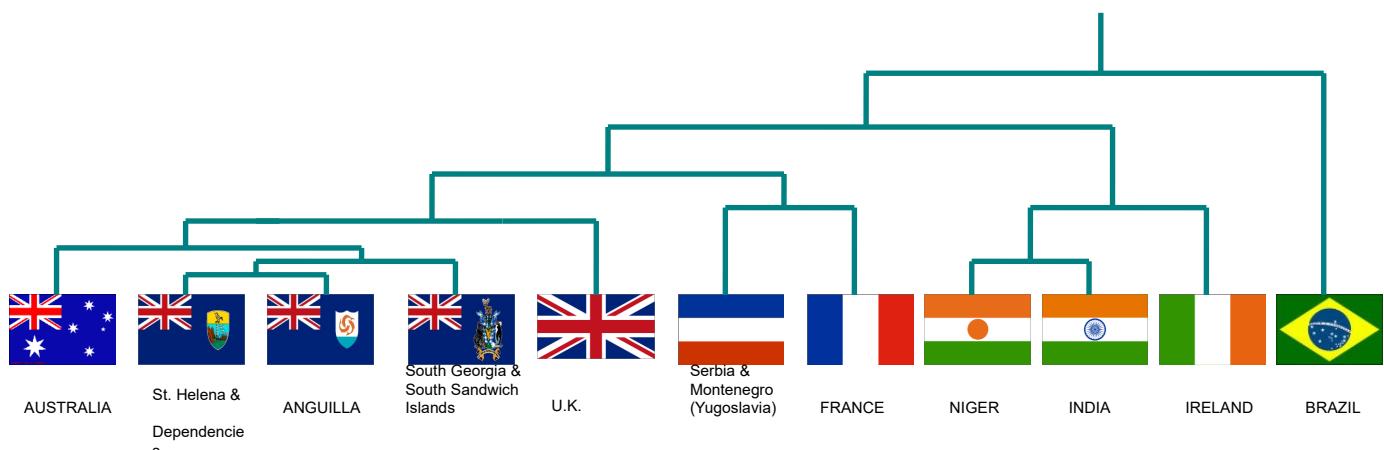


Slide based on one by Eamonn Keogh

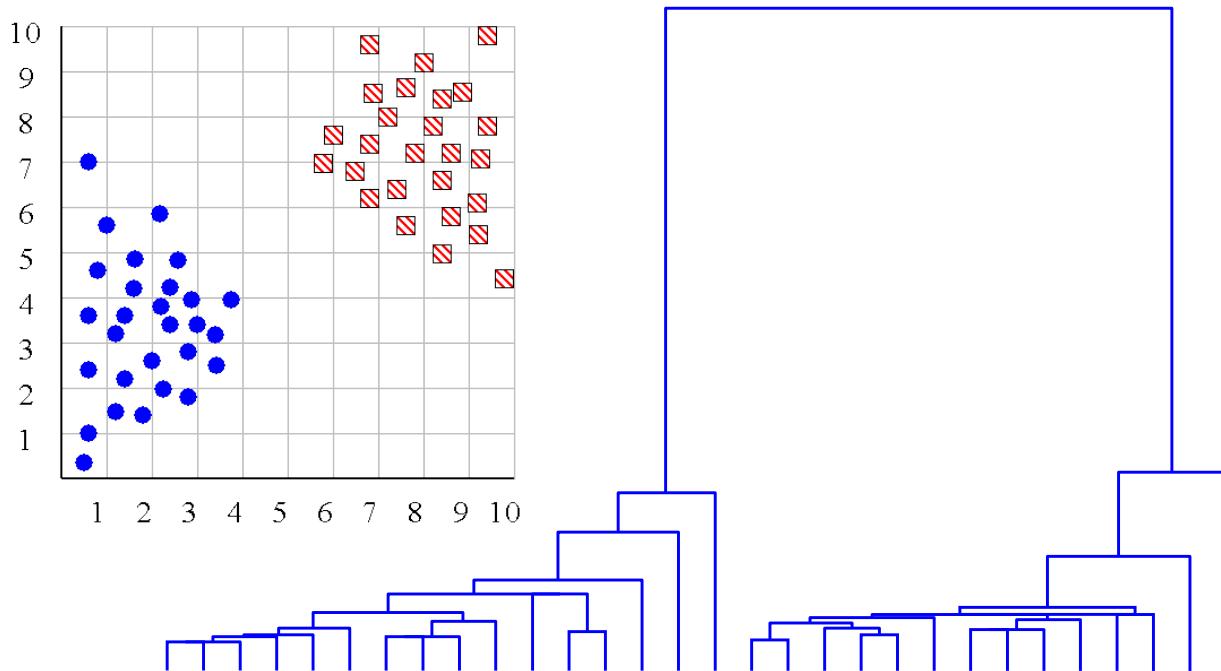
Hierachal clustering can sometimes show patterns that are meaningless or spurious

The tight grouping of Australia, Anguilla, St. Helena etc is meaningful; all these countries are former UK colonies

However the tight grouping of Niger and India is completely spurious; there is no connection between the two.

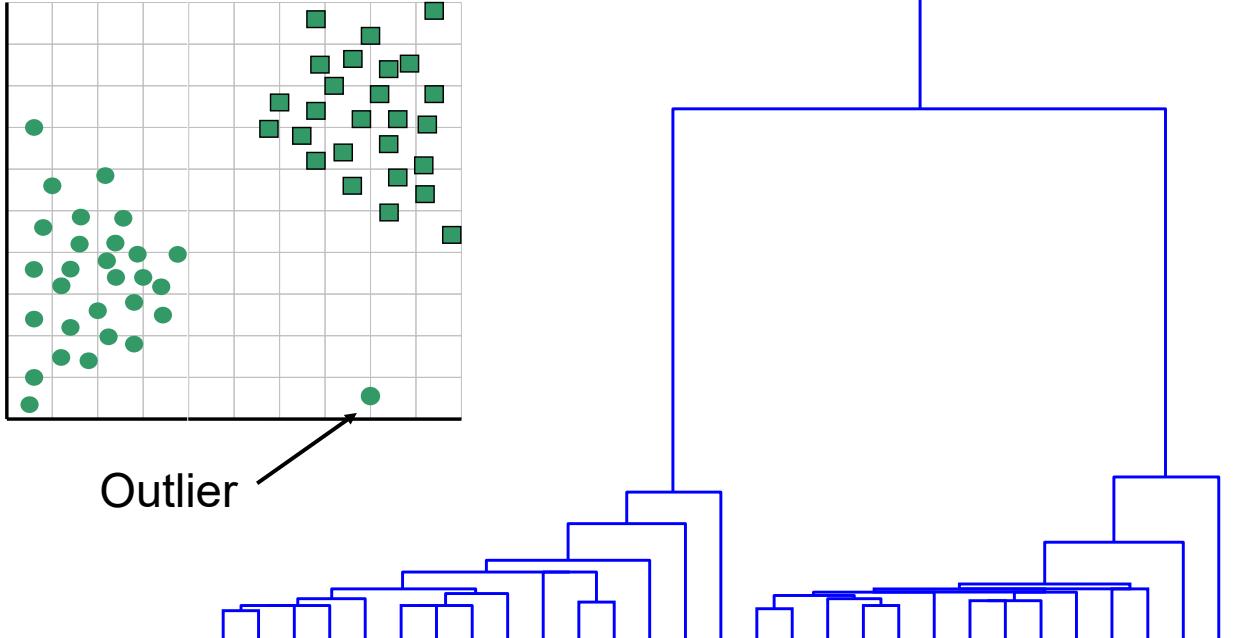


We can look at the dendrogram to determine the “correct” number of clusters.



One potential use of a dendrogram: detecting outliers

The single isolated branch is suggestive of a data point that is very different to all others

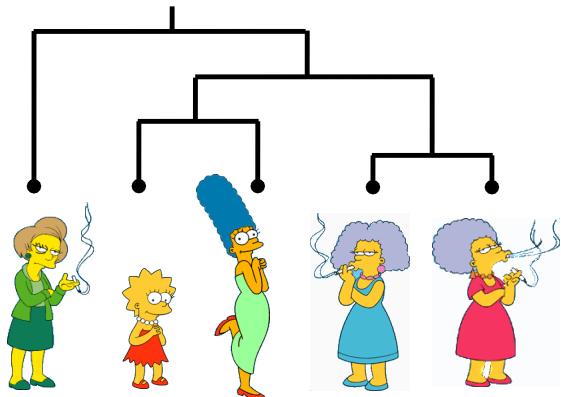


Hierarchical Clustering

Slide based on one by Eamonn Keogh

The number of dendograms with n leafs = $(2n - 3)!/[2^{(n-2)}(n-2)!]$

Number of Leafs	Number of Possible Dendograms
2	1
3	3
4	15
5	105
...	...
10	34,459,425



Since we cannot test all possible trees we will have to heuristic search of all possible trees. We could do this..

Bottom-Up (agglomerative):

Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

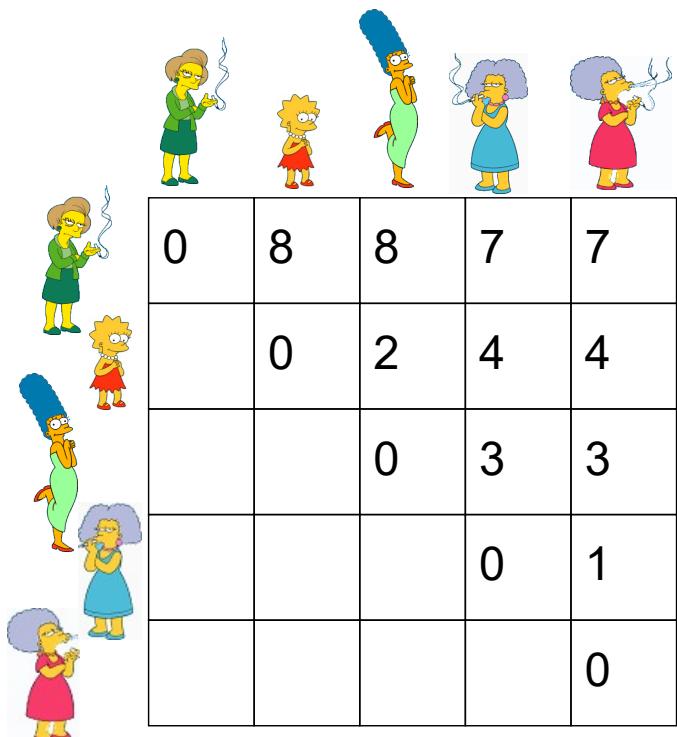
Top-Down (divisive): Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

Slide based on one by Eamonn Keogh

We begin with a distance matrix which contains the distances between every pair of objects in our database.

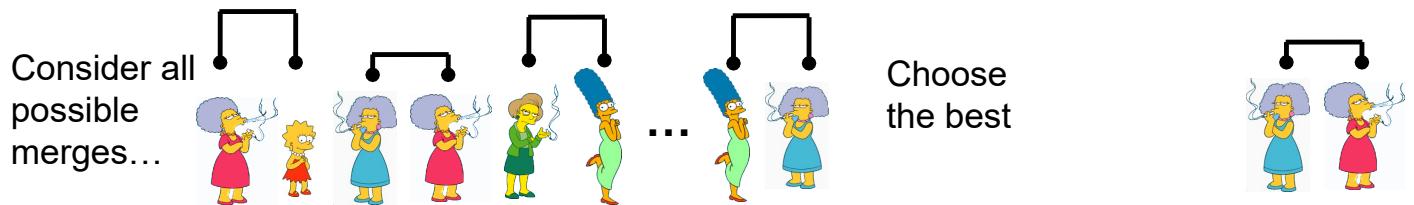
$$D(\text{Marge, Lisa}) = 8$$

$$D(\text{Marge, Marge}) = 1$$

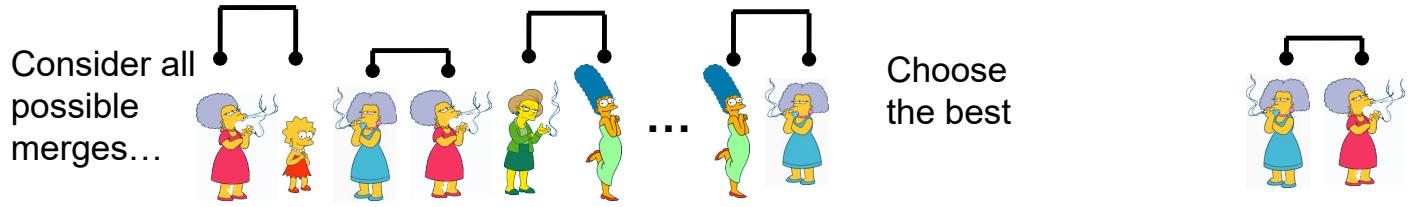
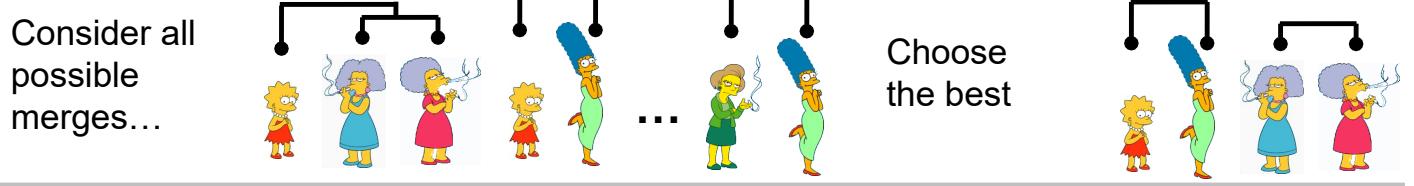


Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

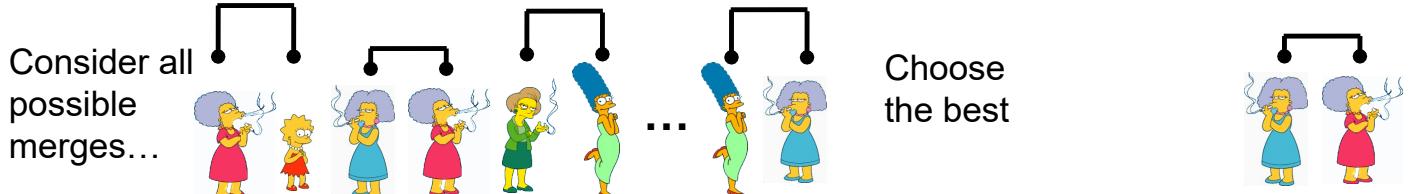
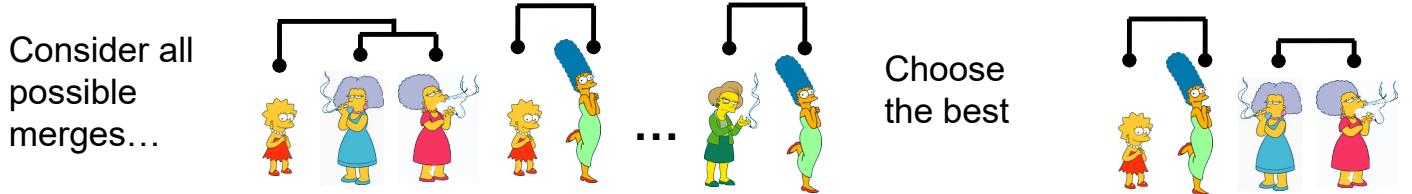
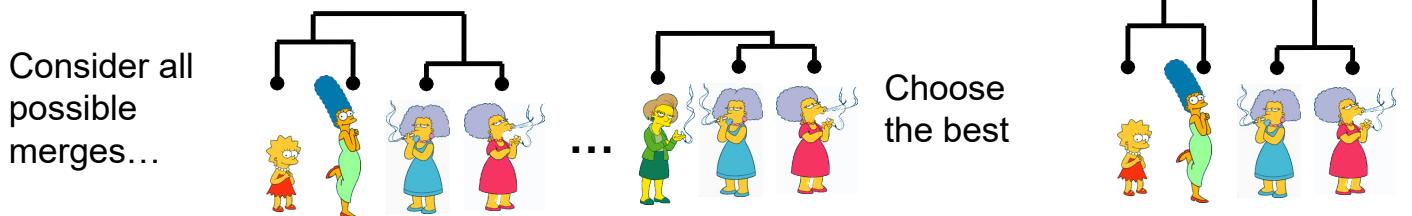
This slide and next 4 based on slides by Eamonn Keogh



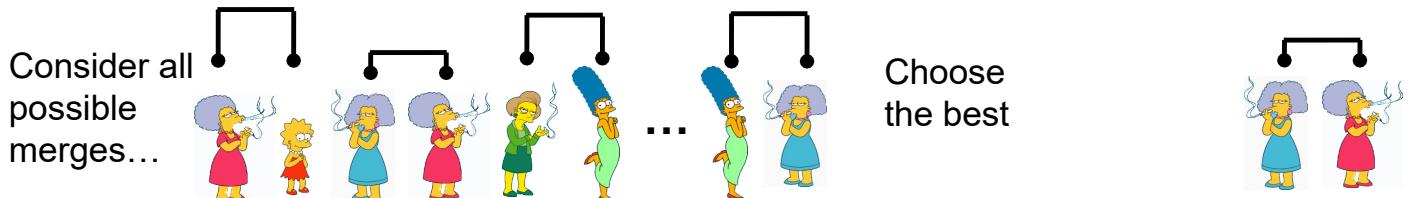
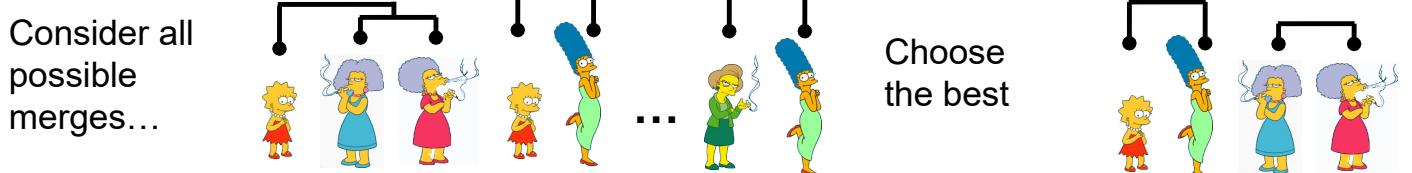
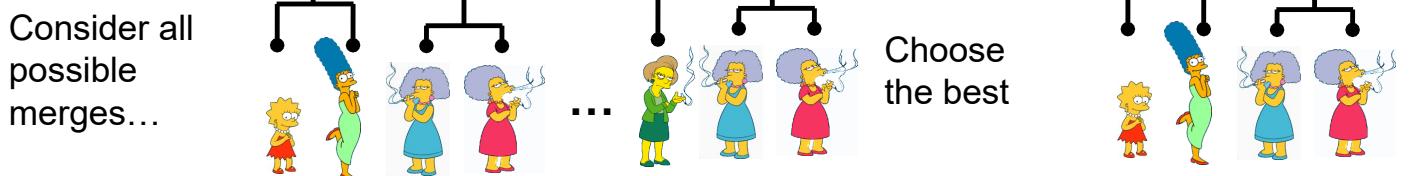
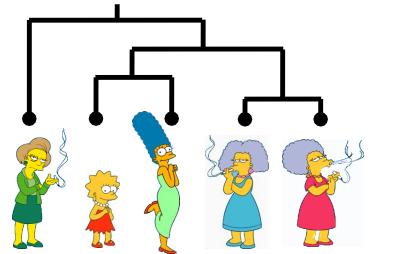
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

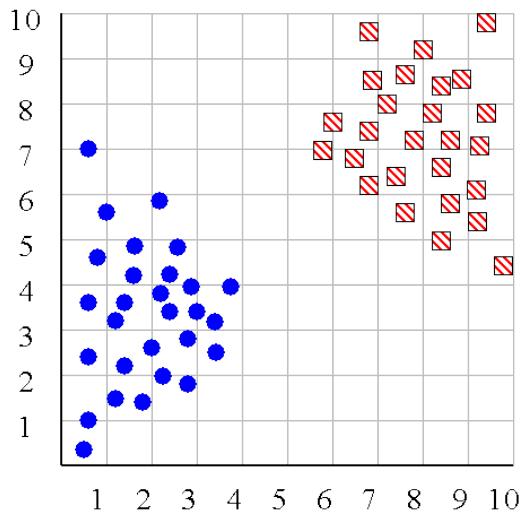


Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

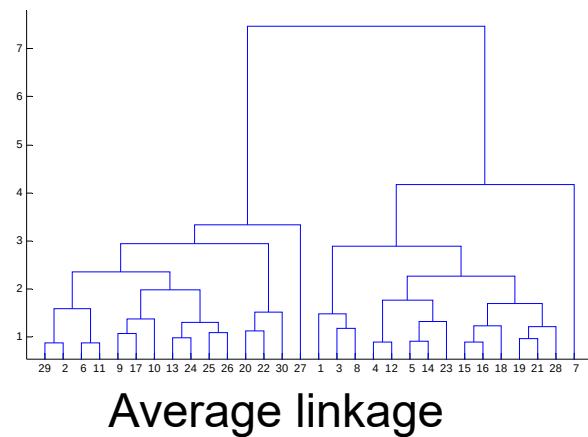
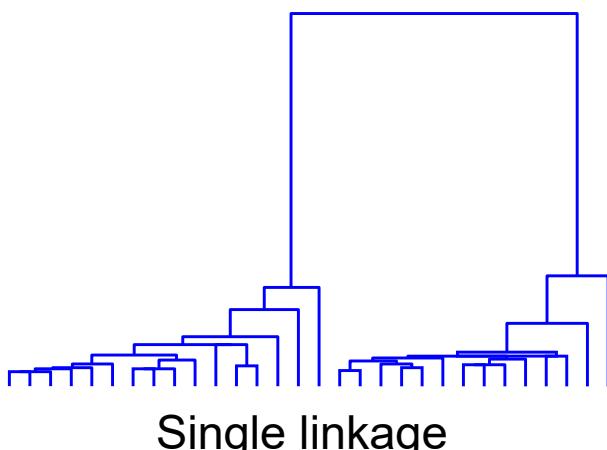


We know how to measure the distance between two objects, but defining the distance between an object and a cluster, or defining the distance between two clusters is non obvious.

- **Single linkage (nearest neighbor):** In this method the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.
- **Complete linkage (furthest neighbor):** In this method, the distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors").
- **Group average linkage:** In this method, the distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters.



Slide based on one by Eamonn Keogh



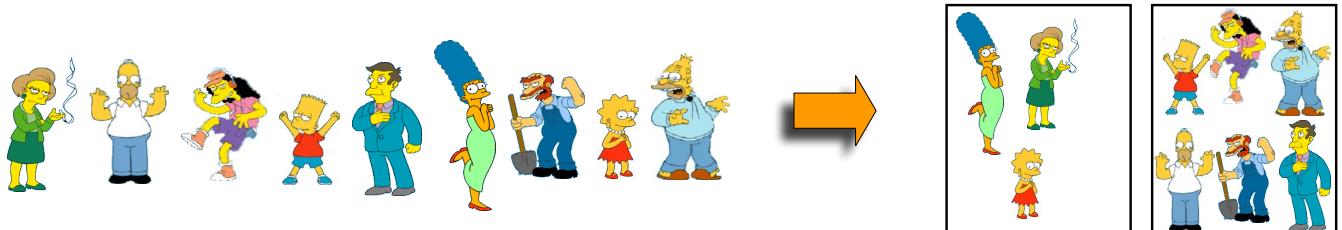
Hierarchal Clustering Methods Summary

- No need to specify the number of clusters in advance
- Hierarchal nature maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
- Like any heuristic search algorithms, local optima are a problem
- Interpretation of results is (very) subjective

Slide based on one by Eamonn Keogh

Partitional Clustering

- Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters K.



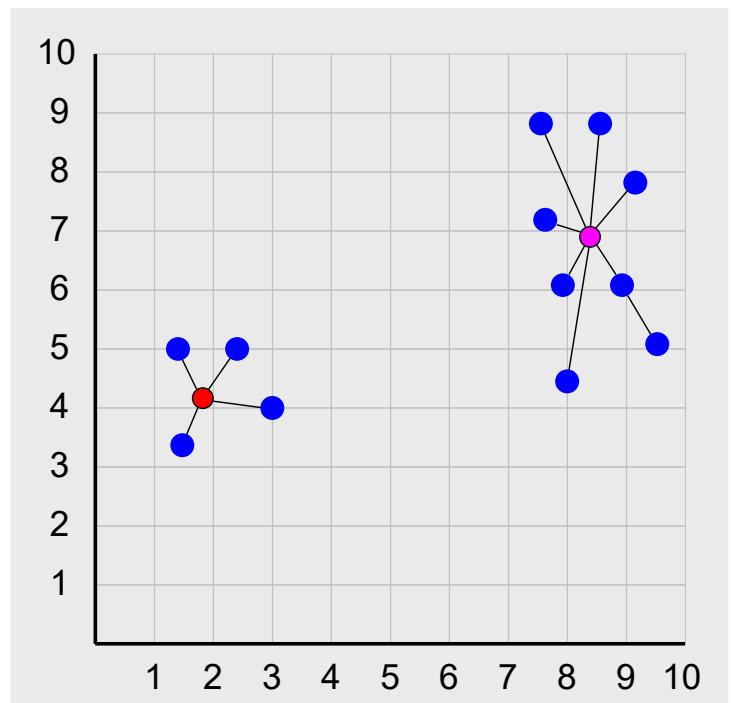
Squared Error

$$se_{K_i} = \sum_{j=1}^m \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^k se_{K_j}$$



Objective Function



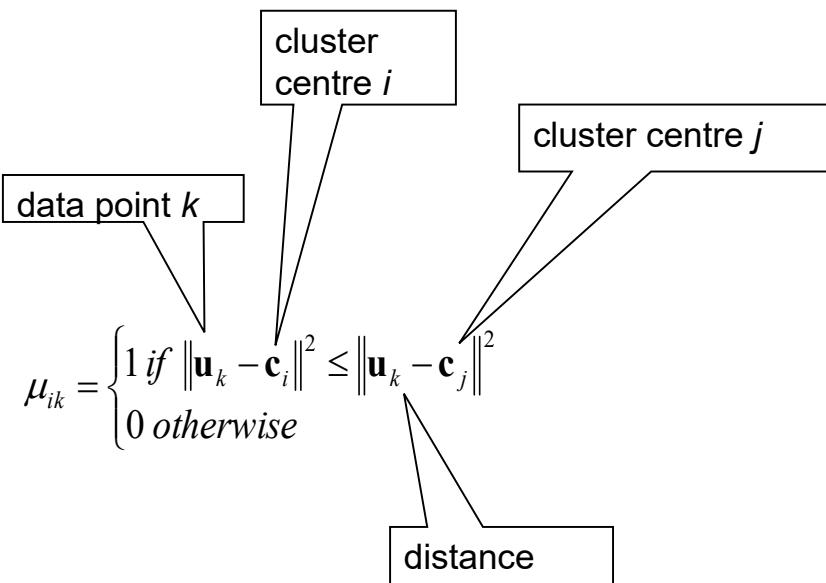
Types of Clusters: Objective Function

- **Clusters Defined by an Objective Function**
 - Finds clusters that minimize or maximize an objective function.
 - Enumerate all possible ways of dividing the points into clusters and evaluate the 'goodness' of each potential set of clusters by using the given objective function. (No. of feasible partitions could be very large and the problem is NP Hard)
 - Can have global or local objectives.
 - Hierarchical clustering algorithms typically have local objectives
 - Partitional algorithms typically have global objectives
 - A variation of the global objective function approach is to fit the data to a parameterized model.
 - Parameters for the model are determined from the data.
 - Mixture models assume that the data is a 'mixture' of a number of statistical distributions.

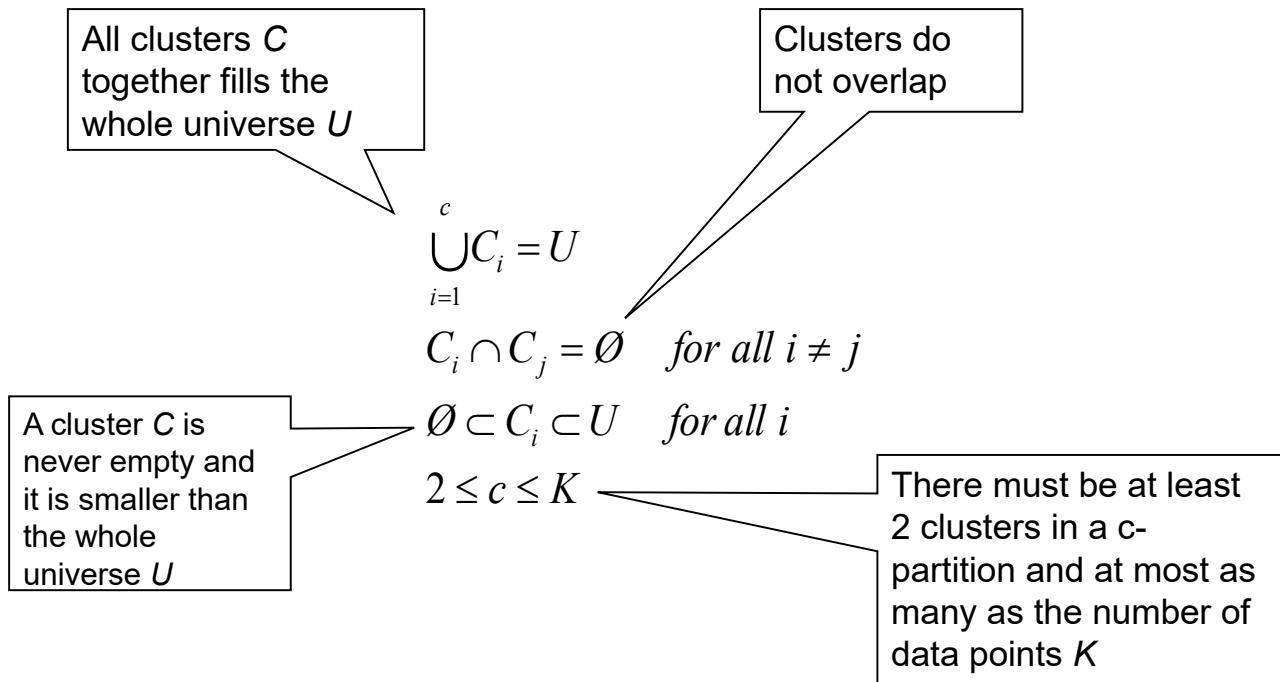
Types of Clusters: Objective Function ...

- Map the clustering problem to a different domain and solve a related problem in that domain
 - Proximity matrix defines a weighted graph, where the nodes are the points being clustered, and the weighted edges represent the proximities between points
 - Clustering is equivalent to breaking the graph into connected components, one for each cluster.
 - Want to minimize the edge weight between clusters and maximize the edge weight within clusters

Membership matrix \mathbf{M}



c-partition



Objective Functions: Hard and Fuzzy C Means

The diagram shows the objective function for k -means clustering:

$$J_{k-means} = \sum_{i=1}^c J_i = \sum_{i=1}^c \left(\sum_{k, \mathbf{u}_k \in C_i} \|\mathbf{u}_k - \mathbf{c}_i\|^2 \right)$$

Minimise the total sum of all distances

The diagram shows the objective function for Fuzzy C Means (FCM) clustering:

$$J_{FCM} = \sum_{i=1}^c J_i = \sum_{i=1}^c \sum_{k=1}^n \mu_{ik}^m \|\mathbf{u}_k - \mathbf{c}_i\|^2$$

K-Means Type Clustering Algorithms

- Given k , the *k-means* algorithm consists of four steps:
 - Select initial representative points (means or centroids) at random.
 - Assign each object to the cluster with the nearest centroid.
 - Compute each centroid as the mean of the objects assigned to it.
 - Repeat previous 2 steps until no change.

35

The Objective Function of K-Means

- Most common measure is Sum of Squared Error (SSE) or summed Intra Cluster Spread (ICS)
 - For each point, the error is the distance to the nearest cluster center
 - To get SSE, we square these errors and sum them.

$$SSE = ICS(\vec{c}_1, \vec{c}_2, \dots, \vec{c}_k) = \sum_{j=1}^k \sum_{\vec{X}_i \in C_j} d^2(\vec{X}_i, \vec{c}_j)$$

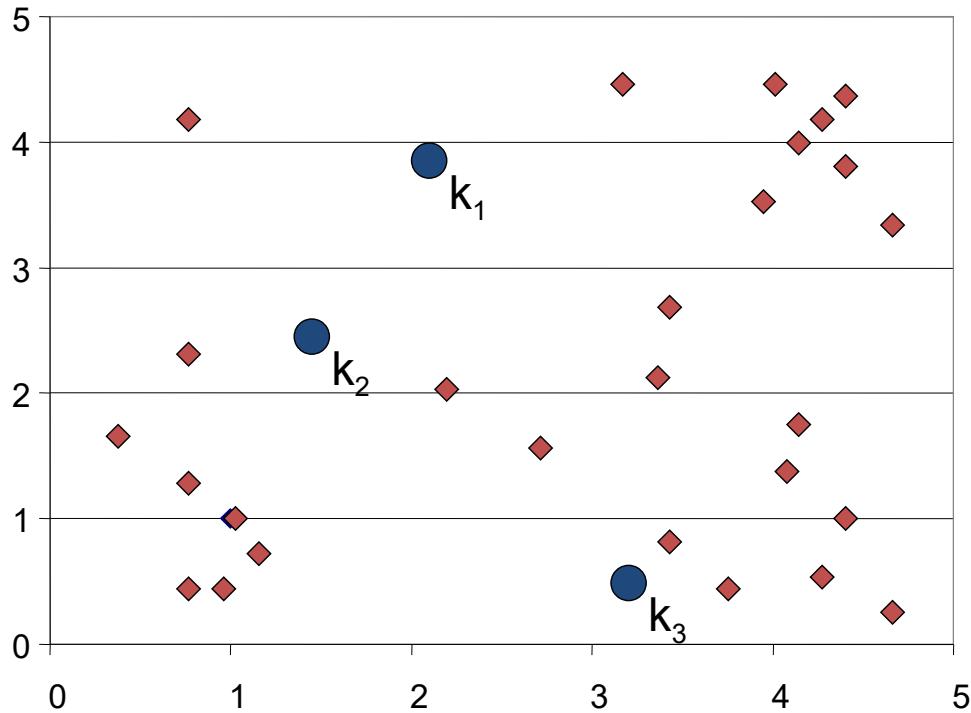
- \vec{X}_i is a data point in cluster C_j and \vec{c}_j is the representative point (most often the mean) for cluster C_j .
- Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase k , the number of clusters
 - A good clustering with smaller k can have a higher SSE than a poor clustering with higher k

Partition Algorithm 1: k-means

1. Decide on a value for k .
2. Initialize the k cluster centers (randomly, if necessary).
3. Decide the class memberships of the N objects by assigning them to the nearest cluster center.
4. Re-estimate the k cluster centers, by assuming the memberships found above are correct.
5. If none of the N objects changed membership in the last iteration, exit. Otherwise goto 3.

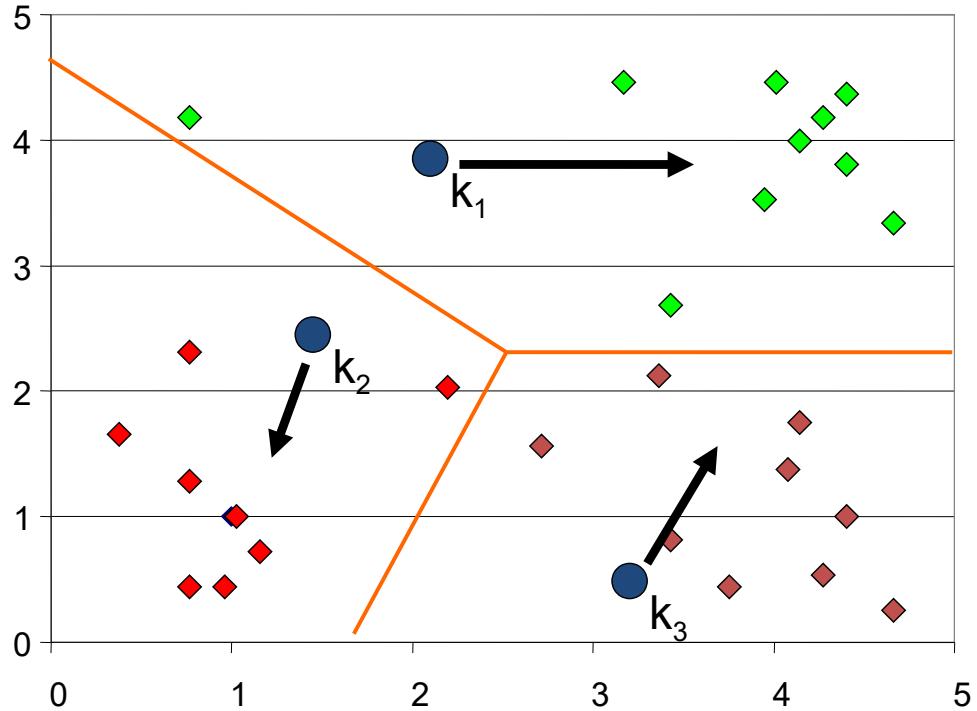
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 2

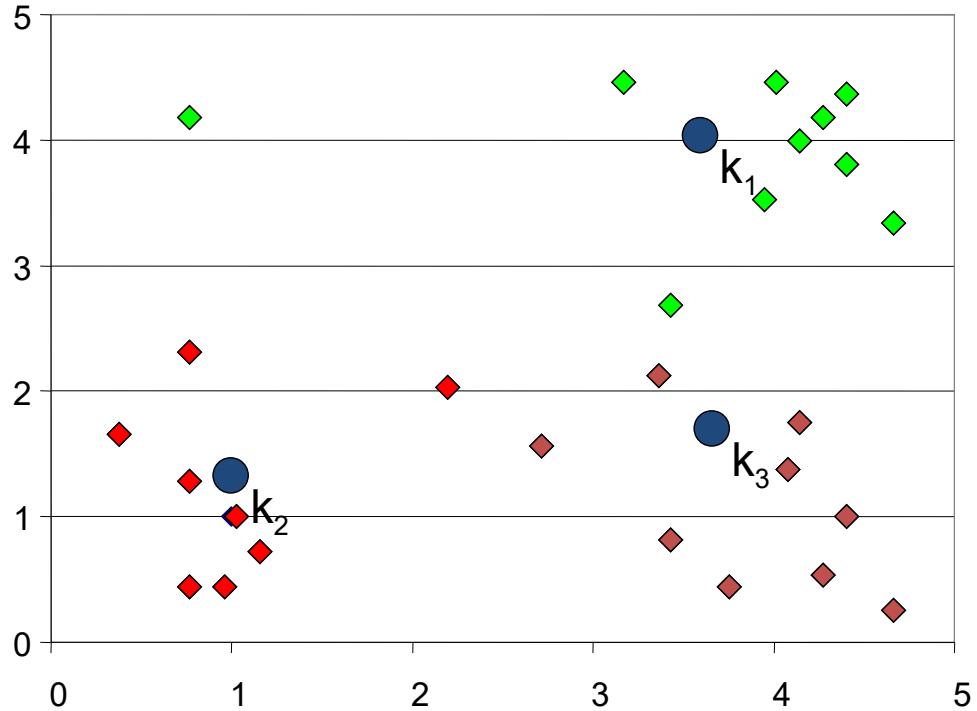
Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

K-means Clustering: Step 3

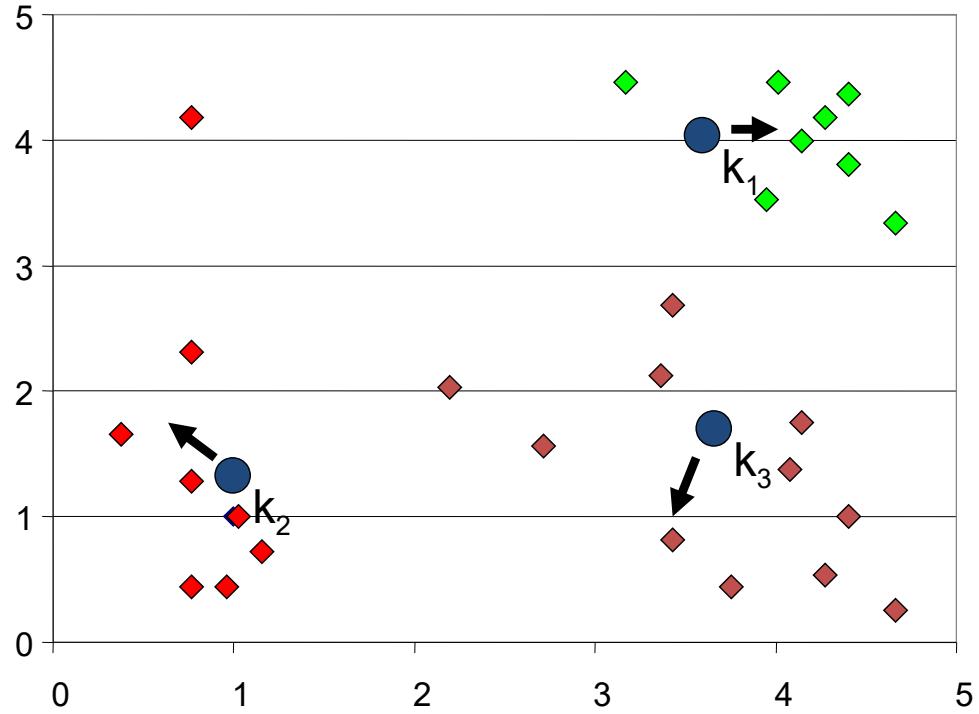
Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

K-means Clustering: Step 4

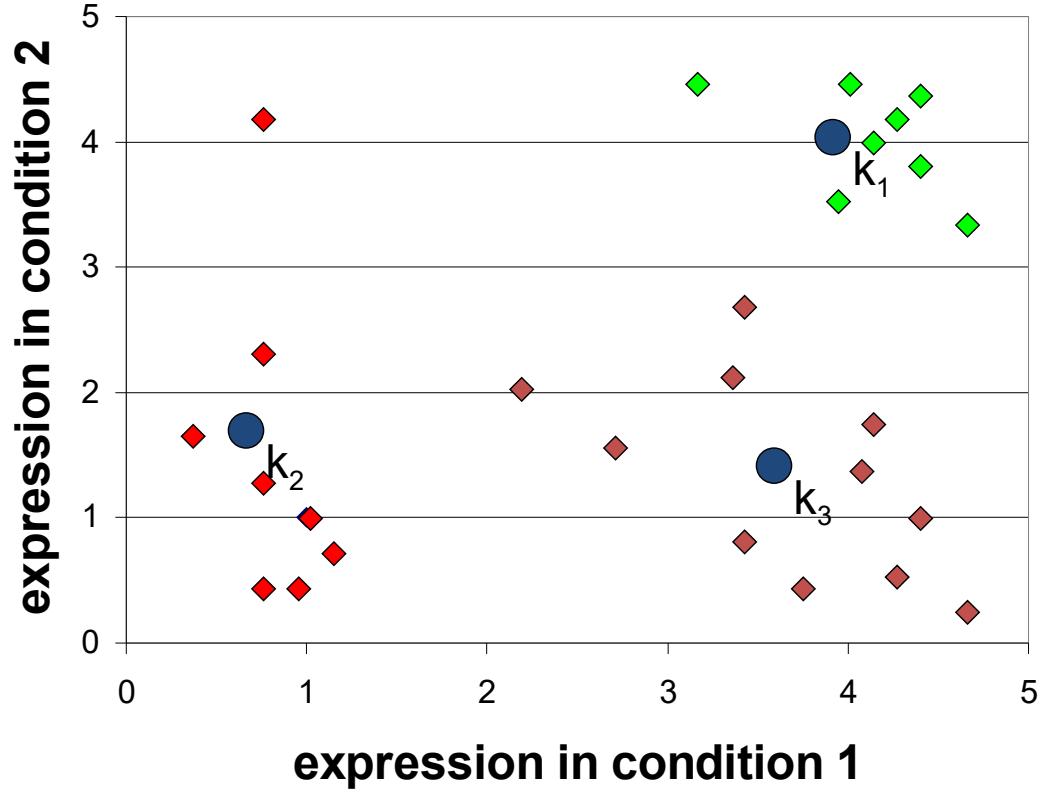
Algorithm: k-means, Distance Metric: Euclidean Distance



Slide based on one by Eamonn Keogh

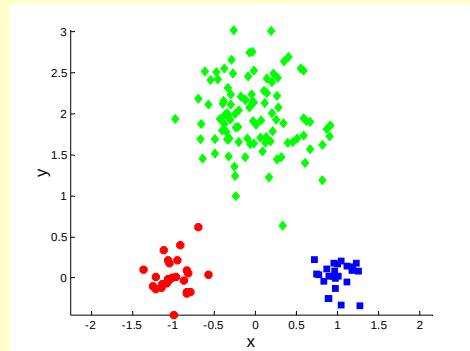
K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance

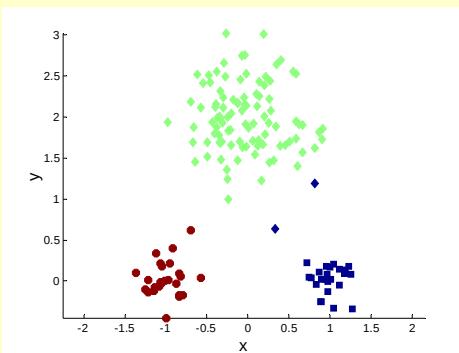


Slide based on one by Eamonn Keogh

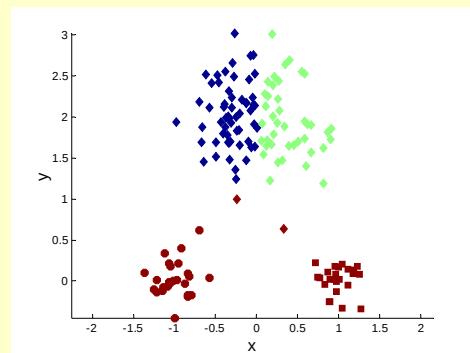
Two different K-means Clustering



Original Points

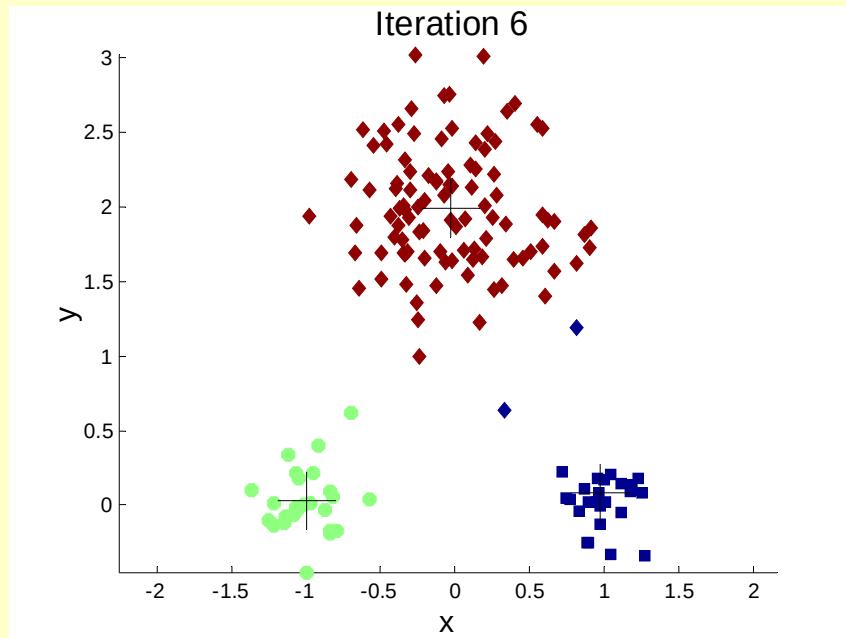


Optimal Clustering

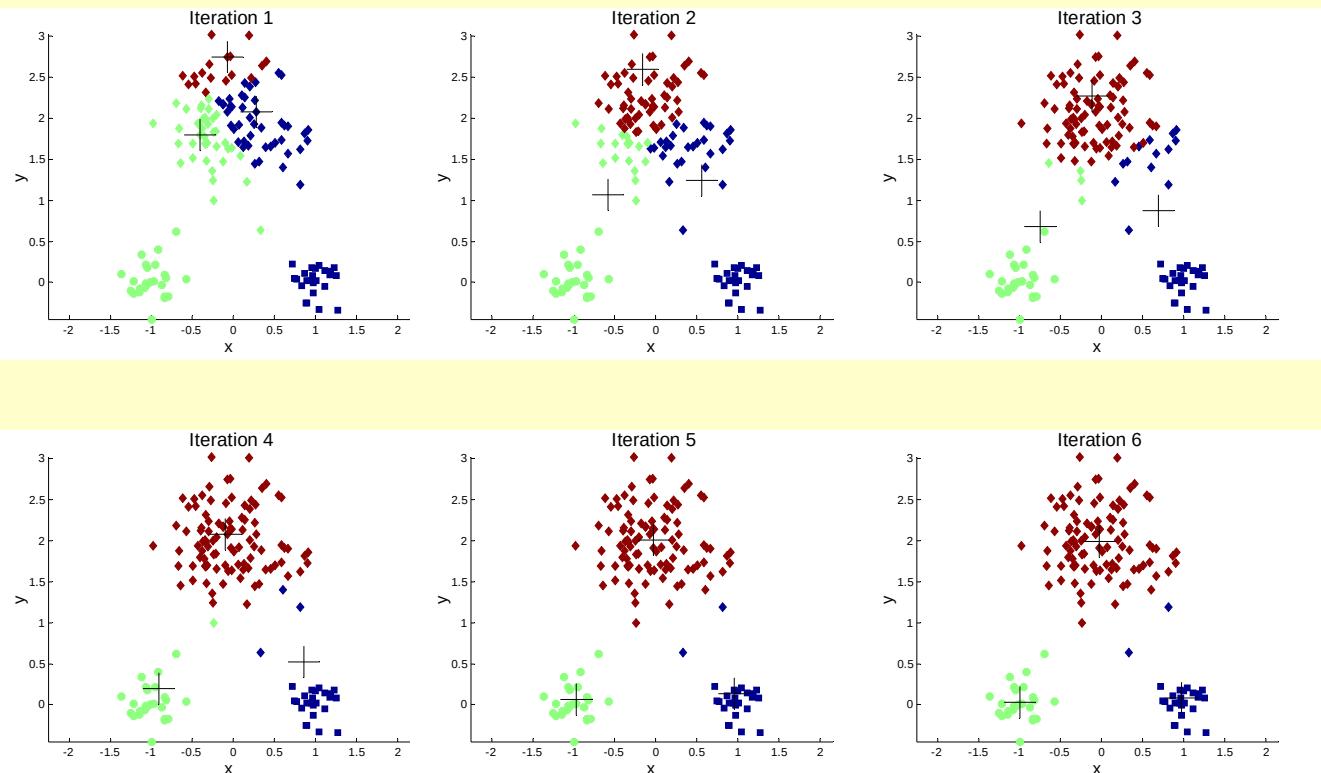


Sub-optimal Clustering

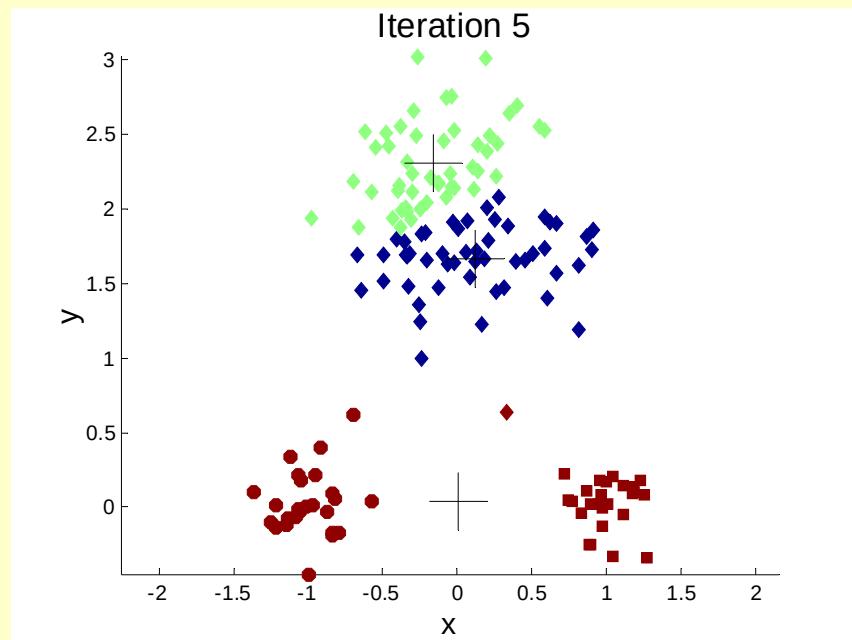
Importance of Choosing Initial Centroids



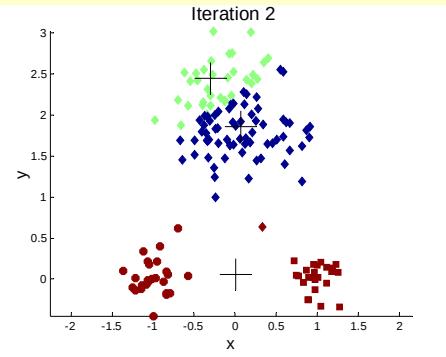
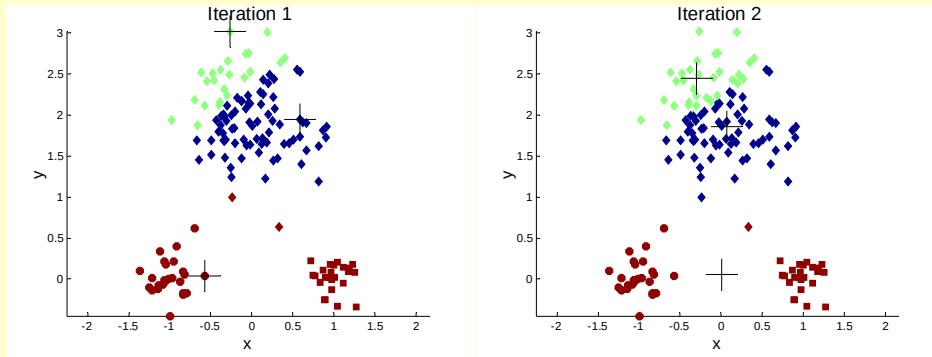
Importance of Choosing Initial Centroids



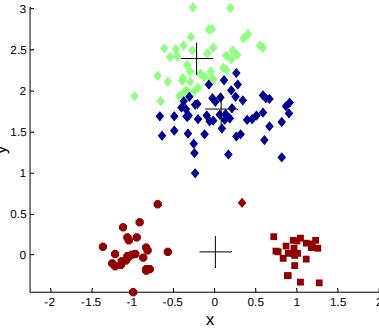
Importance of Choosing Initial Centroids ...



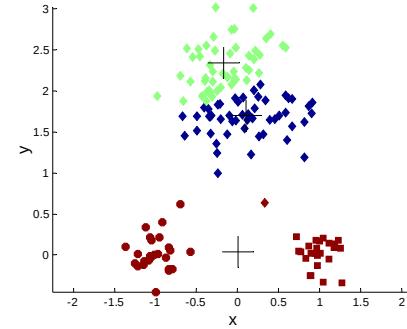
Importance of Choosing Initial Centroids ...



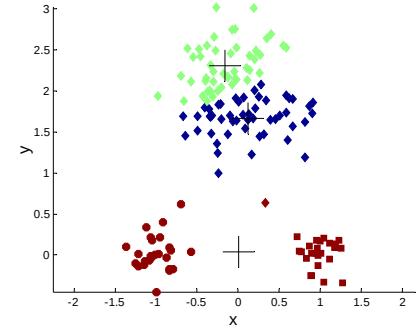
Iteration 3



Iteration 4



Iteration 5

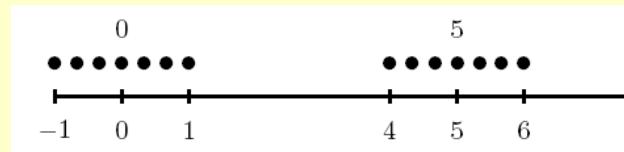


Landscape for Clustering Problem

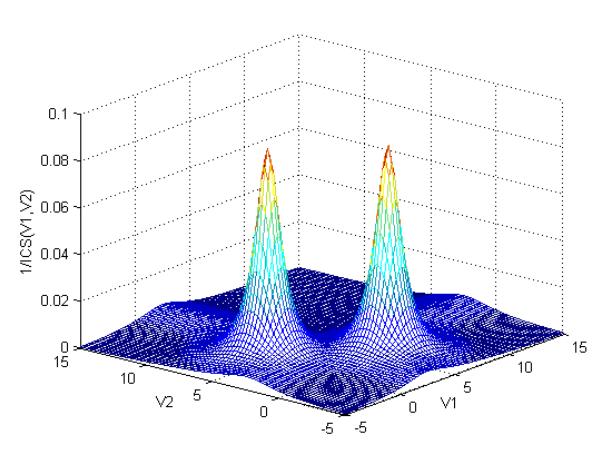
The k-means objective function

$$ICS(C_1, C_2, \dots, C_k) = \sum_{j=1}^k \sum_{Z_i \in C_j} d^2(Z_i, V_j)$$

Example of an extremely simple one-dimensional dataset

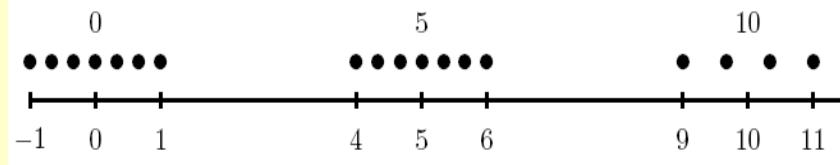


Fitness function (Reciprocal of ICS) plot of the hard c-means algorithm for above dataset

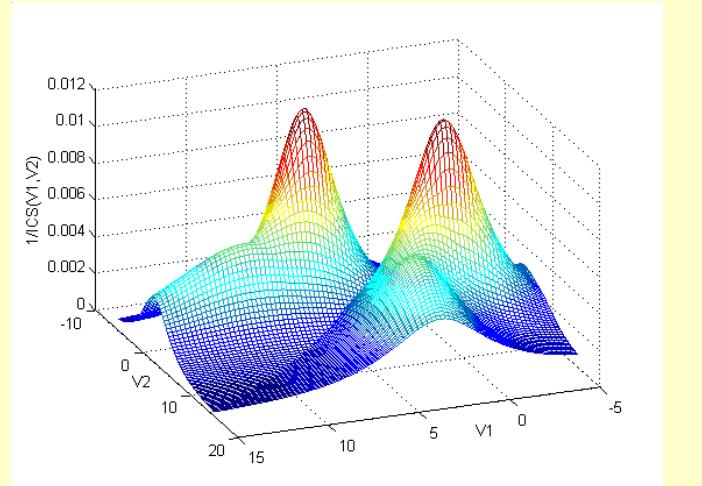


Landscape for Clustering Problem (Contd.)

The previous data but now
With some noise points



Fitness function (Reciprocal of ICS)
plot of the hard c-means algorithm
for above dataset

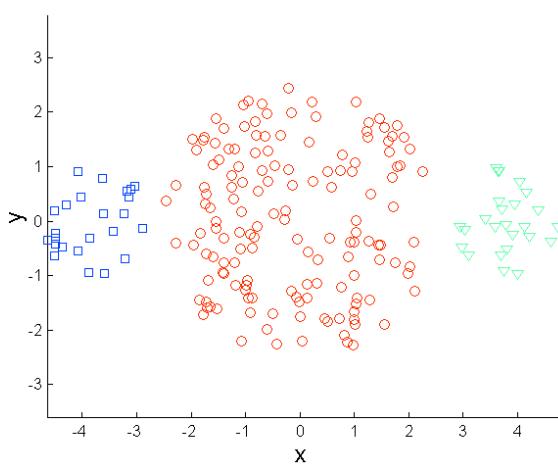


In addition to two global minima, we have local minima at: approximately $(0, 10)$, $(5, 10)$, $(10, 0)$, $(10, 5)$. For these local minima one prototype covers one of the data clusters, while the other prototype is mislead to the noise cluster.

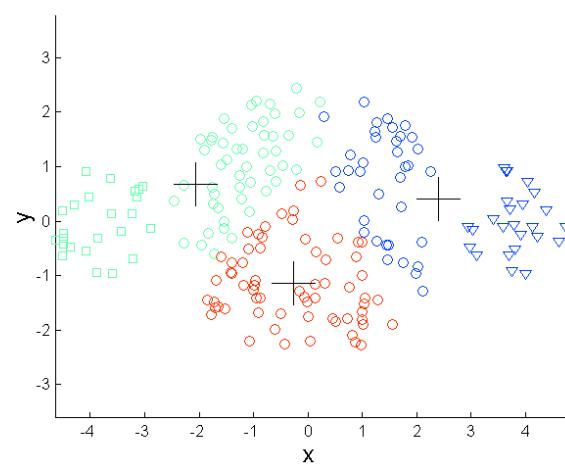
Limitations of K-means

- K-means has problems when clusters are of differing
 - Sizes
 - Densities
 - Non-globular shapes
- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

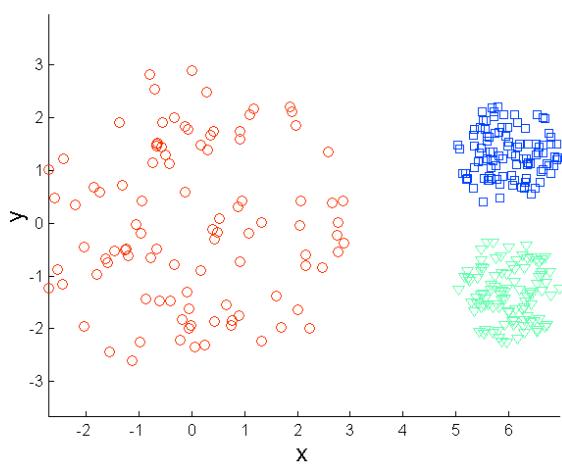


Original Points

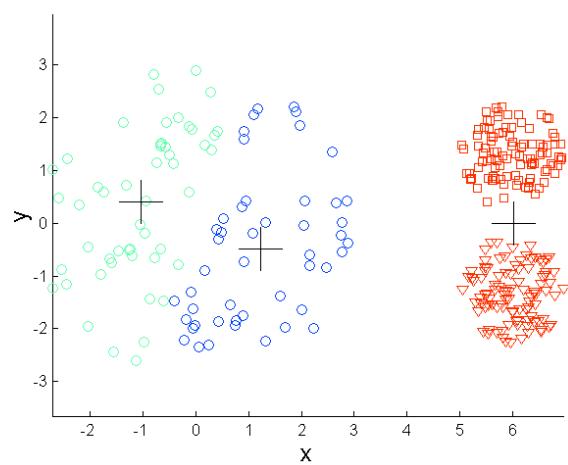


K-means (3 Clusters)

Limitations of K-means: Differing Density

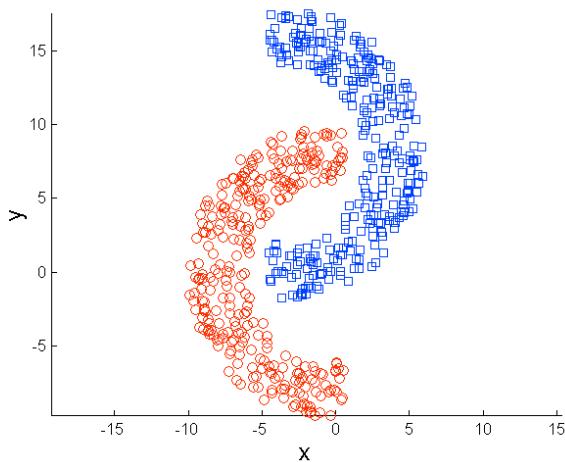


Original Points

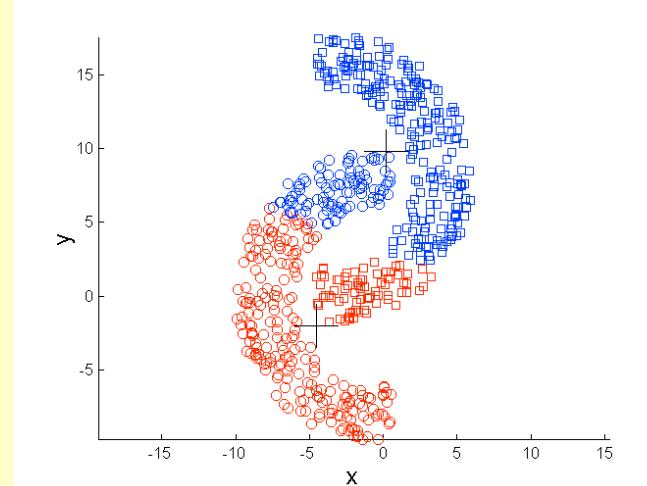


K-means (3 Clusters)

Limitations of K-means: Non-globular and linearly non-separable Shapes



Original Points

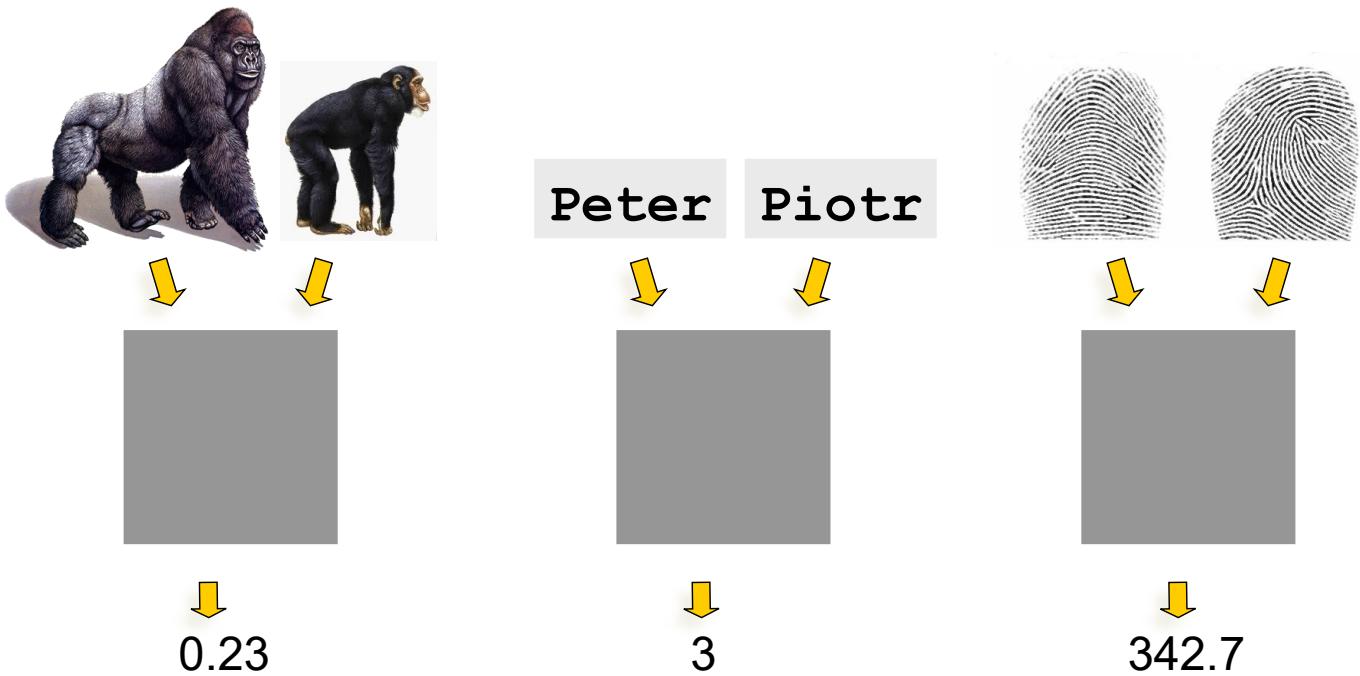


K-means (2 Clusters)

Comments on k-Means

- Strengths
 - Relatively efficient: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t \ll n$.
 - Often terminates at a local optimum.
- Weakness
 - Applicable only when mean is defined, then what about categorical data?
 - Need to specify k , the number of clusters, in advance
 - Unable to handle noisy data and outliers
 - Not suitable to discover clusters with non-convex shapes

How do we measure similarity?



Slide based on one by Eamonn Keogh

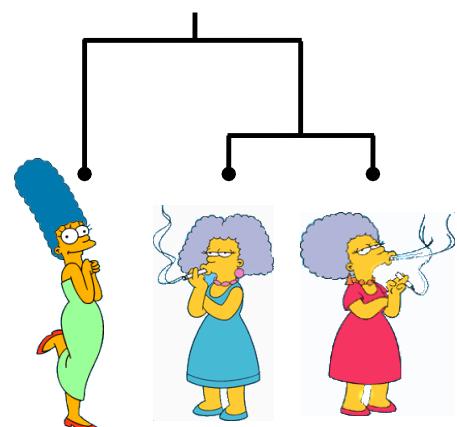
A generic technique for measuring similarity

To measure the similarity between two objects, transform one into the other, and measure how much effort it took. The measure of effort becomes the distance measure.

The distance between Patty and Selma:

Change dress color, 1 point
Change earring shape, 1 point
Change hair part, 1 point

$$D(\text{Patty}, \text{Selma}) = 3$$



The distance between Marge and Selma:

Change dress color, 1 point
Add earrings, 1 point
Decrease height, 1 point
Take up smoking, 1 point
Lose weight, 1 point

$$D(\text{Marge}, \text{Selma}) = 5$$

This is called the “edit distance” or the “transformation distance”

Slide based on one by Eamonn Keogh

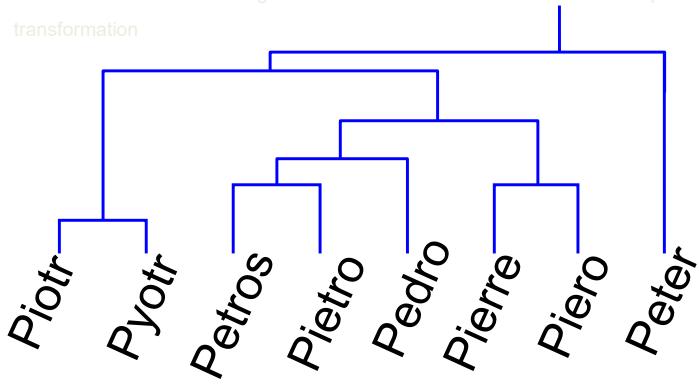
Edit Distance Example

It is possible to transform any string Q into string C , using only *Substitution*, *Insertion* and *Deletion*.

Assume that each of these operators has a cost associated with it.

The similarity between two strings can be defined as the cost of the cheapest transformation from Q to C .

Note that for now we have ignored the issue of how we can find this cheapest transformation



Slide based on one by Eamonn Keogh

How similar are the names “Peter” and “Piotr”?

Assume the following cost function

<i>Substitution</i>	1 Unit
---------------------	--------

<i>Insertion</i>	1 Unit
------------------	--------

<i>Deletion</i>	1 Unit
-----------------	--------

$D(\text{Peter}, \text{Piotr})$ is 3

Peter

Piter

Pioter

Piotr

Substitution of 'e'

Deletion of 'e'

Insertion of 't'

What distance metric did k-means use?

What assumptions is it making about the data?

Partition Algorithm 2: Using a Euclidean Distance Threshold to Define Clusters

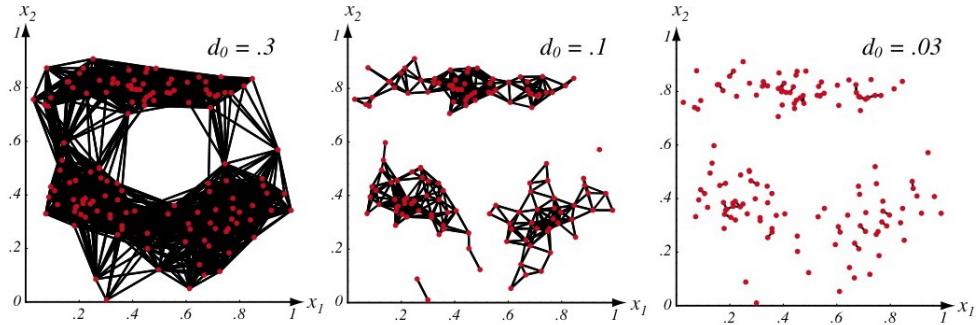


FIGURE 10.7. The distance threshold affects the number and size of clusters in similarity based clustering methods. For three different values of distance d_0 , lines are drawn between points closer than d_0 —the smaller the value of d_0 , the smaller and more numerous the clusters. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

But should we use Euclidean Distance?

Good if data is isotropic and spread evenly along all directions

Not invariant to linear transformations, or any transformation that distorts distance relationships

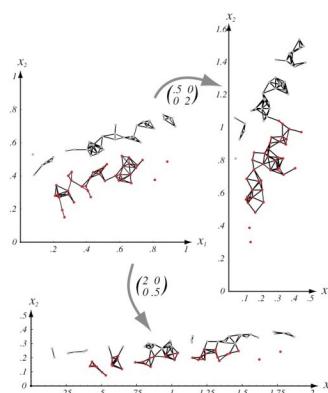
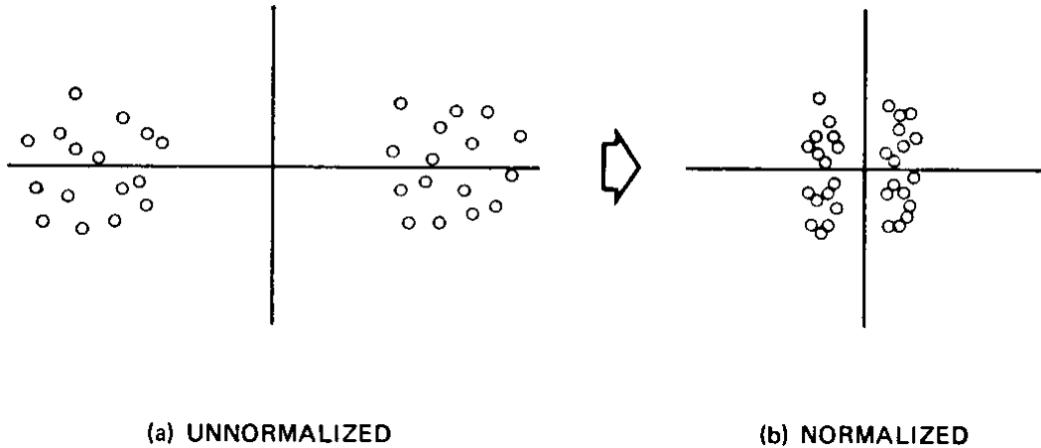


FIGURE 10.8. Scaling axes affects the clusters in a minimum distance cluster method. The original data and minimum-distance clusters are shown in the upper left; points in one cluster are shown in red, while the others are shown in gray. When the vertical axis is expanded by a factor of 2.0 and the horizontal axis shrunk by a factor of 0.5, the clustering is altered (as shown at the right). Alternatively, if the vertical axis is shrunk by a factor of 0.5 and the horizontal axis is expanded by a factor of 2.0, smaller more numerous clusters result (shown at the bottom). In both these scaled cases, the assignment of points to clusters differ from that in the original space. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Is normalization desirable?



Other distance/similarity measures

Distance between two instances x and x' , where $q \geq 1$ is a selectable parameter and d is the number of attributes (called the Minkowski Metric)

$$d(x, x') = \left(\sum_{j=1}^d |x_j - x'_j|^q \right)^{1/q}$$

Cosine of the angle between two vectors (instances) gives a similarity function:

$$s(x, x') = \frac{x' \cdot x'}{\|x\| \|x'\|}$$

Other distance/similarity measures

Distance between two instances x and x' , where $q \geq 1$ is a selectable parameter and d is the number of attributes (called the Minkowski Metric)

$$d(x, x') = \left(\sum_{j=1}^d |x_j - x'_j|^q \right)^{1/q}$$

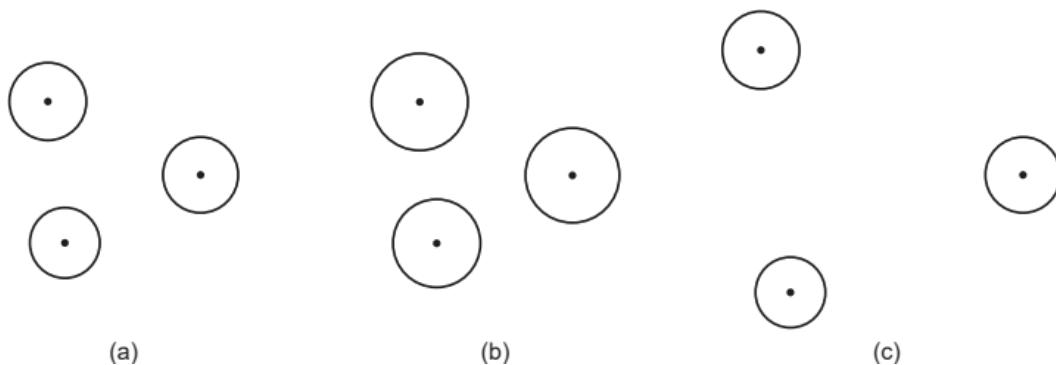
Cosine of the angle between two vectors
a similarity function:

$$s(x, x') = \frac{x' \cdot x'}{\|x\| \|x'\|}$$

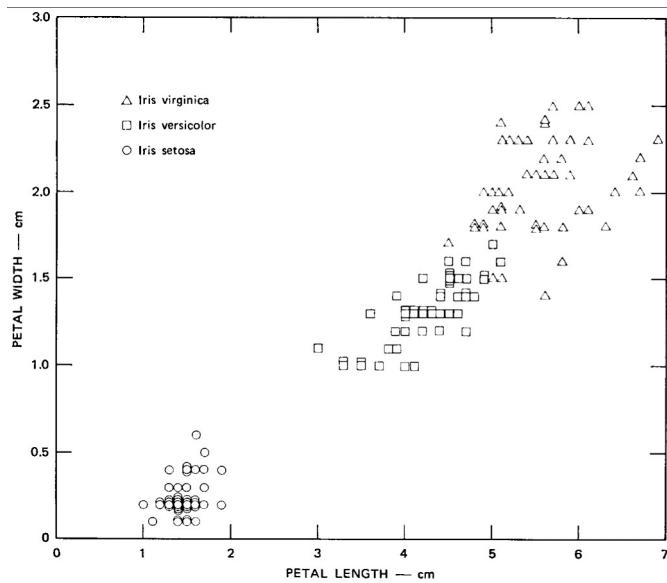
When features are binary
this becomes the number of
attributes shared by x and x'
divided by the geometric mean of
the number of attributes in x and
the number in x' . A simplification
of this is:

$$s(x, x') = \frac{x' \cdot x'}{d}$$

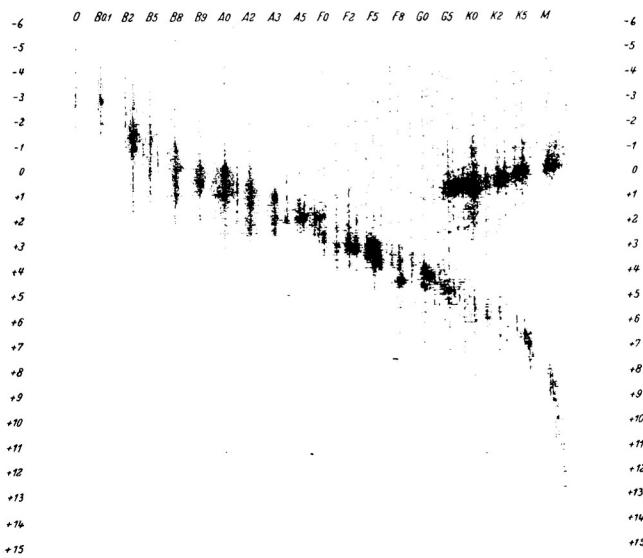
What is a good clustering?



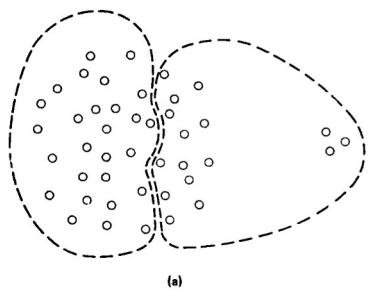
Clustering Criteria: Sum of Squared Error



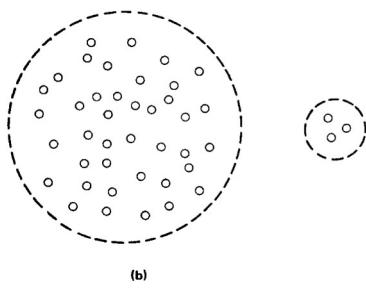
A Dataset for which SSE is not a good criterion.



How does cluster size impact performance?



(a)



(b)

Scattering Criteria – on board

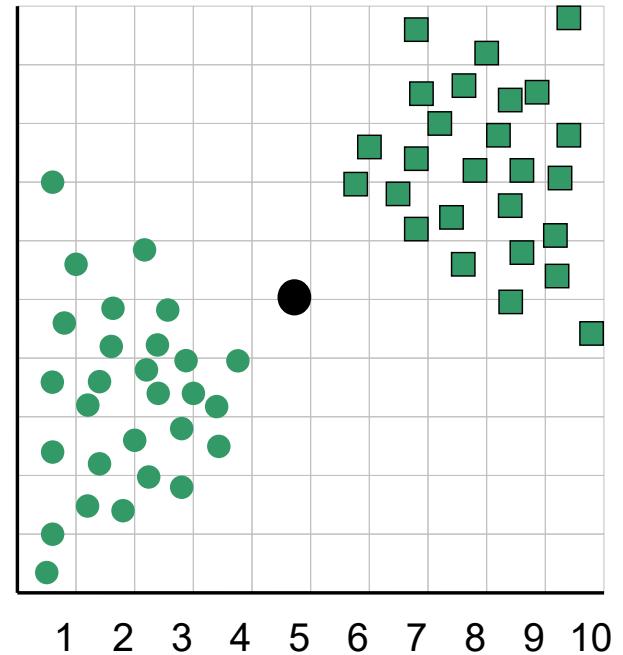
To apply partitional clustering we need to:

- Select features to characterize the data
- Collect representative data
- Choose a clustering algorithm
- Specify the number of clusters

Um, what about k?

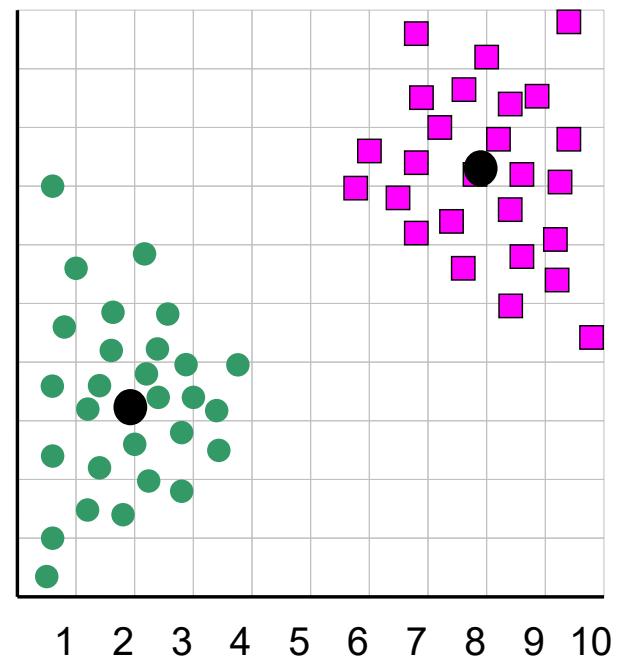
- Idea 1: Use our new trick of cross validation to select k
 - What should we optimize? SSE? Trace?
 - Problem?
- Idea 2: Let our domain expert look at the clustering and decide if they like it
 - How should we show this to them?
 - Problem?
- Idea 3: The “knee” solution

When $k = 1$, the objective function is 873.0



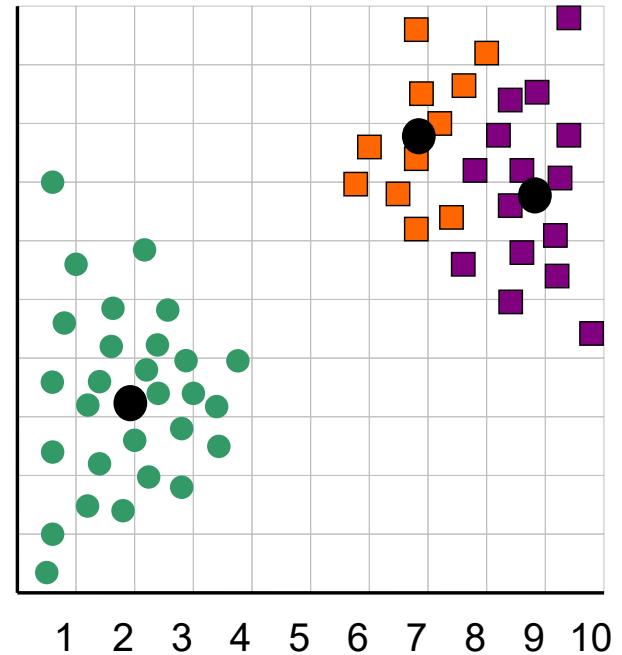
Slide based on one by Eamonn Keogh

When $k = 2$, the objective function is 173.1



Slide based on one by Eamonn Keogh

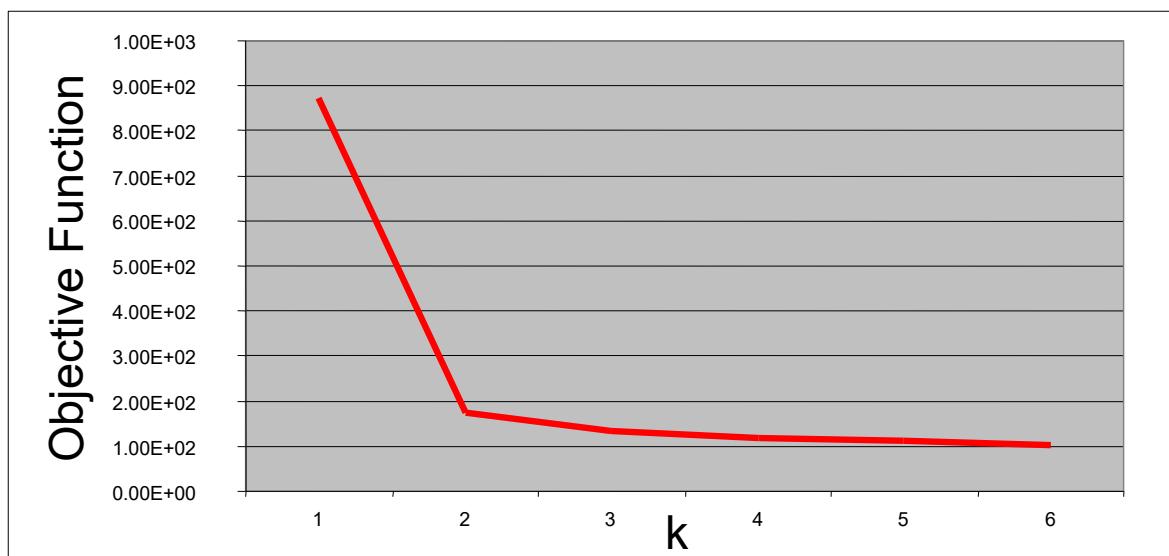
When $k = 3$, the objective function is 133.6



Slide based on one by Eamonn Keogh

We can plot the objective function values for k equals 1 to 6...

The abrupt change at $k = 2$, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as “knee finding” or “elbow finding”.

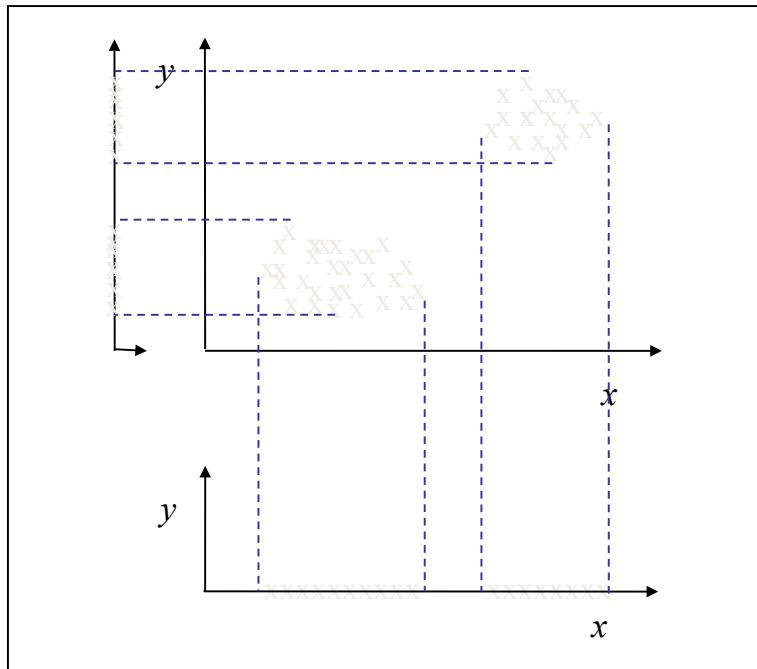


Slide based on one by Eamonn Keogh

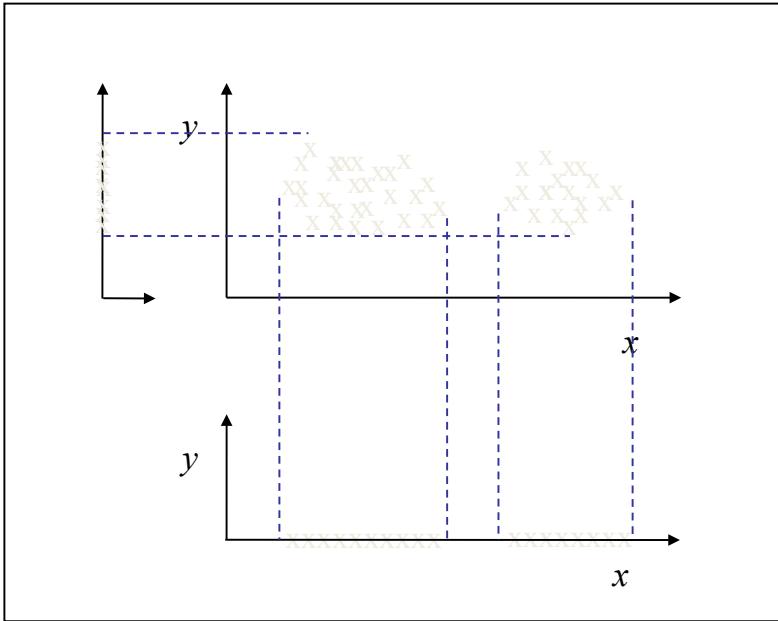
High-Dimensional Data poses Problems for Clustering

- Difficult to find true clusters
 - Irrelevant and redundant features
 - All points are equally close
- Solutions: Dimension Reduction
 - Feature subset selection
 - Cluster ensembles using random projection (in a later lecture....)

Redundant



Irrelevant



Curse of Dimensionality

100 observations cover the 1-D unit interval
[0,1] well

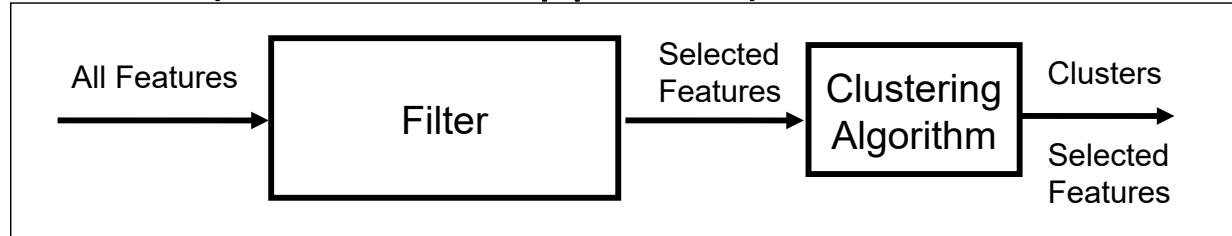
Consider the 10-D unit hypersquare, 100
observations are now isolated points in a
vast empty space.

Consequence of the Curse

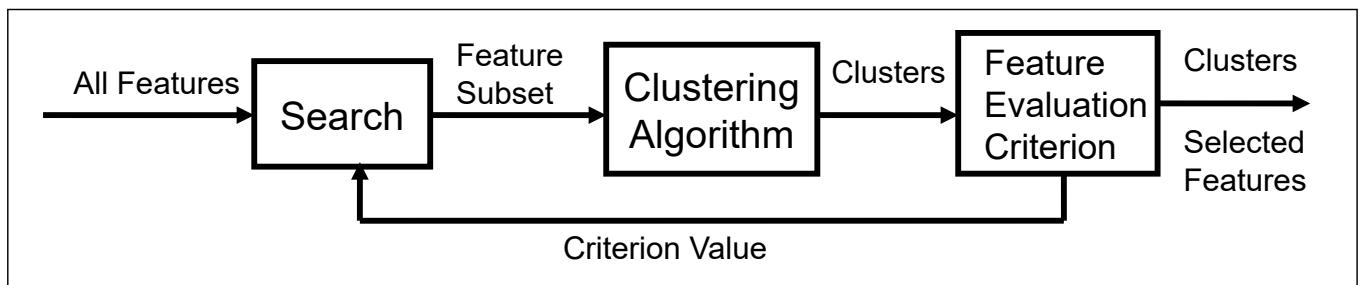
- Suppose the number of samples given to us in the total sample space is fixed
- Let the dimension increase
- Then the distance of the k nearest neighbors of any point increases

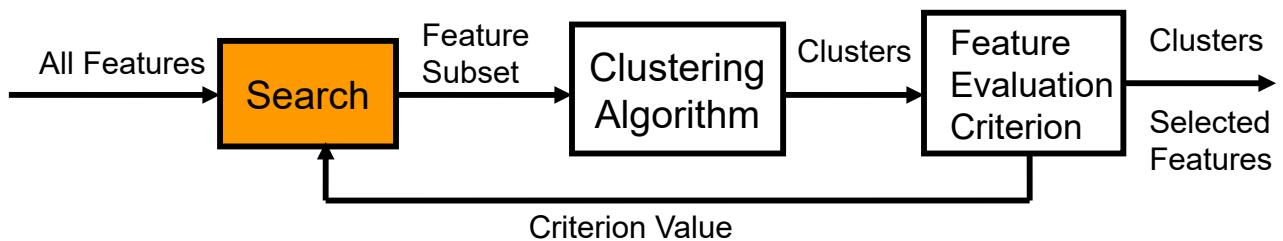
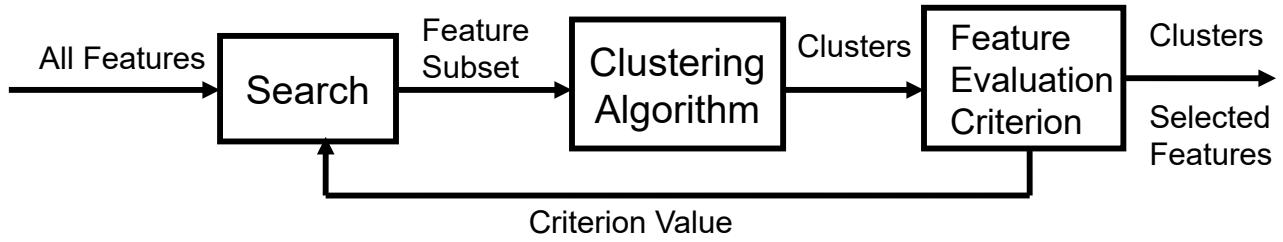
Feature Selection Methods

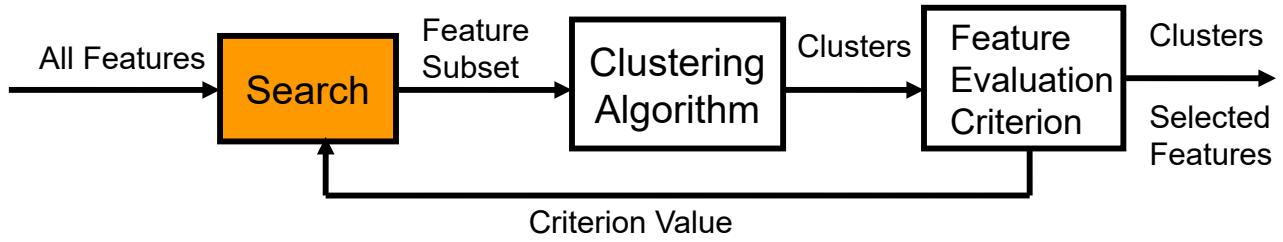
- Filter (Traditional approach)



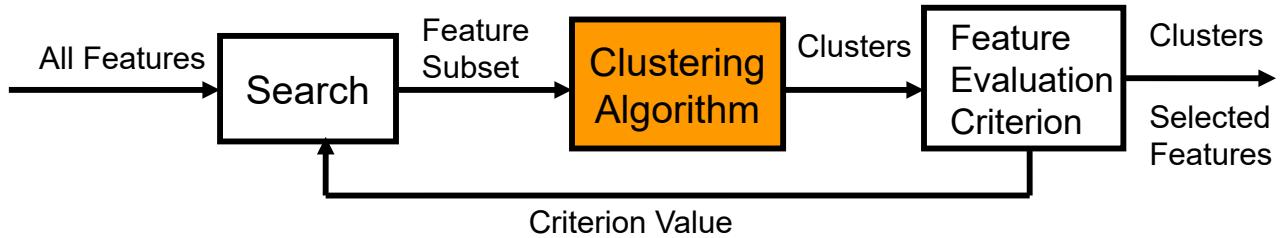
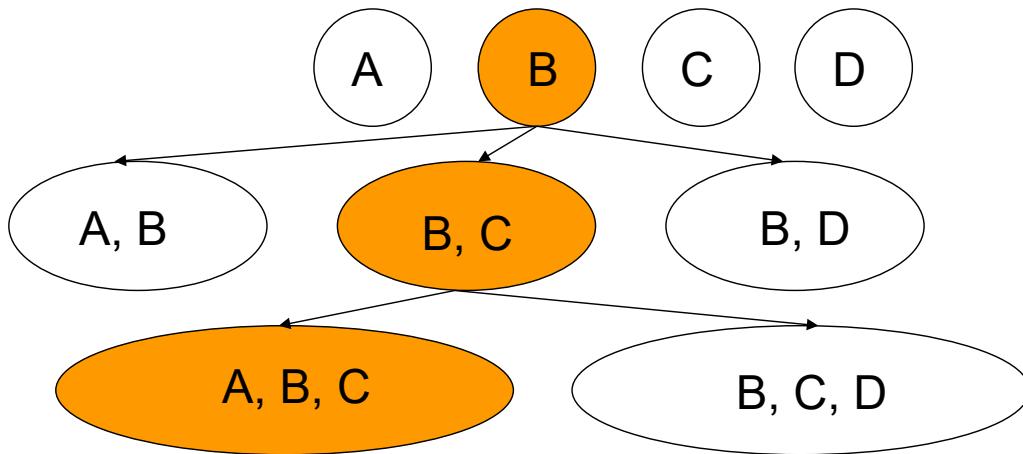
- Apply wrapper approach (Dy and Brodley, 2004)

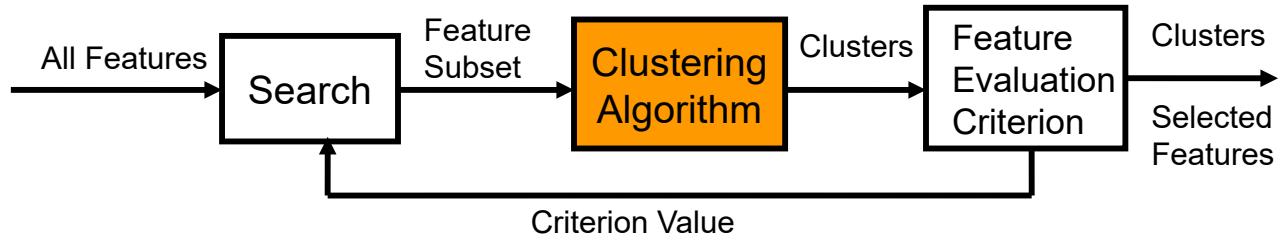






FSSEM Search Method: sequential forward search

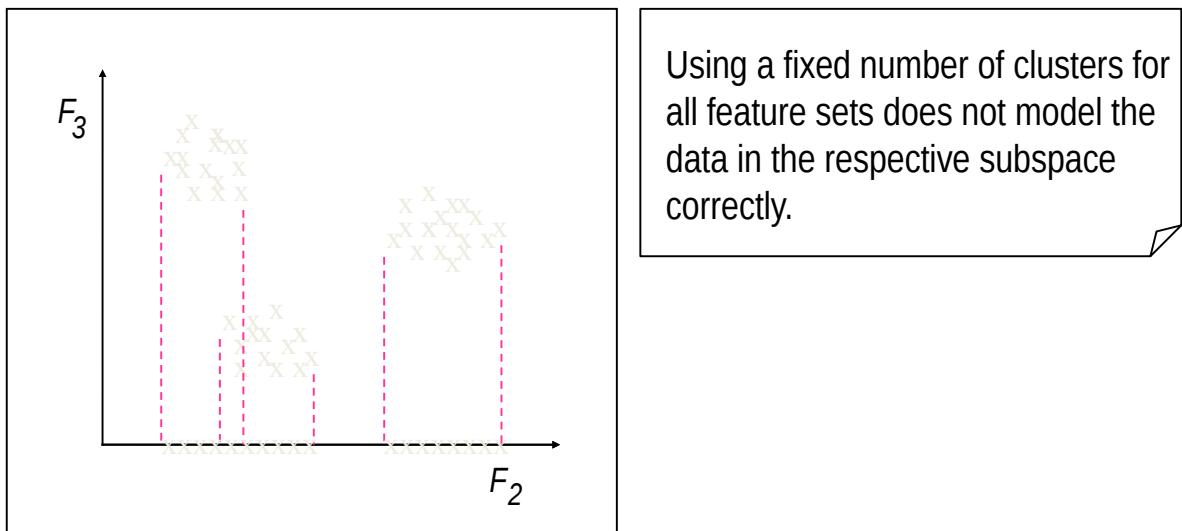


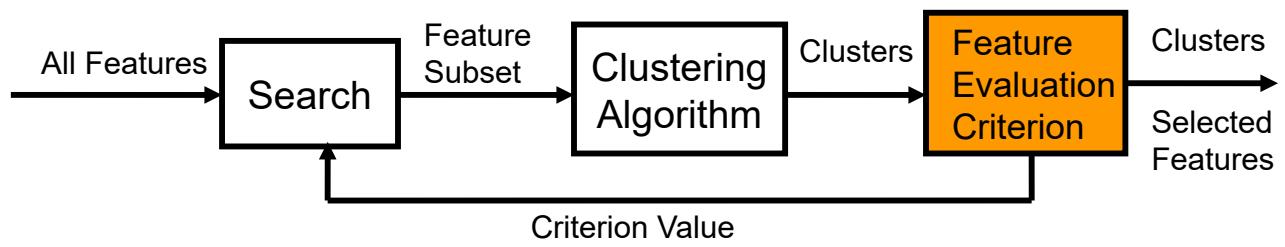


Clustering Algorithm:

Expectation Maximization (EM or EM-k) – coming soon

Searching for the Number of Clusters





Support Vector Machines: A Simple Introduction

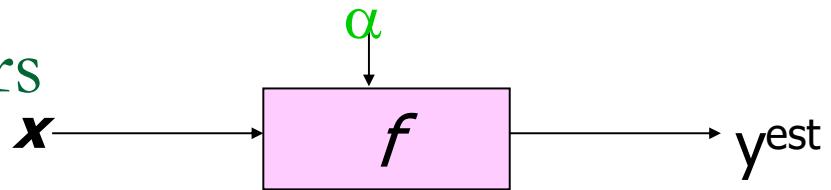
Introduction

- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are **linear classifiers** that find a hyperplane to separate **two class** of data, positive and negative.
- **Kernel functions** are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.
- It is perhaps the best (non-deep) classifier for text classification.

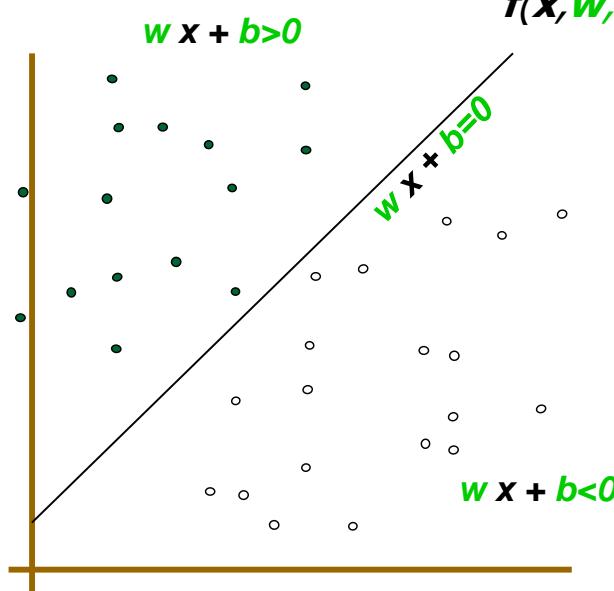
An excellent resource:

https://www.syncfusion.com/ebooks/support_vector_machines_succinctly/solving-the-optimization-problem

Linear Classifiers



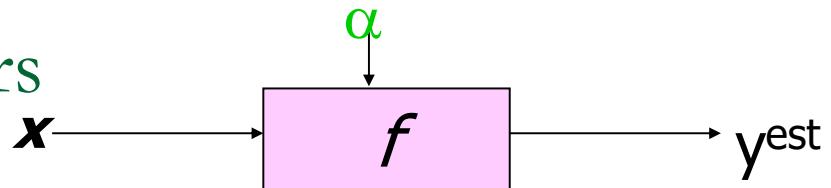
- denotes +1
- denotes -1



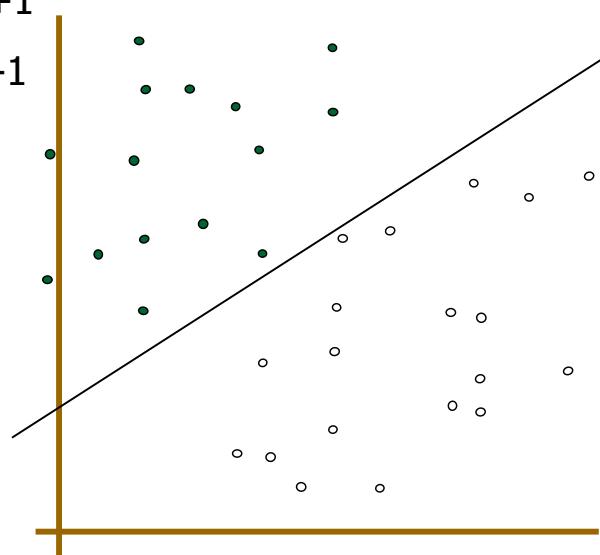
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}\mathbf{x} + b)$$

How would you classify this data?

Linear Classifiers



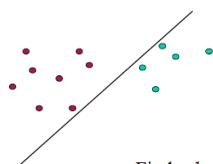
- denotes +1
- denotes -1



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}\mathbf{x} + b)$$

How would you classify this data?

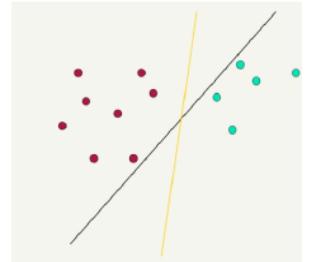
2-D Case



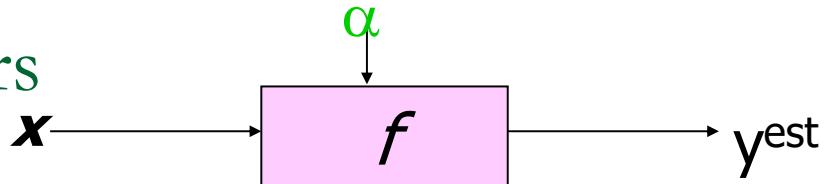
Find a, b, c , such that
 $ax + by \geq c$ for red points
 $ax + by \leq (or <) c$ for green points.

Which Hyperplane to pick?

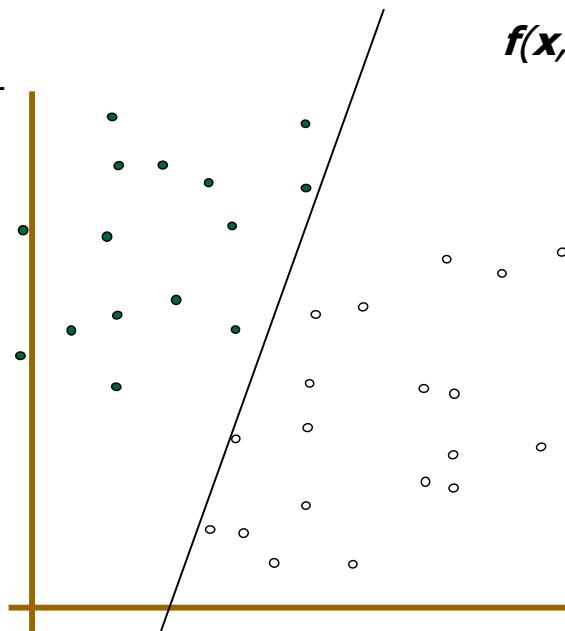
- Lots of possible solutions for a, b, c .
- Some methods find a separating hyperplane, but not the optimal one (e.g., neural net)
- But: Which points should influence optimality?
 - All points?
 - Linear regression
 - Neural nets
 - Or only “difficult points” close to decision boundary
 - Support vector machines



Linear Classifiers



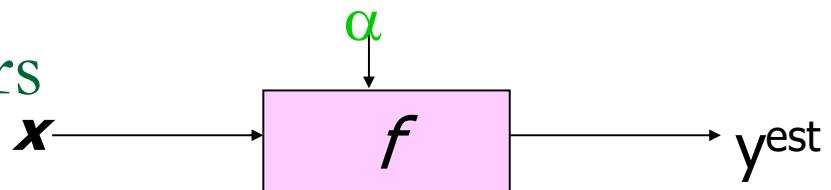
- denotes +1
- denotes -1



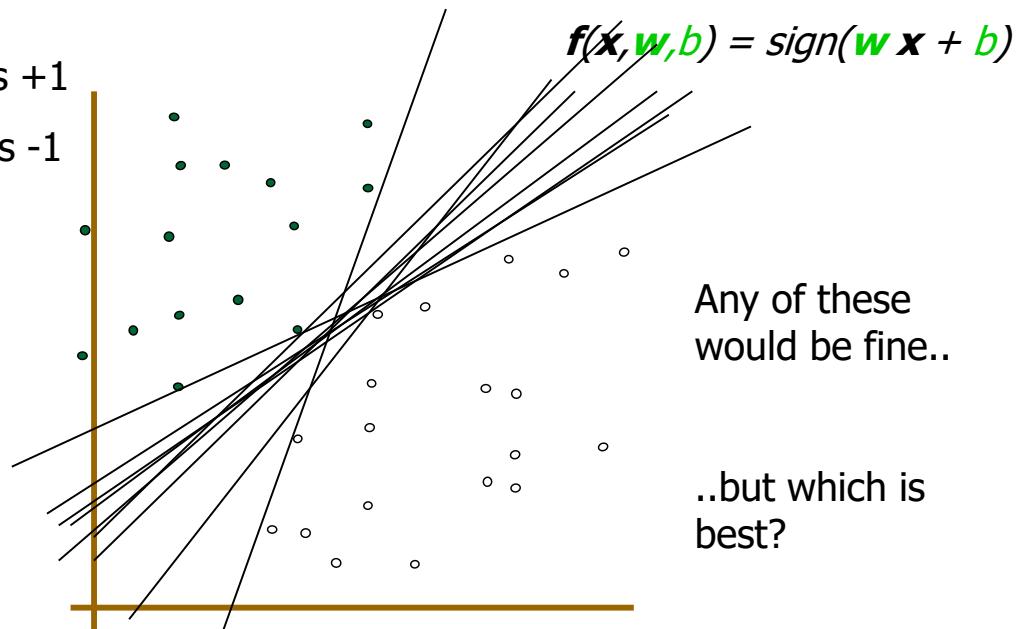
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

How would you
classify this data?

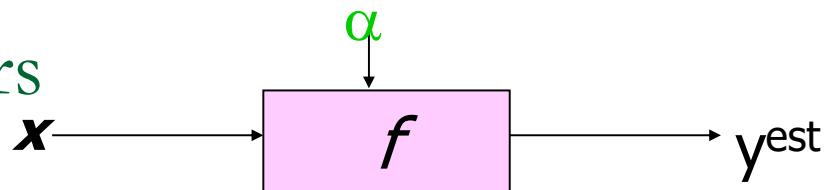
Linear Classifiers



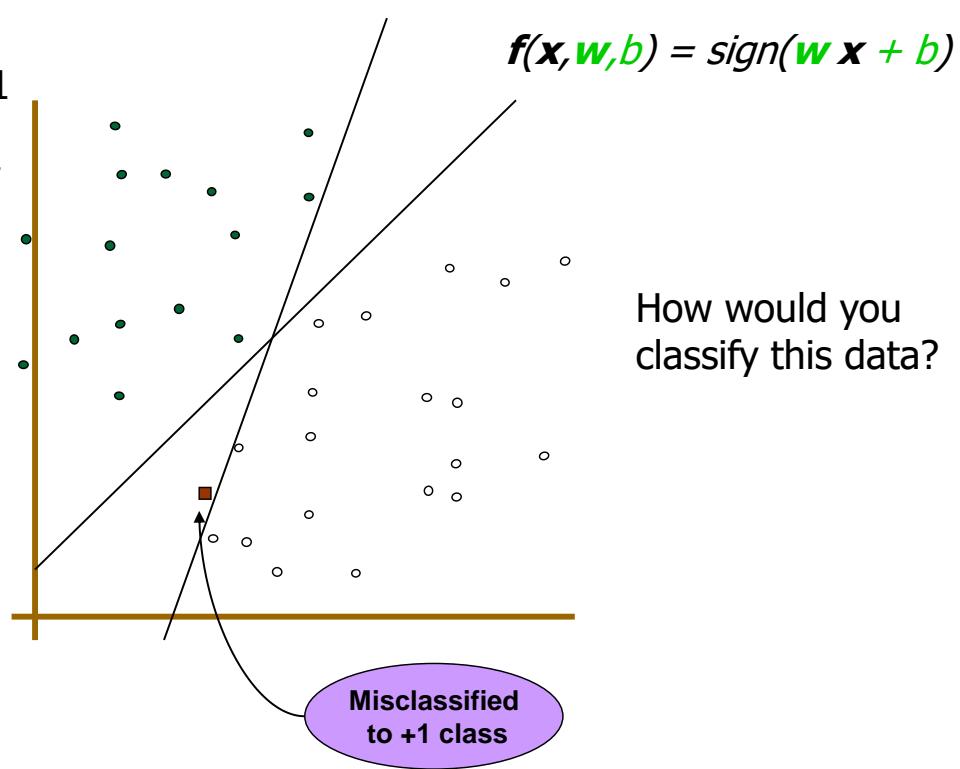
- denotes +1
- denotes -1



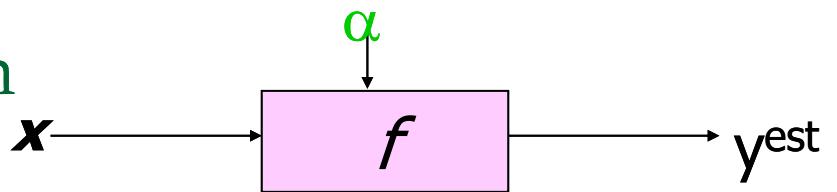
Linear Classifiers



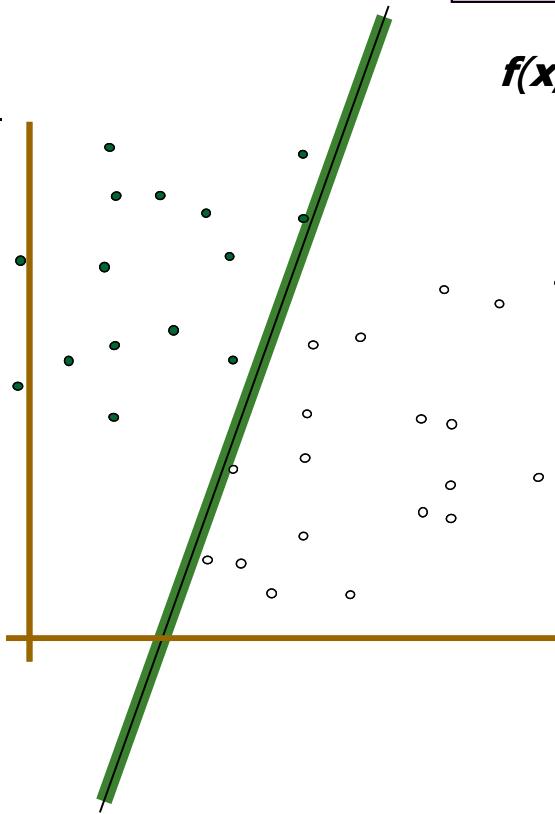
- denotes +1
- denotes -1



Classifier Margin



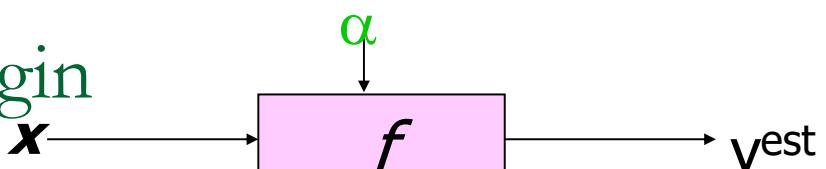
- denotes +1
- denotes -1



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

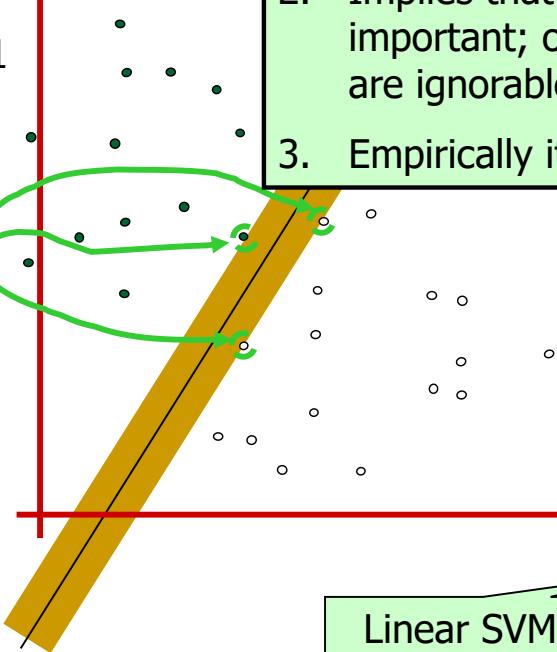
Maximum Margin



- denotes +1
- denotes -1

1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Support Vectors are those datapoints that the margin pushes up against



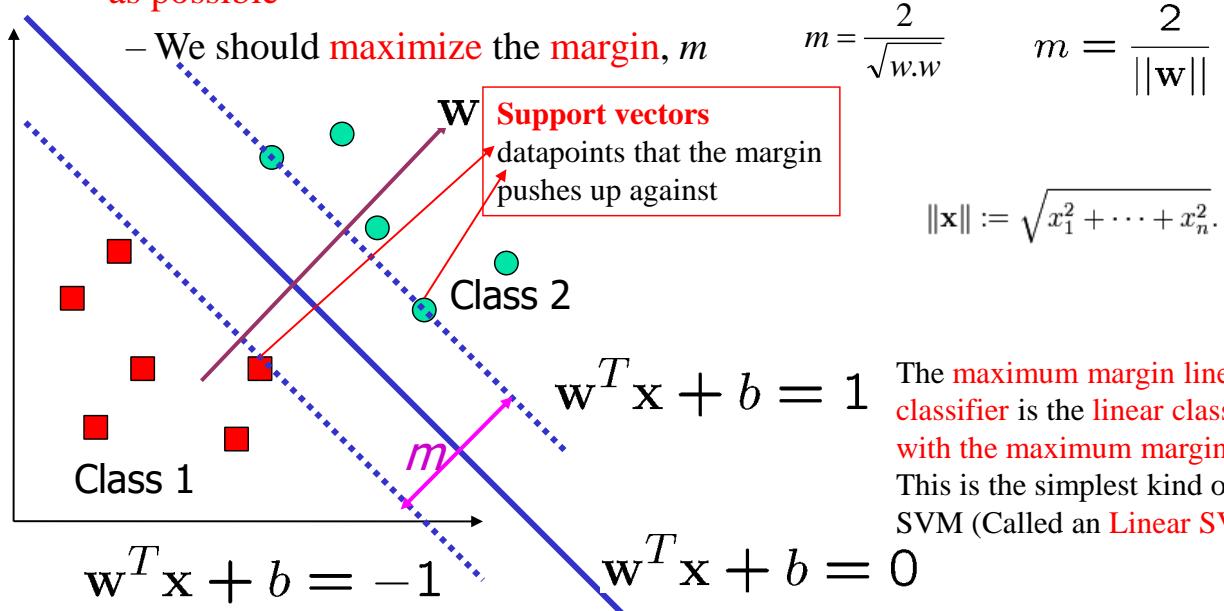
linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Good Decision Boundary: Margin Should Be Large

The decision boundary should be **as far away from the data of both classes as possible**



The **maximum margin linear classifier** is the **linear classifier with the maximum margin**. This is the simplest kind of SVM (Called an **Linear SVM**)

The SVMs optimization problem

- If $F=1$ it will not affect the result of the optimization problem and $M = \frac{2F}{\|w\|}$.
- $$\underset{w,b}{\text{maximize}} \quad M$$
- $$\text{subject to } f_i \geq F, i = 1, \dots, m$$

- The problem becomes:

$$\underset{w,b}{\text{maximize}} \quad \frac{2}{\|w\|}.$$

$$\text{subject to } f_i \geq 1, i = 1, \dots, m$$

This maximization problem is equivalent to the following minimization problem:

$$\underset{w,b}{\text{minimize}} \quad \|w\|.$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1, i = 1, \dots, m$$

The original optimization problem is difficult to solve. Instead,

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2} \|w\|^2.$$

$$\text{subject to } y_i(w \cdot x_i + b) - 1 \geq 0, i = 1, \dots, m$$

It is convex quadratic optimization problem

Throughout the slides, $x^T y = x \cdot y = \text{the 'dot' product of vectors } x \text{ and } y$.

Solving the Optimization Problem

- The method of Lagrange multipliers
- Given an optimization problem:

$$\begin{aligned} & \underset{x}{\text{minimize}} \quad f(x) \\ & \text{subject to } g(x) = 0 \end{aligned}$$

The minimum of f is found when its gradient point in the same direction as the gradient of g :

$$\nabla f(x) = \alpha \nabla g(x)$$

So if we want to find the minimum of f under the constraint g , we just need to solve for:

$$\nabla f(x) - \alpha \nabla g(x) = 0$$

The constant α is called a Lagrange multiplier.

The Lagrange multiplier method can be summarized by these three steps:

1. Construct the Lagrangian function L by introducing one multiplier per constraint
2. Get the gradient ∇L of the Lagrangian
3. Solve for $\nabla L(x, \alpha)=0$

The SVM Lagrangian problem

- SVM optimization problem:

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2} \|w\|^2.$$

$$\text{subject to } y_i(w \cdot x_i + b) - 1 \geq 0, i = 1, \dots, m$$

- Then $f(w) = \frac{1}{2} \|w\|^2$, m constraint functions $g_i(w, b) = y_i(w \cdot x_i + b) - 1, i = 1, \dots, m$

- Lagrangian function:

$$\begin{aligned} L(w, b, \alpha) &= f(w) - \sum_{i=1}^m \alpha_i g_i(w, b) \\ L(w, b, \alpha) &= \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y_i(w \cdot x_i + b) - 1] \end{aligned}$$

Note that we introduced one Lagrange multiplier α_i for each constraint function.

To get the solution of the primal problem, we need to solve the following Lagrangian problem.

$$\begin{aligned} & \min_{w,b} \max_{\alpha} L(w, b, \alpha). \\ & \text{subject to } \alpha_i \geq 0, \quad i = 1, \dots, m \end{aligned}$$

To sum up, if a solution satisfies the KKT conditions, we are guaranteed that it is the optimal solution.

The Karush-Kuhn-Tucker conditions are:

- Stationarity condition:

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{i=1}^m \alpha_i y_i = 0$$

- Primal feasibility condition:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \text{for all } i = 1, \dots, m$$

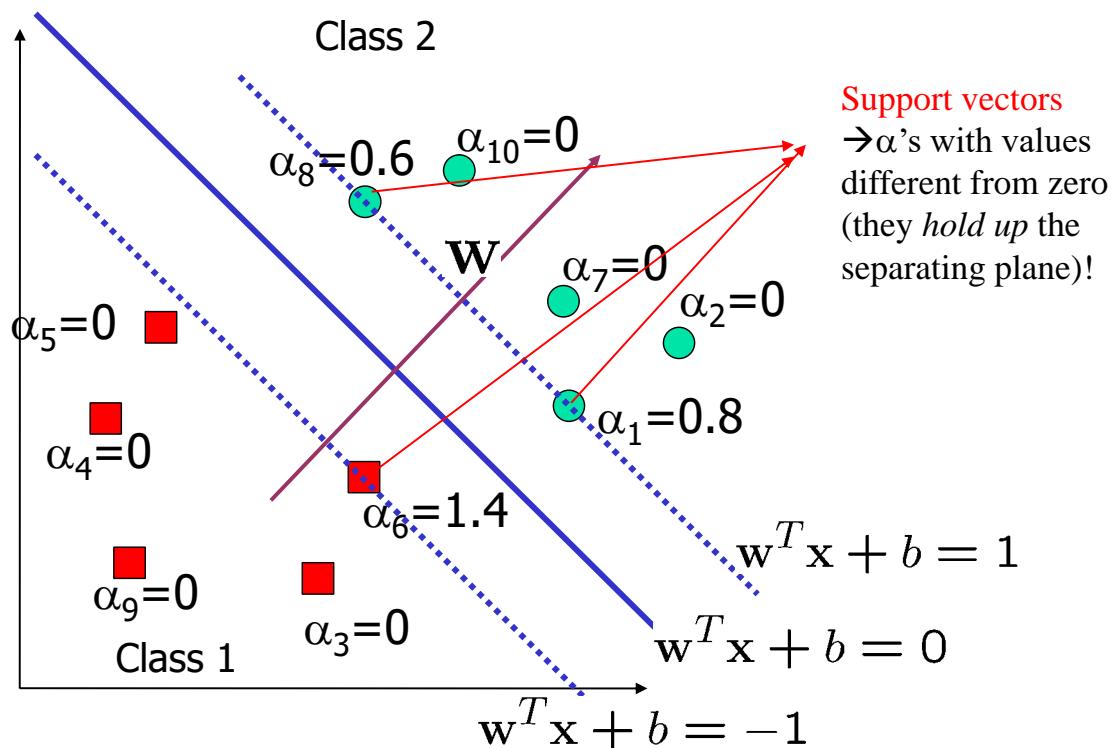
- Dual feasibility condition:

$$\alpha_i \geq 0 \quad \text{for all } i = 1, \dots, m$$

- Complementary slackness condition:

$$\alpha_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0 \quad \text{for all } i = 1, \dots, m$$

A Geometrical Interpretation



Lagrange Dual Function

- **Definition:** the (Lagrange) dual function associated to the constraint optimization problem is defined by

$$\begin{aligned}\forall \alpha \geq 0, F(\alpha) &= \inf_{\mathbf{x} \in X} L(\mathbf{x}, \alpha) \\ &= \inf_{\mathbf{x} \in X} f(\mathbf{x}) + \sum_{i=1}^m \alpha_i g_i(\mathbf{x}).\end{aligned}$$

- F is always concave: Lagrangian is linear with respect to α and \inf preserves concavity.
- $\forall \alpha \geq 0, F(\alpha) \leq p^*$: for a feasible \mathbf{x} ,

$$f(\mathbf{x}) + \sum_{i=1}^m \alpha_i g_i(\mathbf{x}) \leq f(\mathbf{x}).$$

Slide: [Mehryar Mohri - Introduction to Machine Learning](#)

Dual Optimization Problem

- **Definition:** the dual (optimization) problem associated to the constraint optimization is

$$\max_{\alpha} F(\alpha)$$

subject to: $\alpha \geq 0$.

- always a convex optimization problem.
- optimal value d^* .

Weak and Strong Duality

- Weak duality: $d^* \leq p^*$.
 - always holds (clear from previous observations).
- Strong duality: $d^* = p^*$.
 - does not hold in general.
 - holds for convex problems with constraint qualifications.

Slater's constraint qualification condition for convex programming states that strong duality holds if there exists an x that is strictly feasible (i.e. all constraints are satisfied and the nonlinear constraints are satisfied with strict inequalities).

Saddle Point - Sufficient Condition

(Lagrange, 1797)

- **Theorem:** Let P be a constrained optimization problem over $X = \mathbb{R}^N$. If (x^*, α^*) is a saddle point, that is $\forall x \in \mathbb{R}^N, \forall \alpha \geq 0, L(x^*, \alpha) \leq L(x^*, \alpha^*) \leq L(x, \alpha^*)$, then it is a solution of P .

The Wolfe dual problem

- The Lagrangian problem has m inequality constraints (where m is the number of training examples) and is typically solved using its dual form.
- According to duality principle the maximum of the dual problem will always be less than or equal to the minimum of the primal problem (it provides a lower bound to the solution of the primal problem)
- Lagrangian function

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y_i(w \cdot x_i + b) - 1]$$

Solving the minimization problem involves taking the partial derivatives of L with respect to w and b :

$$\begin{aligned}\nabla_w L &= w - \sum_{i=1}^m \alpha_i y_i x_i = 0 & \rightarrow & w = \sum_{i=1}^m \alpha_i y_i x_i \\ \nabla_b L &= -\sum_{i=1}^m \alpha_i y_i = 0 & \rightarrow & \sum_{i=1}^m \alpha_i y_i = 0\end{aligned}$$

Let us substitute w by this value L and we get:

$$W(a, b) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

This is the **Wolfe dual Lagrangian function**

Optional

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y_i(w \cdot x_i + b) - 1] \quad w = \sum_{i=1}^m \alpha_i y_i x_i$$

Let us substitute w by this value into \mathcal{L} :

$$\begin{aligned}W(\alpha, b) &= \frac{1}{2} \left(\sum_{i=1}^m \alpha_i y_i x_i \right) \cdot \left(\sum_{j=1}^m \alpha_j y_j x_j \right) - \sum_{i=1}^m \alpha_i \left[y_i \left(\left(\sum_{j=1}^m \alpha_j y_j x_j \right) \cdot x_i + b \right) - 1 \right] \\ &= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i=1}^m \alpha_i y_i \left(\left(\sum_{j=1}^m \alpha_j y_j x_j \right) \cdot x_i + b \right) + \sum_{i=1}^m \alpha_i \\ &= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - b \sum_{i=1}^m \alpha_i y_i + \sum_{i=1}^m \alpha_i \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j - b \sum_{i=1}^m \alpha_i y_i\end{aligned}$$

Wolfe dual problem

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ & \text{subject to } \alpha_i \geq 0, \quad i = 1, \dots, m \\ & \quad \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned}$$

The main advantage of the Wolfe dual problem over the Lagrangian problem is that the objective function W now depends only on the Lagrange multipliers.

When we solve the Wolfe dual problem, we get a vector α containing all Lagrange multipliers and then we will compute w and b .

Compute w :

$$w = \sum_{i=1}^m \alpha_i y_i x_i$$

Compute b :

$$\begin{aligned} w \cdot x_i + b &= y_i \\ b &= y_i - w \cdot x_i \end{aligned}$$

Hypothesis function

The SVMs use the same hypothesis function as the perceptron. The class of an example

$$h(x_i) = \text{sign}(w \cdot x_i + b)$$

When using the dual formulation, it is computed using only the support vectors:

$$h(x_i) = \text{sign} \left(\sum_{j=1}^S \alpha_j y_j (x_j \cdot x_i) + b \right)$$

This formulation of the SVM is called the **hard margin SVM**. It cannot work when the data is not linearly separable. Soft margin SVM applied non-linearly separable data.

The Wolfe Dual Optimization Problem

We can transform the problem to its dual

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

Dot product of X

α 's → New variables
(Lagrangian multipliers)

This is a convex quadratic programming (QP) problem

- Global maximum of α_i can always be found
- well established tools for solving this optimization problem (e.g. cplex)

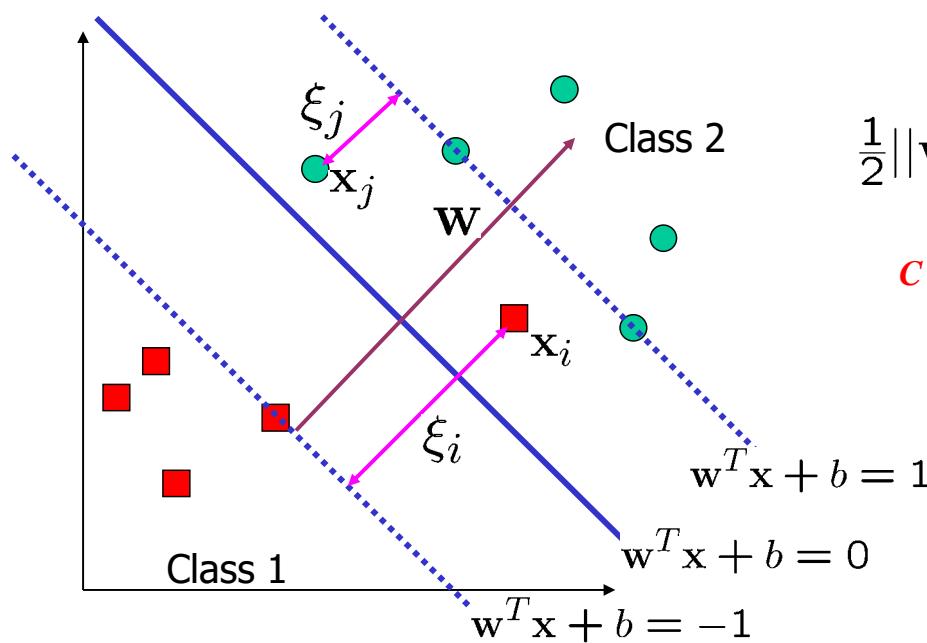
Note:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

Non-linearly Separable Problems

We allow “error” ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T \mathbf{x} + b$

ξ_i approximates the number of misclassified samples



New objective function:

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

C : tradeoff parameter between error and margin;
chosen by the user;
large C means a higher penalty to errors

Soft margin SVM

Slack variables

Given a training set: $D = \{(x_i, y_i) \mid x_i \in R^n, y_i \in \{-1, 1\}\}_{i=1}^m$. In 1995, Vapnik and Cortes introduced a modified version of the original SVM that allows the classifier to make some mistakes. The goal is now not to make zero classification mistakes, but to make as few mistakes as possible. To do so, they modified the constraints of the optimization problem by adding a variable ξ .

$$\begin{aligned} & \underset{w, b, \xi}{\text{minimize}} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ & \text{subject to } y_i(w \cdot x_i + b) \geq 1 - \xi_i, \\ & \quad \xi_i \geq 0 \quad i = 1, \dots, m \end{aligned}$$

where ξ_i -slack variables, C is the regularization parameter.

Wolfe dual problem

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ & \text{subject to} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \\ & \quad \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned}$$

The optimization problem is also called **1-norm soft margin** because we are minimizing the 1-norm of the slack vector ξ .

The Wolfe dual problem

$$\begin{aligned} & \max. \quad W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ & \text{subject to} \quad C \geq \alpha_i \geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0 \\ & \quad \mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j} \end{aligned}$$

\mathbf{w} is also recovered as

The only difference with the linear separable case is that there is an upper bound C on α_i

Once again, a QP solver can be used to find α_i efficiently!!!

Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points \mathbf{x}_i are support vectors with non-zero Lagrangian multipliers α_i .
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Probability Density Estimation:

MLE (Maximum Likelihood Estimation)

and

MAP (Maximum A Posteriori) Estimation

Probability that Liverpool FC wins a match in the next season

In 2018-19 season, Liverpool FC won 30 matches out of 38 matches in Premier league. Having this data, we'd like to make a guess at the probability that Liverpool FC wins a match in the next season.

The simplest guess here would be $30/38 = 79\%$, which is the best possible guess based on the data. This actually is an estimation with **MLE** method.

Then, assume we know that Liverpool's winning percentages for the past few seasons were around 50%.

Do you think our best guess is still 79%? I think some value between 50% and 79% would be more realistic, considering the prior knowledge as well as the data from this season. This is an estimation with **MAP** method.

MLE and MAP are methods of estimating **parameters of statistical models**.

In this example, we're simplifying that Liverpool has a single winning probability (let's call this as θ) throughout all matches across seasons, regardless of uniqueness of each match and any complex factors of real football matches. On the other words, we're assuming each of Liverpool's match as a Bernoulli trial with the winning probability θ .

With this assumption, we can describe **probability that Liverpool wins k times out of n matches** for any given number k and n ($k \leq n$). More precisely, we assume that the number of wins of Liverpool follows **binomial distribution** with parameter θ . The formula of the probability that Liverpool wins k times out of n matches, given the winning probability θ , is below.

$$P(k \text{ wins out of } n \text{ matches} | \theta) = \binom{n}{k} \theta^k (1 - \theta)^{n-k}$$

This simplification (describing the probability using just a single parameter θ regardless of real world complexity) is the statistical modelling of this example, and θ is the parameter to be estimated.

Maximum Likelihood Estimation (MLE):

In the previous section, we got the formula of probability that Liverpool wins k times out of n matches for given θ . Since we have the observed data from this season, which is *30 wins out of 38 matches* (let's call this data as D), we can calculate $P(D|\theta)$ — the probability that this data D is observed for given θ . Let's calculate $P(D|\theta)$ for $\theta=0.1$ and $\theta=0.7$ as examples.

When Liverpool's winning probability $\theta = 0.1$, the probability that this data D (*30 wins in 38 matches*) is observed is following.

$$P(30 \text{ wins in } 38 \text{ matches} | \theta = 0.1) = \binom{38}{30} 0.1^{30} (1 - 0.1)^{38-30} = 2.11 \times 10^{-21}$$

$P(D|\theta) = 0.0000000000000000000211$. So, if Liverpool's winning probability θ is actually 0.1, this data D (30 wins in 38 matches) is extremely unlikely to be observed. Then what if $\theta = 0.7$?

$$P(30 \text{ wins in } 38 \text{ matches} | \theta = 0.7) = \binom{38}{30} 0.7^{30} (1 - 0.7)^{38-30} = 0.072$$

Much **higher** than previous one. So if Liverpool's winning probability θ is 0.7, this data D is much more likely to be observed than when $\theta = 0.1$.

Based on this comparison, we would be able to say that θ is more likely to be 0.7 than 0.1 considering the actual observed data D .

Here, we've been calculating the probability that D is observed for each θ , but at the same time, we can also say that we've been checking likelihood of each value of θ based on the observed data.

Because of this, $P(D|\theta)$ is also considered as **Likelihood** of θ . The next question here is, what is the exact value of θ which maximise the likelihood $P(D|\theta)$? Yes, this is the **Maximum Likelihood Estimation!**

The value of θ maximising the likelihood can be obtained by having derivative of the likelihood function with respect to θ , and setting it to zero.

$$\begin{aligned} \frac{dP(D|\theta)}{d\theta} &= \binom{n}{k} (k\theta^{k-1}(1-\theta)^{n-k} - (n-k)\theta^k(1-\theta)^{n-k-1}) \\ &= \binom{n}{k} \theta^{k-1}(1-\theta)^{n-k-1} (k(1-\theta) - (n-k)\theta) \\ &= 0 \end{aligned}$$

By solving this, $\theta = 0, 1$ or k/n . Since likelihood goes to zero when $\theta = 0$ or 1 , the value of θ maximize the likelihood is k/n .

$$\theta = \frac{k}{n}$$

In this example, the estimated value of θ is $30/38 = 78.9\%$ when estimated with MLE.

Likelihood of Data

- Consider n I.I.D. random variables X_1, X_2, \dots, X_n
 - X_i is a sample from density function $f(X_i | \theta)$
 - Note: now explicitly specify parameter θ of distribution
 - We want to determine how “likely” the observed data (x_1, x_2, \dots, x_n) is based on density $f(X_i | \theta)$
 - Define the **Likelihood function**, $L(\theta)$:

$$L(\theta) = \prod_{i=1}^n f(X_i | \theta)$$

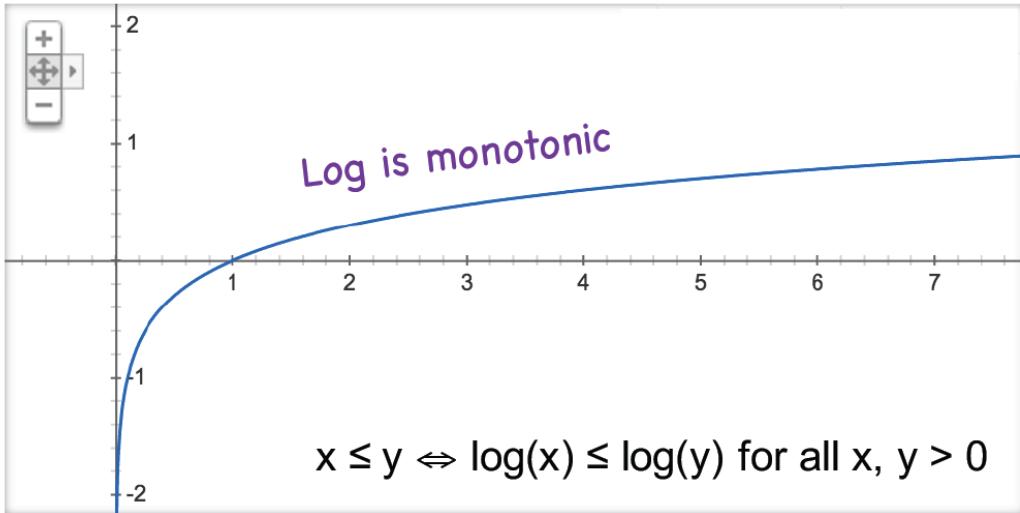
- This is just a product since X_i are I.I.D.
- Intuitively: what is probability of observed data using density function $f(X_i | \theta)$, for some choice of θ

Maximum Likelihood Estimator

- The **Maximum Likelihood Estimator** (MLE) of θ , is the value of θ that maximizes $L(\theta)$
 - More formally: $\theta_{MLE} = \arg \max_{\theta} L(\theta)$

Argmax of Log

Graph for $\log(x)$



Claim: $\operatorname{argmax}_x f(x) = \operatorname{argmax}_x \log f(x)$

Maximum Likelihood Estimator

- The **Maximum Likelihood Estimator** (MLE) of θ , is the value of θ that maximizes $L(\theta)$
 - More formally: $\theta_{MLE} = \arg \max_{\theta} L(\theta)$
 - More convenient to use **log-likelihood function**, $LL(\theta)$:
$$LL(\theta) = \log L(\theta) = \log \prod_{i=1}^n f(X_i | \theta) = \sum_{i=1}^n \log f(X_i | \theta)$$
 - Note that *log* function is “monotone” for positive values
 - Formally: $x \leq y \Leftrightarrow \log(x) \leq \log(y)$ for all $x, y > 0$
 - So, θ that maximizes $LL(\theta)$ also maximizes $L(\theta)$
 - Formally: $\arg \max_{\theta} LL(\theta) = \arg \max_{\theta} L(\theta)$
 - Similarly, for any positive constant c (not dependent on θ):
$$\arg \max_{\theta} (c \cdot LL(\theta)) = \arg \max_{\theta} LL(\theta) = \arg \max_{\theta} L(\theta)$$

Computing the MLE

- General approach for finding MLE of θ
 - Determine formula for $LL(\theta)$
 - Differentiate $LL(\theta)$ w.r.t. (each) θ : $\frac{\partial LL(\theta)}{\partial \theta}$
 - To maximize, set $\frac{\partial LL(\theta)}{\partial \theta} = 0$
 - Solve resulting (simultaneous) equations to get θ_{MLE}
 - Make sure that derived $\hat{\theta}_{MLE}$ is actually a maximum (and not a minimum or saddle point). E.g., check $LL(\theta_{MLE} \pm \varepsilon) < LL(\theta_{MLE})$
 - This step often ignored in expository derivations
 - So, we'll ignore it here too (and won't require it in this class)

Maximizing Likelihood with Poisson

- Consider I.I.D. random variables X_1, X_2, \dots, X_n
 - $X_i \sim \text{Poi}(\lambda)$
 - PMF: $f(X_i | \lambda) = \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}$ Likelihood: $L(\theta) = \prod_{i=1}^n \frac{e^{-\lambda} \lambda^{X_i}}{X_i!}$
 - Log-likelihood:
$$\begin{aligned} LL(\theta) &= \sum_{i=1}^n \log\left(\frac{e^{-\lambda} \lambda^{X_i}}{X_i!}\right) = \sum_{i=1}^n [-\lambda \log(e) + X_i \log(\lambda) - \log(X_i!)] \\ &= -n\lambda + \log(\lambda) \sum_{i=1}^n X_i - \sum_{i=1}^n \log(X_i!) \end{aligned}$$
 - Differentiate w.r.t. λ , and set to 0:
$$\frac{\partial LL(\lambda)}{\partial \lambda} = -n + \frac{1}{\lambda} \sum_{i=1}^n X_i = 0 \Rightarrow \lambda_{MLE} = \frac{1}{n} \sum_{i=1}^n X_i$$

Maximizing Likelihood with Normal

- Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim N(\mu, \sigma^2)$

- PDF: $f(X_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(X_i - \mu)^2 / (2\sigma^2)}$

- Log-likelihood:

$$LL(\theta) = \sum_{i=1}^n \log\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-(X_i - \mu)^2 / (2\sigma^2)}\right) = \sum_{i=1}^n \left[-\log(\sqrt{2\pi}\sigma) - (X_i - \mu)^2 / (2\sigma^2) \right]$$

- First, differentiate w.r.t. μ , and set to 0:

$$\frac{\partial LL(\mu, \sigma^2)}{\partial \mu} = \sum_{i=1}^n 2(X_i - \mu) / (2\sigma^2) = \frac{1}{\sigma^2} \sum_{i=1}^n (X_i - \mu) = 0$$

- Then, differentiate w.r.t. σ , and set to 0:

$$\frac{\partial LL(\mu, \sigma^2)}{\partial \sigma} = \sum_{i=1}^n -\frac{1}{\sigma} + 2(X_i - \mu)^2 / (2\sigma^3) = -\frac{n}{\sigma} + \sum_{i=1}^n (X_i - \mu)^2 / (\sigma^3) = 0$$

Being Normal, Simultaneously

- Now have two equations, two unknowns:

$$\frac{1}{\sigma^2} \sum_{i=1}^n (X_i - \mu) = 0 \quad -\frac{n}{\sigma} + \sum_{i=1}^n (X_i - \mu)^2 / (\sigma^3) = 0$$

- First, solve for μ_{MLE} :

$$\frac{1}{\sigma^2} \sum_{i=1}^n (X_i - \mu) = 0 \Rightarrow \sum_{i=1}^n X_i = n\mu \Rightarrow \mu_{MLE} = \frac{1}{n} \sum_{i=1}^n X_i$$

- Then, solve for σ^2_{MLE} :

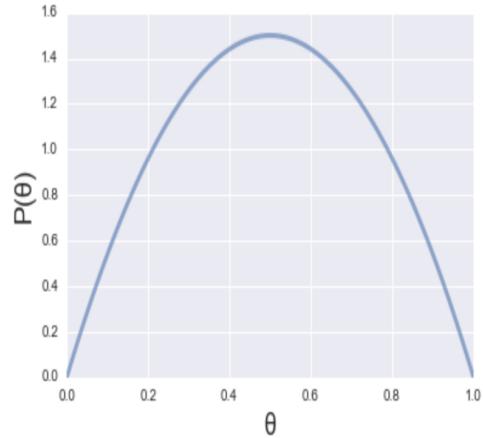
$$-\frac{n}{\sigma} + \sum_{i=1}^n (X_i - \mu)^2 / (\sigma^3) = 0 \Rightarrow n\sigma^2 = \sum_{i=1}^n (X_i - \mu)^2$$
$$\sigma^2_{MLE} = \frac{1}{n} \sum_{i=1}^n (X_i - \mu_{MLE})^2$$

Maximum A Posteriori Estimation

MLE is powerful when you have enough data. However, it doesn't work well when observed data size is small. For example, if Liverpool only had 2 matches and they won the 2 matches, then the estimated value of θ by MLE is $2/2 = 1$. It means that the estimation says Liverpool wins 100%, which is unrealistic estimation. MAP can help dealing with this issue.

Assume that we have a prior knowledge that Liverpool's winning percentage for the past few seasons were around 50%.

Then, without the data from this season, we already have somewhat idea of potential value of θ . Based (only) on the prior knowledge, the value of θ is most likely to be 0.5, and less likely to be 0 or 1. On the other words, the probability of $\theta=0.5$ is higher than $\theta=0$ or 1. Calling this as the **prior probability** $P(\theta)$, and if we visualise this, it would be like the figure.



Then, having the observed data D (*30 win out of 38 matches*) from this season, we can update this $P(\theta)$ which is based only on the prior knowledge. The updated probability of θ given D is expressed as $P(\theta|D)$ and called the **posterior probability**.

Now, we want to know the best guess of θ considering both our prior knowledge and the observed data. It means maximising $P(\theta|D)$ and it's the MAP estimation.

$$\operatorname{argmax}_{\theta} P(\theta|D)$$

The question here is, **how to calculate $P(\theta|D)$?** So far in this article, we checked the way to calculate $P(D|\theta)$ but haven't seen the way to calculate $P(\theta|D)$. To do so, we need to use Bayes' theorem below.

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

we can calculate the **posterior probability** $P(\theta|D)$ using the **likelihood** $P(D|\theta)$ and the **prior probability** $P(\theta)$.

There's $P(D)$ in the equation, but $P(D)$ is independent to the value of θ . Since we're only interested in finding θ maximising $P(\theta|D)$, we can ignore $P(D)$ in our maximisation.

$$\operatorname{argmax}_{\theta} P(\theta|D) = \operatorname{argmax}_{\theta} P(D|\theta)P(\theta)$$

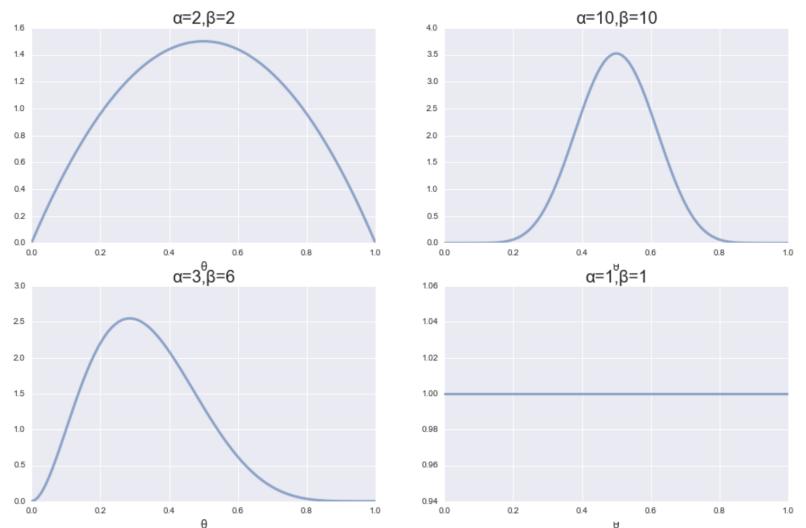
The equation above means that the maximisation of the posterior probability $P(\theta|D)$ with respect to θ is equal to the maximisation of the product of Likelihood $P(D|\theta)$ and Prior probability $P(\theta)$ with respect to θ .

Intrinsically, we can use any formulas describing probability distribution as $P(\theta)$ to express our prior knowledge well. However, for the computational simplicity, specific probability distributions are used corresponding to the probability distribution of likelihood. It's called **conjugate prior distribution**.

In this example, the likelihood $P(D|\theta)$ follows binomial distribution. Since the conjugate prior of binomial distribution is Beta distribution, we use Beta distribution to express $P(\theta)$ here. Beta distribution is described as below.

$$P(\theta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1} (1 - \theta)^{\beta-1}$$

where, α and β are called **hyperparameter**, which cannot be determined by data. Rather we set them subjectively to express our prior knowledge well. For example, graphs next are some visualisation of Beta distribution with different values of α and β . You can see the top left graph is the one we used in the example above (expressing that $\theta=0.5$ is the most likely value based on the prior knowledge), and the top right graph is also expressing the same prior knowledge but this one is for the believer that past seasons' results are reflecting Liverpool's true capability very well.



A note here about the bottom right graph: when $\alpha=1$ and $\beta=1$, it means we don't have any prior knowledge about θ . In this case the estimation will be completely same as the one by MLE.

So, by now we have all the components to calculate $P(D|\theta)P(\theta)$ to maximise.

$$\begin{aligned} P(D|\theta)P(\theta) &= \binom{n}{k} \theta^k (1-\theta)^{n-k} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1} \\ &= \binom{n}{k} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{k+\alpha-1} (1-\theta)^{n-k+\beta-1} \end{aligned}$$

As same as MLE, we can get θ maximising this by having derivative of the this function with respect to θ , and setting it to zero

$$\begin{aligned} \frac{dP(D|\theta)P(\theta)}{d\theta} &= \binom{n}{k} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} ((k+\alpha-1)\theta^{k+\alpha-2}(1-\theta)^{n-k+\beta-1} - (n-k+\beta-1)\theta^{k+\alpha-1}(1-\theta)^{n-k+\beta-2}) \\ &= \binom{n}{k} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{k+\alpha-2}(1-\theta)^{n-k+\beta-2} ((k+\alpha-1)(1-\theta) - (n-k+\beta-1)\theta) \\ &= 0 \end{aligned}$$

By solving this, we obtain following

$$\theta = \frac{k+\alpha-1}{n+\alpha+\beta-2}$$

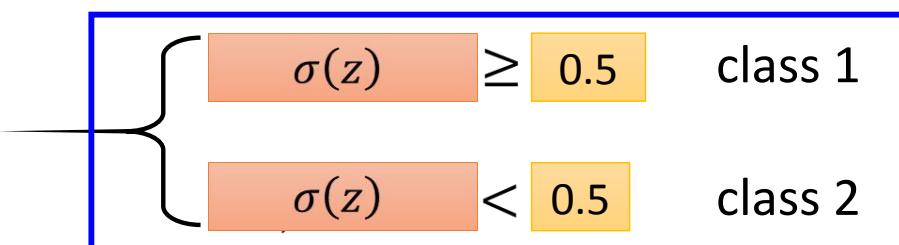
In this example, assuming we use $\alpha=10$ and $\beta=10$, then $\theta=(30+10-1)/(38+10+10-2) = 39/56 = 69.6\%$

As seen in the example above, the ideas behind the complicated mathematical equations of MLE and MAP are surprisingly simple. I used a binomial distribution as an example in the slides, but MLE and MAP are applicable to other statistical models as well.

Classification: Logistic Regression

Step 1: Function Set

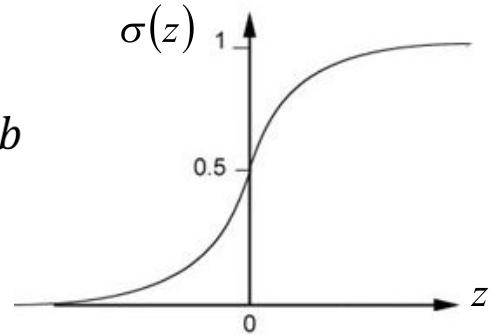
Function set: Including all different w and b



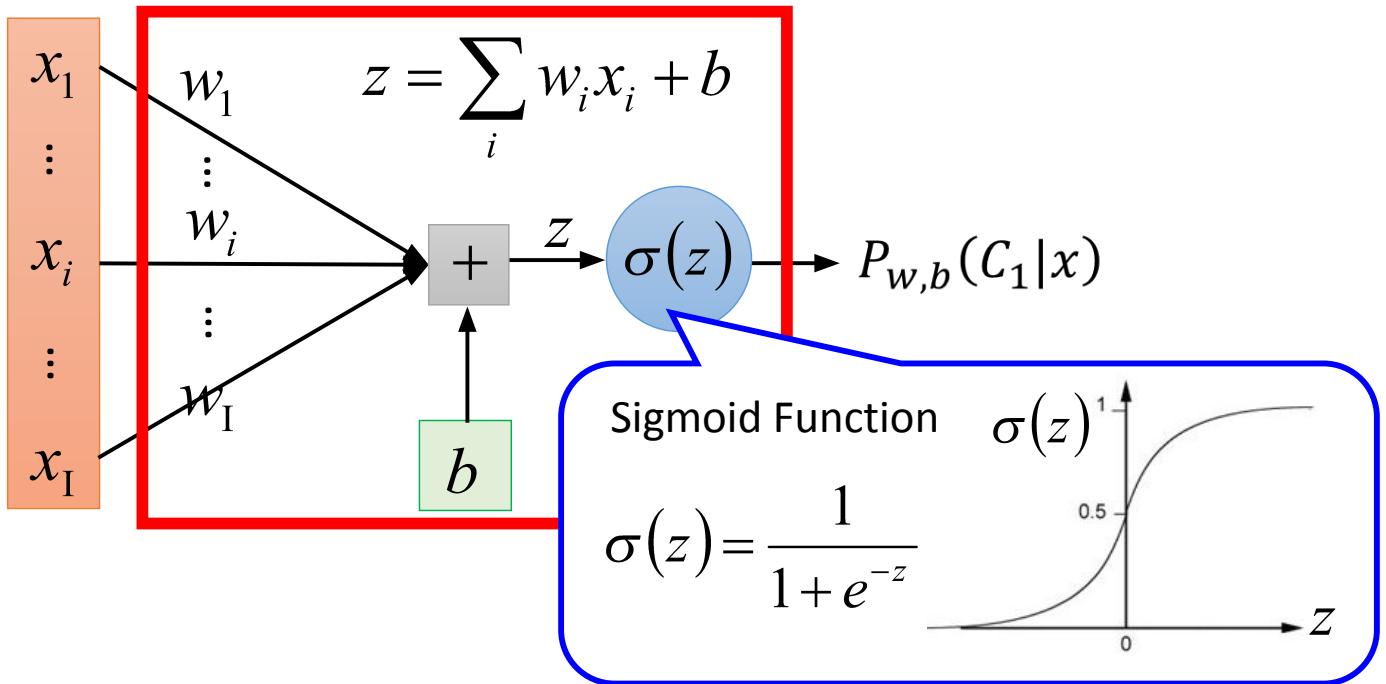
$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



Step 1: Function Set



Step 2: Goodness of a Function

Training Data	x^1	x^2	x^3	x^N
	C_1	C_1	C_2	C_1

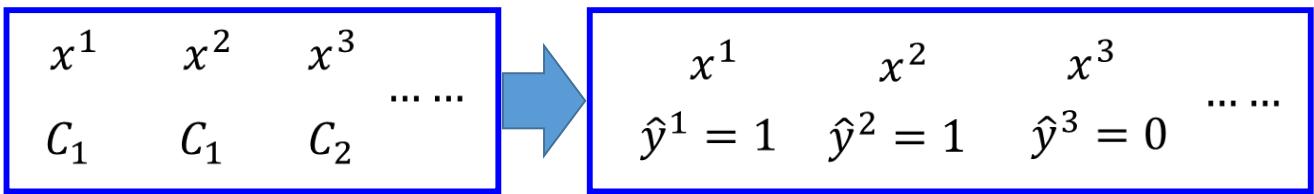
Assume the data is generated based on $f_{w,b}(x) = P_{w,b}(C_1|x)$

Given a set of w and b , what is its probability of generating the data?

$$L(w, b) = f_{w,b}(x^1) f_{w,b}(x^2) (1 - f_{w,b}(x^3)) \cdots f_{w,b}(x^N)$$

The most likely w^* and b^* is the one with the largest $L(w, b)$.

$$w^*, b^* = \arg \max_{w,b} L(w, b)$$



\hat{y}^n : 1 for class 1, 0 for class 2

$$L(w, b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots$$

$$w^*, b^* = \arg \max_{w,b} L(w, b) \quad = \quad w^*, b^* = \arg \min_{w,b} -\ln L(w, b)$$

$$\begin{aligned} & -\ln L(w, b) \\ &= -\ln f_{w,b}(x^1) \rightarrow -[1 \ln f(x^1) + 0 \ln(1 - f(x^1))] \\ & -\ln f_{w,b}(x^2) \rightarrow -[1 \ln f(x^2) + 0 \ln(1 - f(x^2))] \\ & -\ln(1 - f_{w,b}(x^3)) \rightarrow -[0 \ln f(x^3) + 1 \ln(1 - f(x^3))] \\ & \vdots \end{aligned}$$

Step 2: Goodness of a Function

$$L(w, b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots f_{w,b}(x^N)$$

$$-\ln L(w, b) = \ln f_{w,b}(x^1) + \ln f_{w,b}(x^2) + \ln\left(1 - f_{w,b}(x^3)\right)\cdots$$

\hat{y}^n : 1 for class 1, 0 for class 2

$$= \sum_n -[\hat{y}^n \ln f_{w,b}(x^n) + (1 - \hat{y}^n) \ln(1 - f_{w,b}(x^n))]$$

Cross entropy between two Bernoulli

Distribution p:

$$p(x = 1) = \hat{y}^n$$

$$p(x = 0) = 1 - \hat{y}^n$$

Distribution q:

$$q(x = 1) = f(x^n)$$

$$q(x = 0) = 1 - f(x^n)$$

$$H(p, q) = -\sum_x p(x) \ln(q(x))$$

Step 2: Goodness of a Function

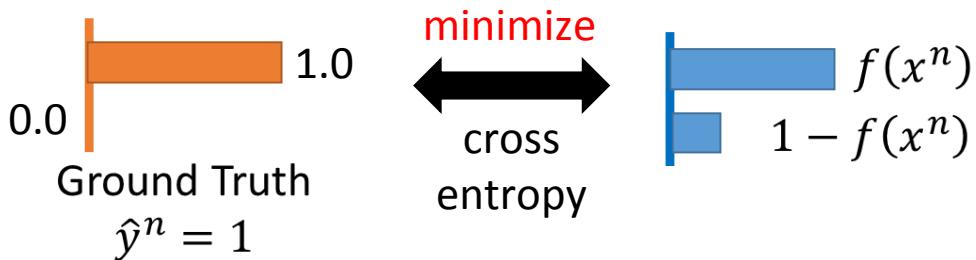
$$L(w, b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots f_{w,b}(x^N)$$

$$-\ln L(w, b) = \ln f_{w,b}(x^1) + \ln f_{w,b}(x^2) + \ln \left(1 - f_{w,b}(x^3)\right)\cdots$$

\hat{y}^n : 1 for class 1, 0 for class 2

$$= \sum_n -\left[\hat{y}^n \ln f_{w,b}(x^n) + (1 - \hat{y}^n) \ln \left(1 - f_{w,b}(x^n)\right) \right]$$

Cross entropy between two Bernoulli distribution

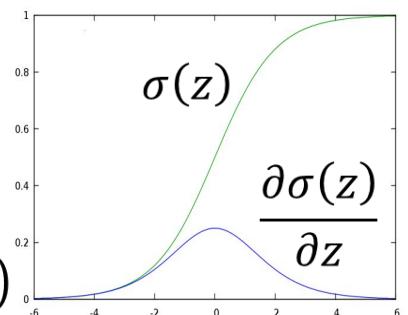


Step 3: Find the best function

$$\frac{\partial \ln L(w, b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \frac{\ln f_{w,b}(x^n)}{\partial w_i} + (1 - \hat{y}^n) \frac{\ln \left(1 - f_{w,b}(x^n)\right)}{\partial w_i} \right]$$

$$\frac{\partial \ln f_{w,b}(x)}{\partial w_i} = \frac{\partial \ln f_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i} \quad \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial \ln \sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \frac{\partial \sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \cancel{\sigma(z)(1 - \sigma(z))}$$



$$\begin{aligned} f_{w,b}(x) &= \sigma(z) \\ &= 1 / (1 + \exp(-z)) \end{aligned}$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

$$\frac{\partial \ln L(w, b)}{\partial w_i} = \sum_n - \left[\hat{y}^n \frac{\ln f_{w,b}(x^n)}{\partial w_i} + (1 - \hat{y}^n) \frac{\ln (1 - f_{w,b}(x^n))}{\partial w_i} \right]$$

$$\frac{\partial \ln (1 - f_{w,b}(x))}{\partial w_i} = \frac{\partial \ln (1 - f_{w,b}(x))}{\partial z} \frac{\partial z}{\partial w_i} \quad \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial \ln (1 - \sigma(z))}{\partial z} = -\frac{1}{1 - \sigma(z)} \frac{\partial \sigma(z)}{\partial z} = -\frac{1}{1 - \sigma(z)} \sigma(z)(1 - \sigma(z))$$

$$f_{w,b}(x) = \sigma(z) \\ = 1/(1 + \exp(-z)) \quad z = w \cdot x + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

$$\frac{\partial \ln L(w, b)}{\partial w_i} = \sum_n - \left[\hat{y}^n \frac{\ln f_{w,b}(x^n)}{\partial w_i} + (1 - \hat{y}^n) \frac{\ln (1 - f_{w,b}(x^n))}{\partial w_i} \right]$$

$$= \sum_n - \left[\hat{y}^n \underbrace{(1 - f_{w,b}(x^n))}_{\textcolor{blue}{\underline{x}_i^n}} x_i^n - (1 - \hat{y}^n) \underbrace{f_{w,b}(x^n)}_{\textcolor{blue}{\underline{x}_i^n}} x_i^n \right]$$

$$= \sum_n - [\hat{y}^n - \hat{y}^n \cancel{f_{w,b}(x^n)} - f_{w,b}(x^n) + \cancel{\hat{y}^n f_{w,b}(x^n)}] \textcolor{blue}{\underline{x}_i^n}$$

$$= \sum_n - (\hat{y}^n - f_{w,b}(x^n)) x_i^n \quad \text{Larger difference, larger update}$$

$$w_i \leftarrow w_i - \eta \sum_n - (\hat{y}^n - f_{w,b}(x^n)) x_i^n$$

Logistic Regression + Square Error

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Step 2: Training data: (x^n, \hat{y}^n) , \hat{y}^n : 1 for class 1, 0 for class 2

$$L(f) = \frac{1}{2} \sum_n (f_{w,b}(x^n) - \hat{y}^n)^2$$

Step 3:

$$\frac{\partial (f_{w,b}(x) - \hat{y})^2}{\partial w_i} = 2(f_{w,b}(x) - \hat{y}) \frac{\partial f_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i}$$

$$= 2(f_{w,b}(x) - \hat{y}) f_{w,b}(x) (1 - f_{w,b}(x)) x_i$$

$\hat{y}^n = 1$ If $f_{w,b}(x^n) = 1$ (close to target) $\rightarrow \partial L / \partial w_i = 0$

If $f_{w,b}(x^n) = 0$ (far from target) $\rightarrow \partial L / \partial w_i = 0$

Logistic Regression + Square Error

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Step 2: Training data: (x^n, \hat{y}^n) , \hat{y}^n : 1 for class 1, 0 for class 2

$$L(f) = \frac{1}{2} \sum_n (f_{w,b}(x^n) - \hat{y}^n)^2$$

Step 3:

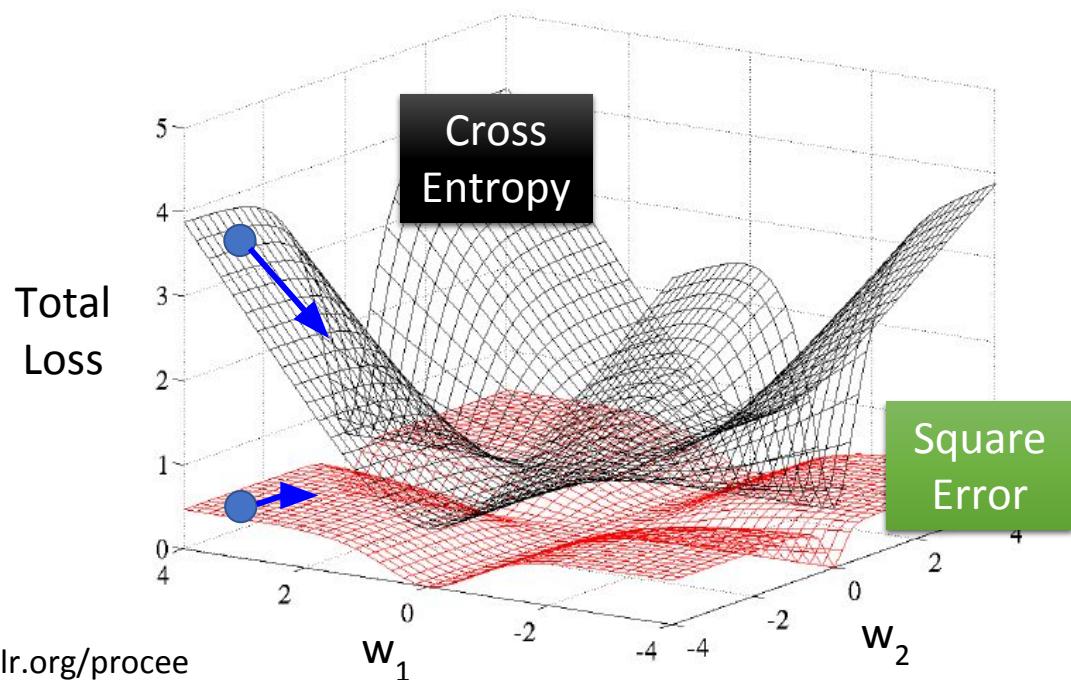
$$\frac{\partial (f_{w,b}(x) - \hat{y})^2}{\partial w_i} = 2(f_{w,b}(x) - \hat{y}) \frac{\partial f_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i}$$

$$= 2(f_{w,b}(x) - \hat{y}) f_{w,b}(x) (1 - f_{w,b}(x)) x_i$$

$\hat{y}^n = 0$ If $f_{w,b}(x^n) = 1$ (far from target) $\rightarrow \partial L / \partial w_i = 0$

If $f_{w,b}(x^n) = 0$ (close to target) $\rightarrow \partial L / \partial w_i = 0$

Cross Entropy v.s. Square Error



<http://jmlr.org/proceedings/papers/v9/glorot10a.pdf>

Logistic Regression

$$\text{Step 1: } f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$$

Output: between 0 and 1

Linear Regression

$$f_{w,b}(x) = \sum_i w_i x_i + b$$

Output: any value

Step 2:

Step 3:

Logistic Regression

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Output: between 0 and 1

Linear Regression

$$f_{w,b}(x) = \sum_i w_i x_i + b$$

Output: any value

Training data: (x^n, \hat{y}^n)

Step 2: \hat{y}^n : 1 for class 1, 0 for class 2

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

Training data: (x^n, \hat{y}^n)

\hat{y}^n : a real number

$$L(f) = \frac{1}{2} \sum_n (f(x^n) - \hat{y}^n)^2$$

Cross entropy:

$$l(f(x^n), \hat{y}^n) = -[\hat{y}^n \ln f(x^n) + (1 - \hat{y}^n) \ln(1 - f(x^n))]$$

Logistic Regression

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Output: between 0 and 1

Linear Regression

$$f_{w,b}(x) = \sum_i w_i x_i + b$$

Output: any value

Training data: (x^n, \hat{y}^n)

Step 2: \hat{y}^n : 1 for class 1, 0 for class 2

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

Training data: (x^n, \hat{y}^n)

\hat{y}^n : a real number

$$L(f) = \frac{1}{2} \sum_n (f(x^n) - \hat{y}^n)^2$$

Step 3:
Logistic regression:

$$w_i \leftarrow w_i - \eta \sum_n -(\hat{y}^n - f_{w,b}(x^n)) x_i^n$$

Linear regression: $w_i \leftarrow w_i - \eta \sum_n -(\hat{y}^n - f_{w,b}(x^n)) x_i^n$

Discriminative v.s. Generative

$$P(C_1|x) = \sigma(w \cdot x + b)$$

directly find w and b

Find $\mu^1, \mu^2, \Sigma^{-1}$

Will we obtain the same set of w and b ?

$$w^T = (\mu^1 - \mu^2)^T \Sigma^{-1}$$

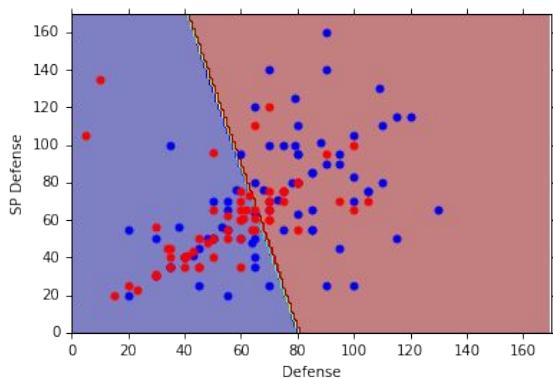
$$b = -\frac{1}{2}(\mu^1)^T (\Sigma^1)^{-1} \mu^1$$

$$+ \frac{1}{2}(\mu^2)^T (\Sigma^2)^{-1} \mu^2 + \ln \frac{N_1}{N_2}$$

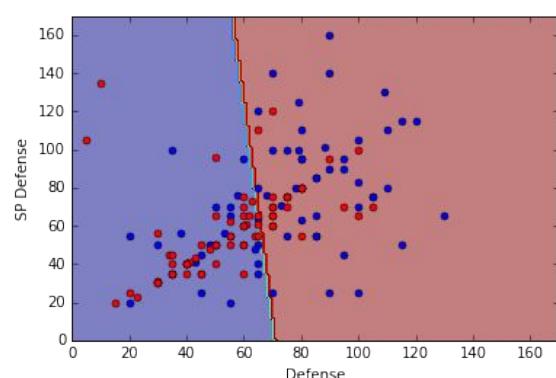
The same model (function set), but different function may be selected by the same training data.

Generative v.s. Discriminative

Generative



Discriminative



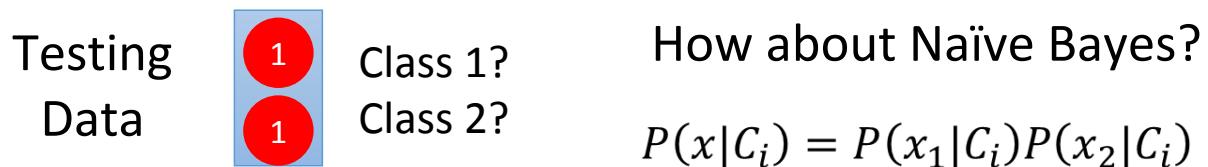
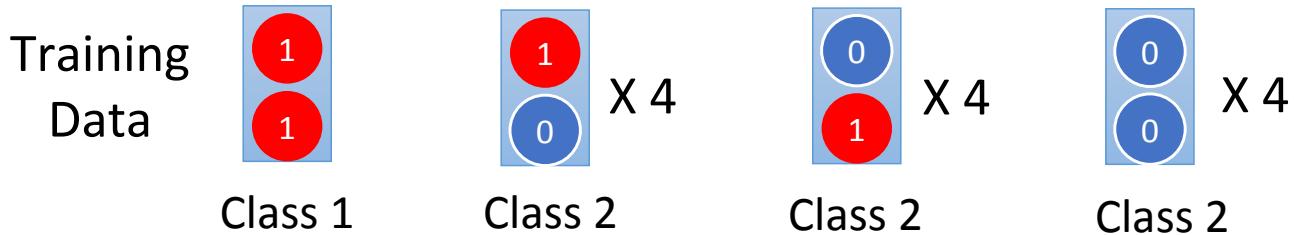
All: hp, att, sp att, de, sp de, speed

73% accuracy

79% accuracy

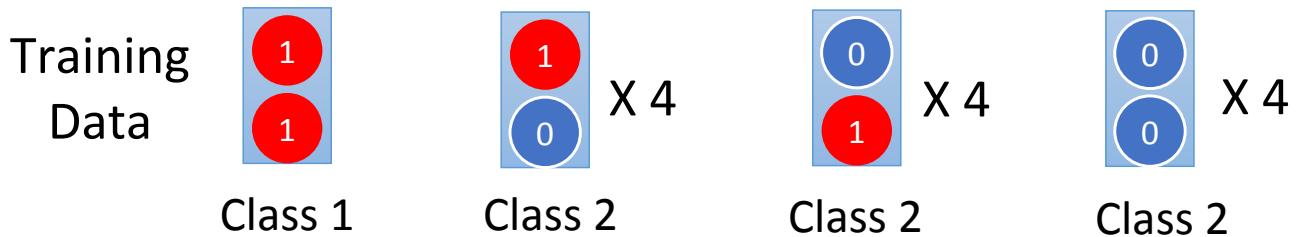
Generative v.s. Discriminative

- Example



Generative v.s. Discriminative

- Example



$$P(C_1) = \frac{1}{13} \quad P(x_1 = 1|C_1) = 1 \quad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \quad P(x_1 = 1|C_2) = \frac{1}{3} \quad P(x_2 = 1|C_2) = \frac{1}{3}$$

Training Data		X 4	X 4	X 4
	Class 1	Class 2	Class 2	Class 2
Testing Data		$P(C_1 x) = \frac{P(x C_1)P(C_1)}{P(x C_1)P(C_1) + P(x C_2)P(C_2)}$ <0.5	$\frac{1 \times 1}{13}$	$\frac{1}{13}$

$$P(C_1) = \frac{1}{13} \quad P(x_1 = 1|C_1) = 1 \quad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \quad P(x_1 = 1|C_2) = \frac{1}{3} \quad P(x_2 = 1|C_2) = \frac{1}{3}$$

Generative v.s. Discriminative

- Usually people believe discriminative model is better
- Benefit of generative model
 - With the assumption of probability distribution
 - less training data is needed
 - more robust to the noise
 - Priors and class-dependent probabilities can be estimated from different sources.

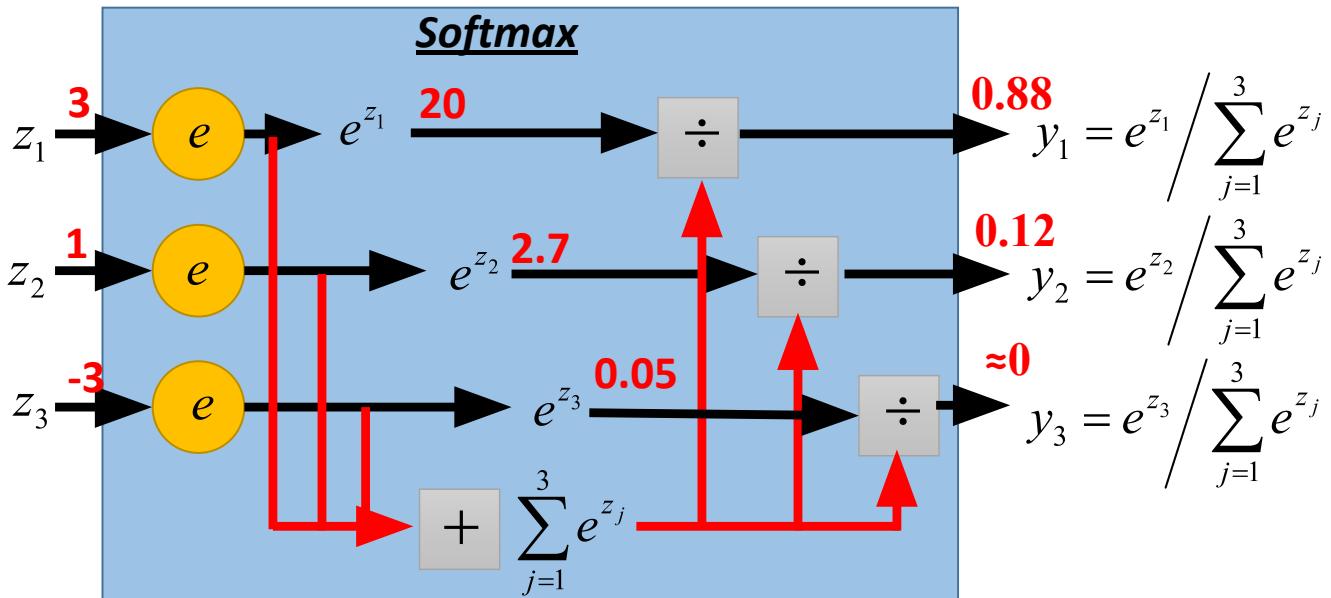
Multi-class Classification (3 classes as example)

$$\begin{aligned} C_1: w^1, b_1 \quad z_1 &= w^1 \cdot x + b_1 \\ C_2: w^2, b_2 \quad z_2 &= w^2 \cdot x + b_2 \\ C_3: w^3, b_3 \quad z_3 &= w^3 \cdot x + b_3 \end{aligned}$$

Probability:

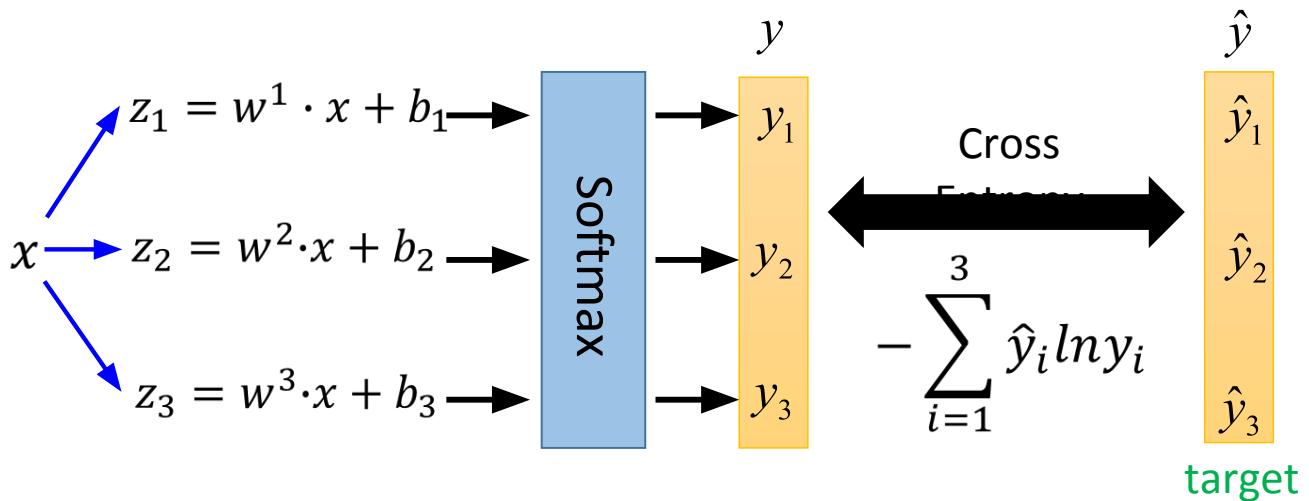
- $1 > y_i > 0$
- $\sum_i y_i = 1$

$$y_i = P(C_i | x)$$



Multi-class Classification (3 classes as example)

[Bishop, P209-210]



If $x \in$ class 1

$$\hat{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$- \ln y_1$$

If $x \in$ class 2

$$\hat{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

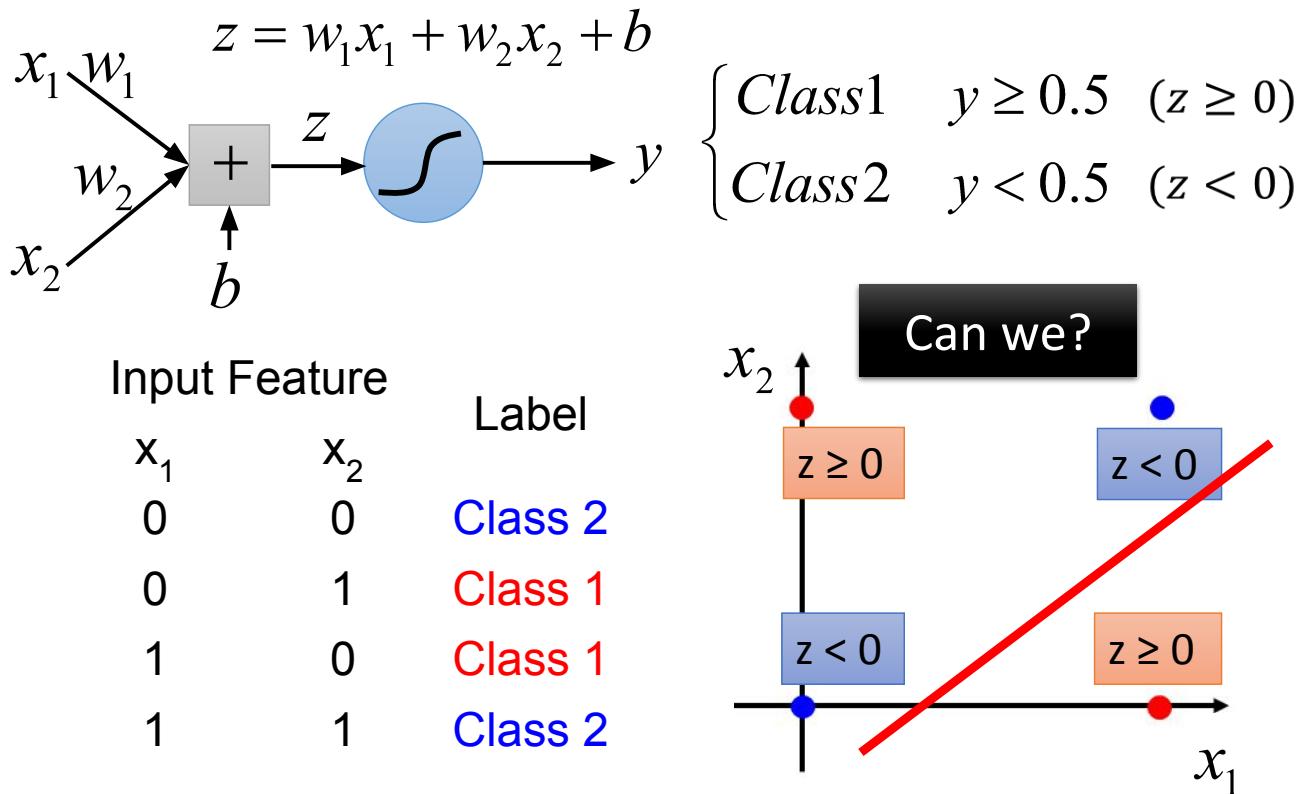
$$- \ln y_2$$

If $x \in$ class 3

$$\hat{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

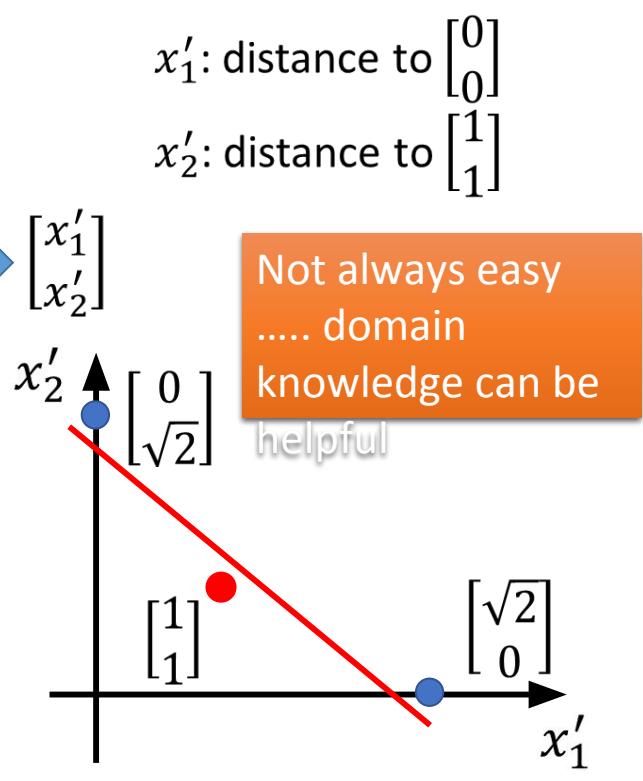
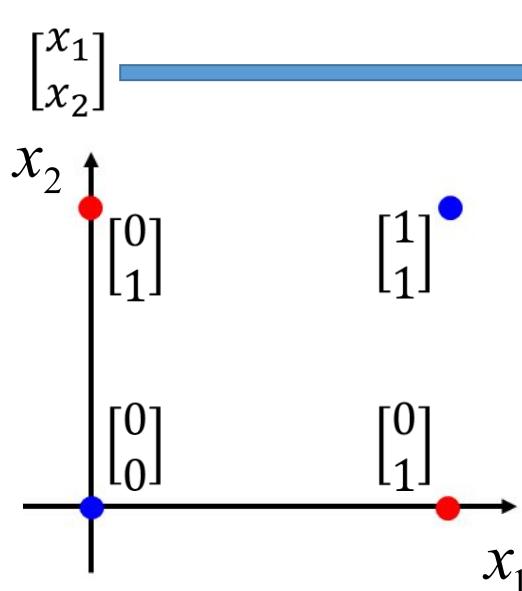
$$- \ln y_3$$

Limitation of Logistic Regression



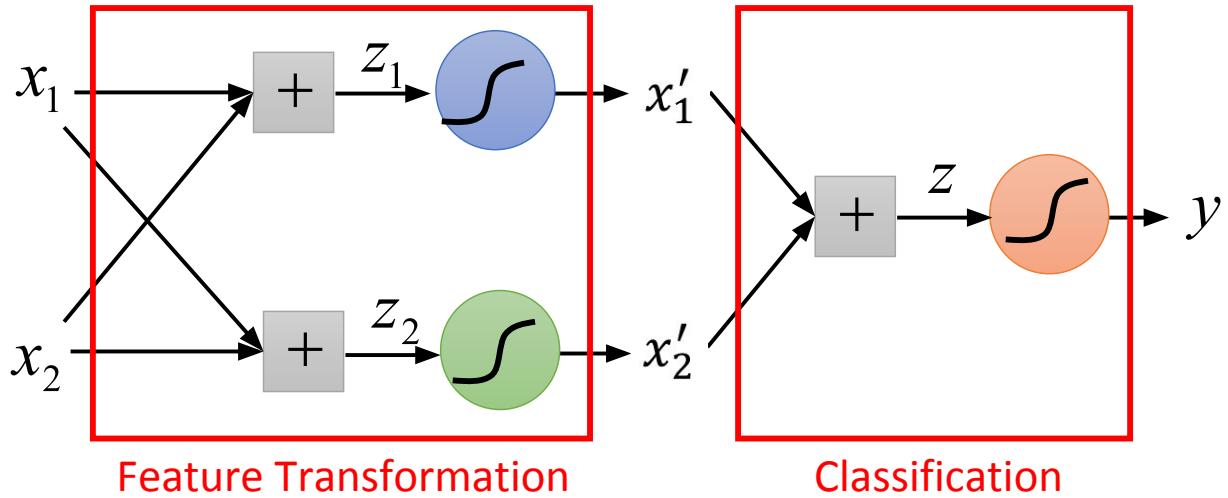
Limitation of Logistic Regression

- Feature transformation**

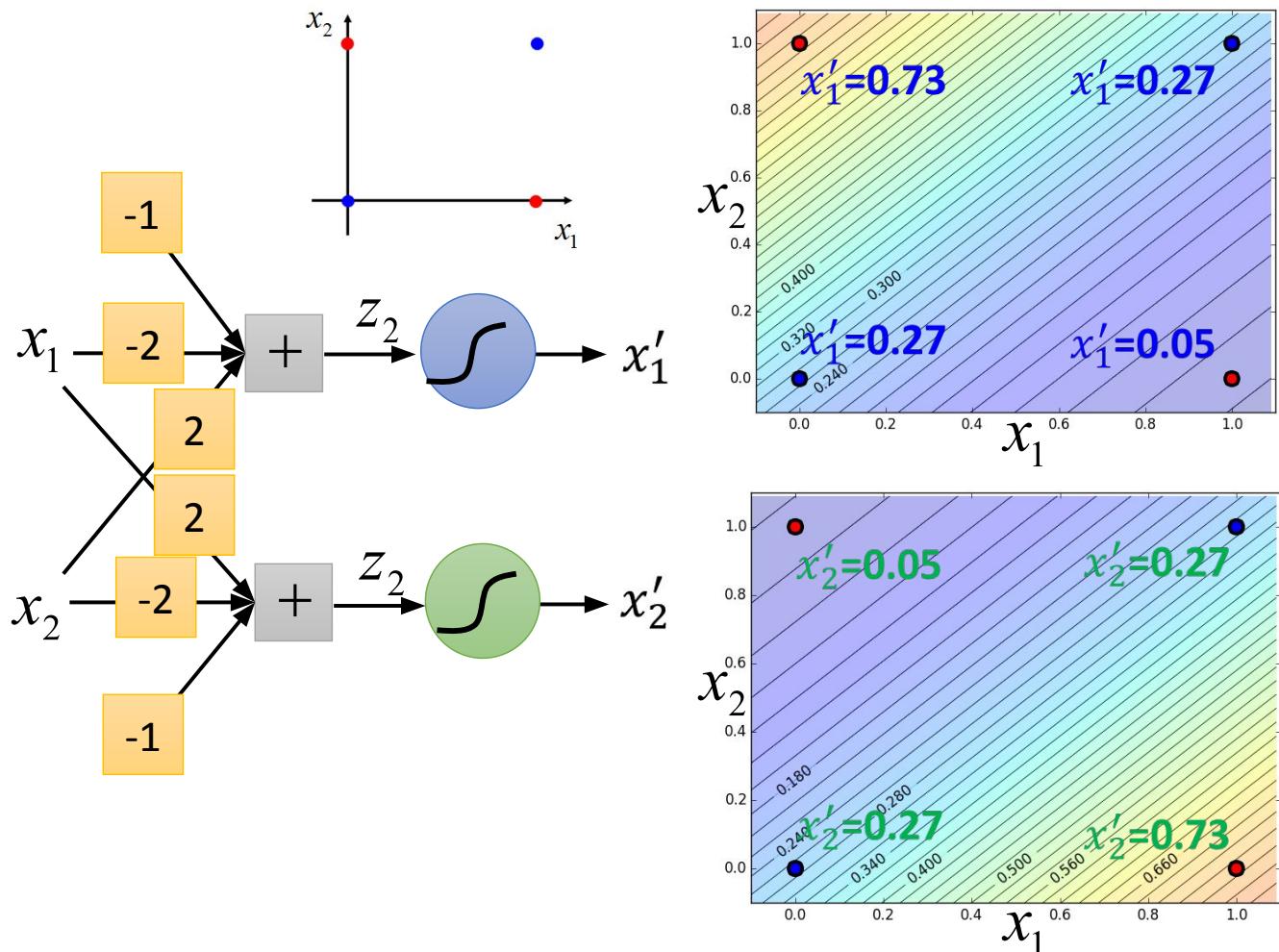


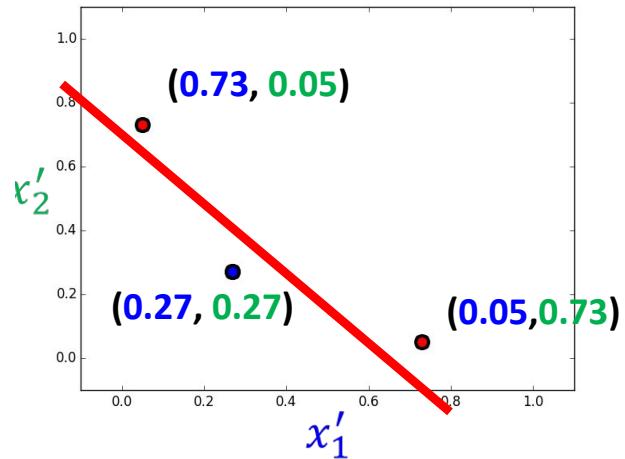
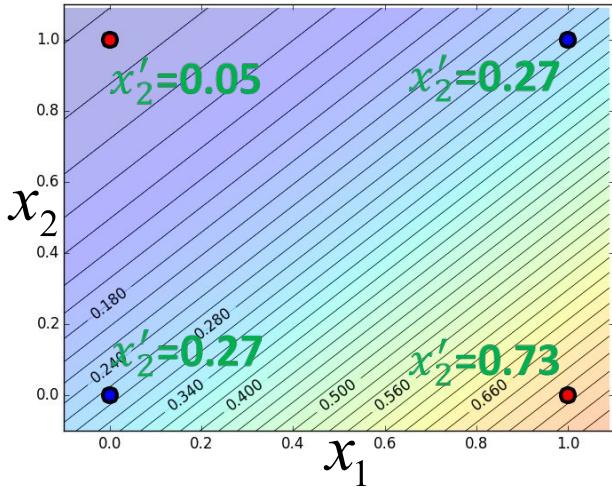
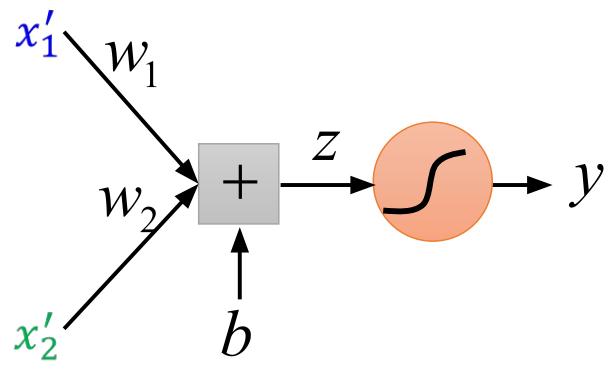
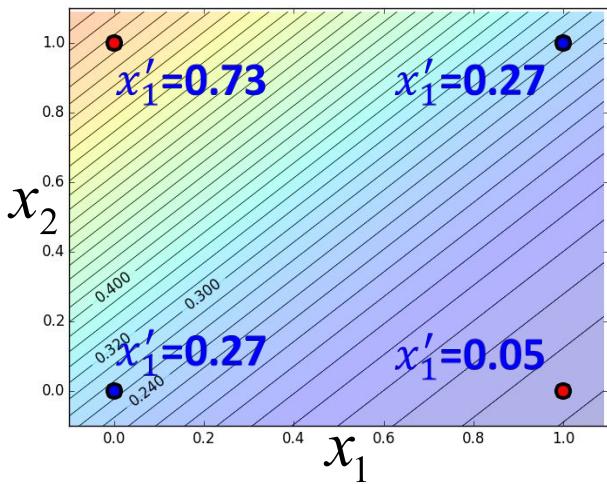
Limitation of Logistic Regression

- Cascading logistic regression models



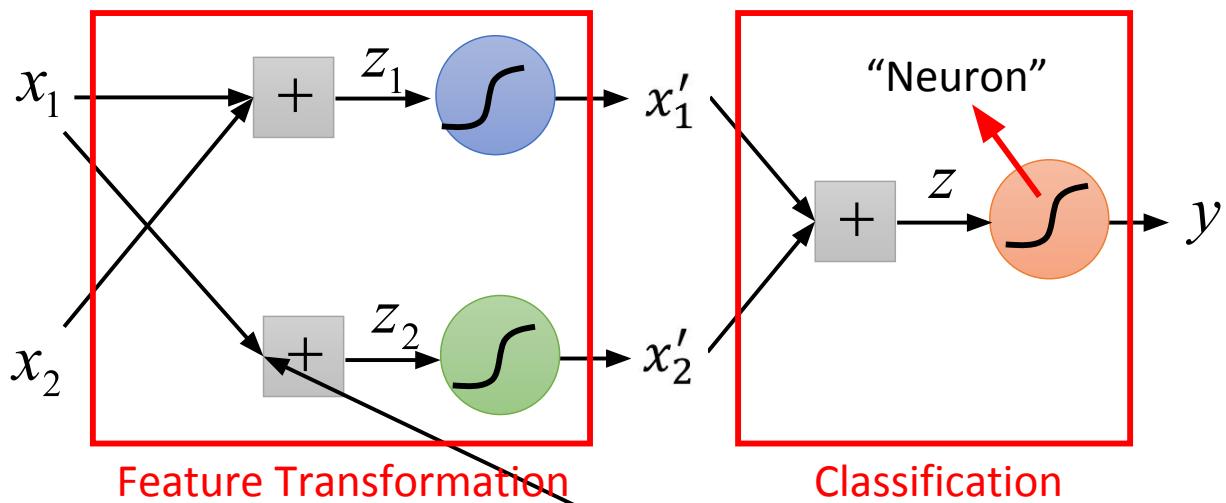
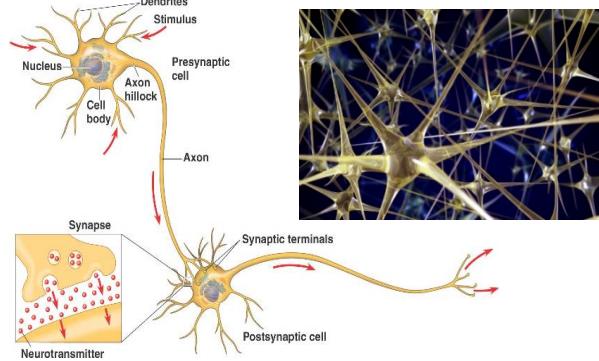
(ignore bias in this figure)





Deep Learning!

All the parameters of the logistic regressions are jointly learned.



Neural Network

Reference

- Bishop: Chapter 4.3

Ensemble Classifiers: Random Forests and Adaboost

By

Swagatam Das

**Electronics and Communication Sciences Unit,
Indian Statistical Institute, Kolkata – 700 108, India.**

E-mail:swagatam.das@isical.ac.in

Learning Is Impossible

What's my rule?

- 1 2 3 \Rightarrow satisfies rule
- 4 5 6 \Rightarrow satisfies rule
- 6 7 8 \Rightarrow satisfies rule
- 9 2 31 \Rightarrow does not satisfy rule

Possible rules

- 3 consecutive single digits
- 3 consecutive integers
- 3 numbers in ascending order
- 3 numbers whose sum is less than 25
- 3 numbers < 10
- 1, 4, or 6 in first column
- “yes” to first 3 sequences, “no” to all others



“What’s My Rule” For Machine Learning

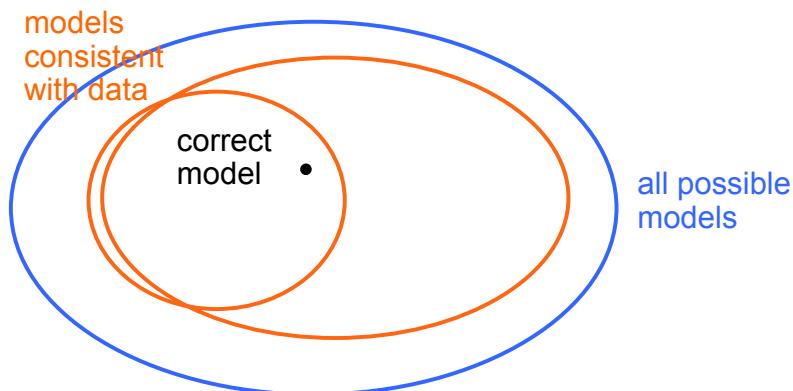
x1	x2	x3	y
0	0	0	1
0	1	1	0
1	0	0	0
1	1	1	1

1	1	0	?
---	---	---	---

16 possible rules (models)

With n binary inputs and p training examples, there are 2^{2^n-p} possible models.

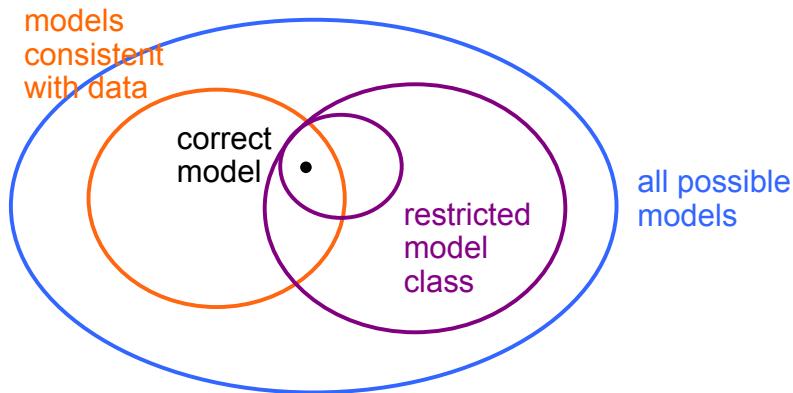
Model Space



More data helps

- In the limit of infinite data, look up table model is fine

Model Space



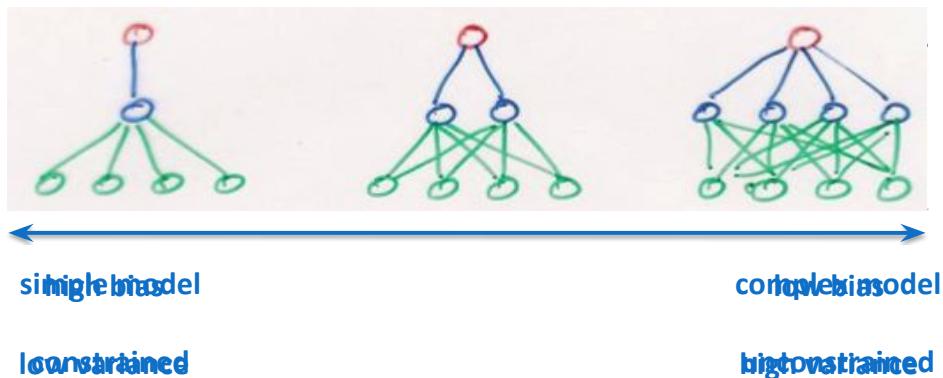
Restricting model class can help

Or it can hurt

Depends on whether restrictions are domain appropriate

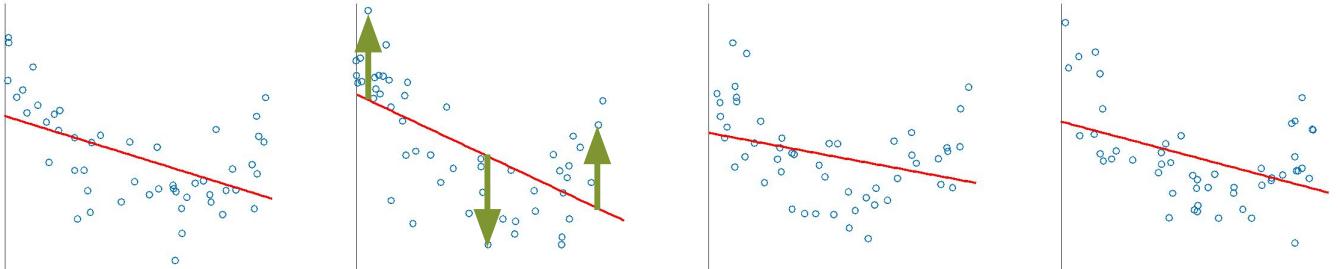
Restricting Models

Models range in their flexibility to fit arbitrary data



Bias

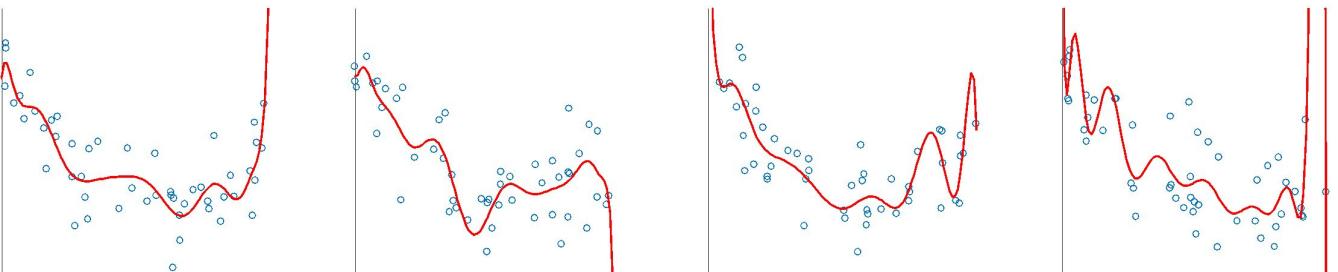
Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.



Regardless of training sample, or size of training sample, model will produce consistent errors

Variance

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.



Different samples of training data yield different model fits

Let the variable we are trying to predict as Y and other covariates as X. We assume there is a relationship between the two such that

$$Y = f(X) + \varepsilon,$$

where ε is the error term and it's normally distributed with a mean of 0. We will make a model $\hat{f}(X)$ of $f(X)$ using linear regression/classifier or any other modeling technique. So the expected squared error at a point x is

$$Err(x) = E[(Y - \hat{f}(x))^2]$$

The $Err(x)$ can be further decomposed as

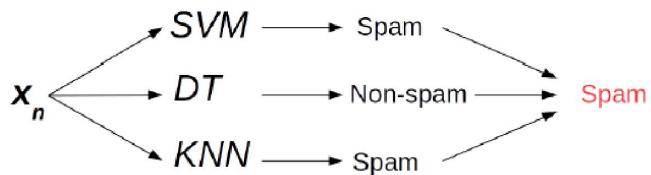
$$Err(x) = \left(E[\hat{f}(x)] - f(x) \right)^2 + E \left[\left(\hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

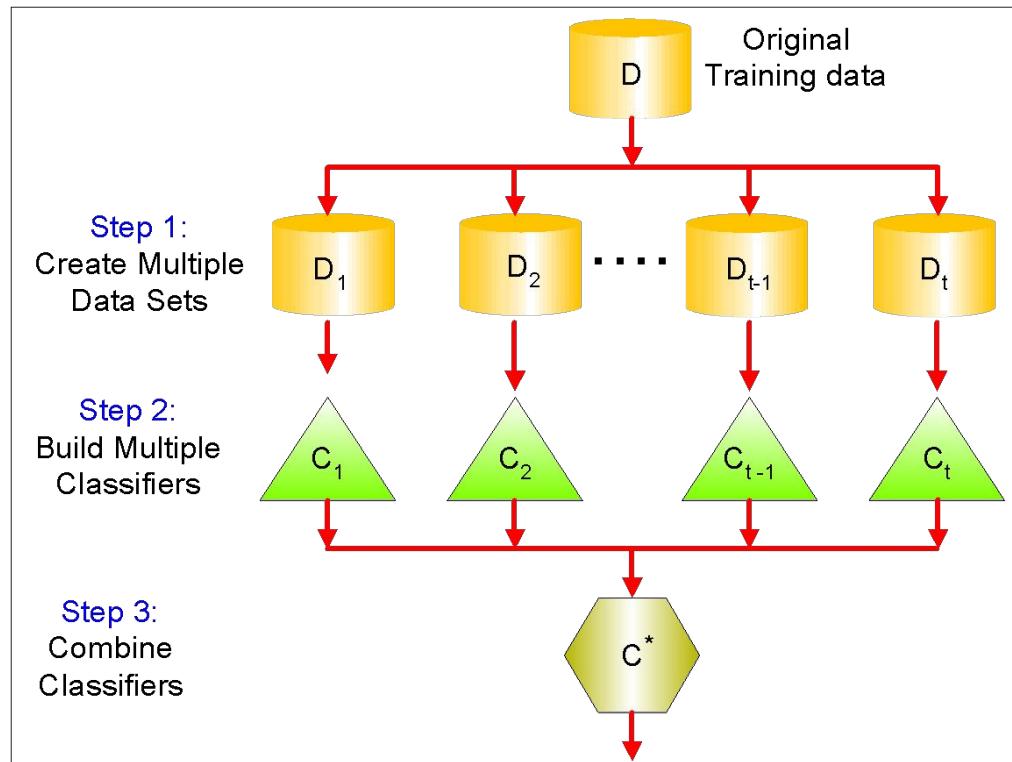
Irreducible error is the error that can't be reduced by creating good models. It is a measure of the amount of noise in our data. Here it is important to understand that no matter how good we make our model, our data will have certain amount of noise or irreducible error that can not be removed

What can be a simple Ensemble in Machine Learning?

- Voting or Averaging of predictions of multiple pre-trained models



Ensemble



Applications: classification, clustering, collaborative filtering, anomaly detection.....

11

<http://ews.uiuc.edu/~jinggao3/sdm10ensemble.htm>

Motivations

- **Motivations of ensemble methods**
 - Ensemble model improves accuracy and robustness over single model methods
 - Applications:
 - distributed computing
 - privacy-preserving applications
 - large-scale data with reusable models
 - multiple sources of data
 - Efficiency: a complex problem can be decomposed into multiple sub-problems that are easier to understand and solve (divide-and-conquer approach)

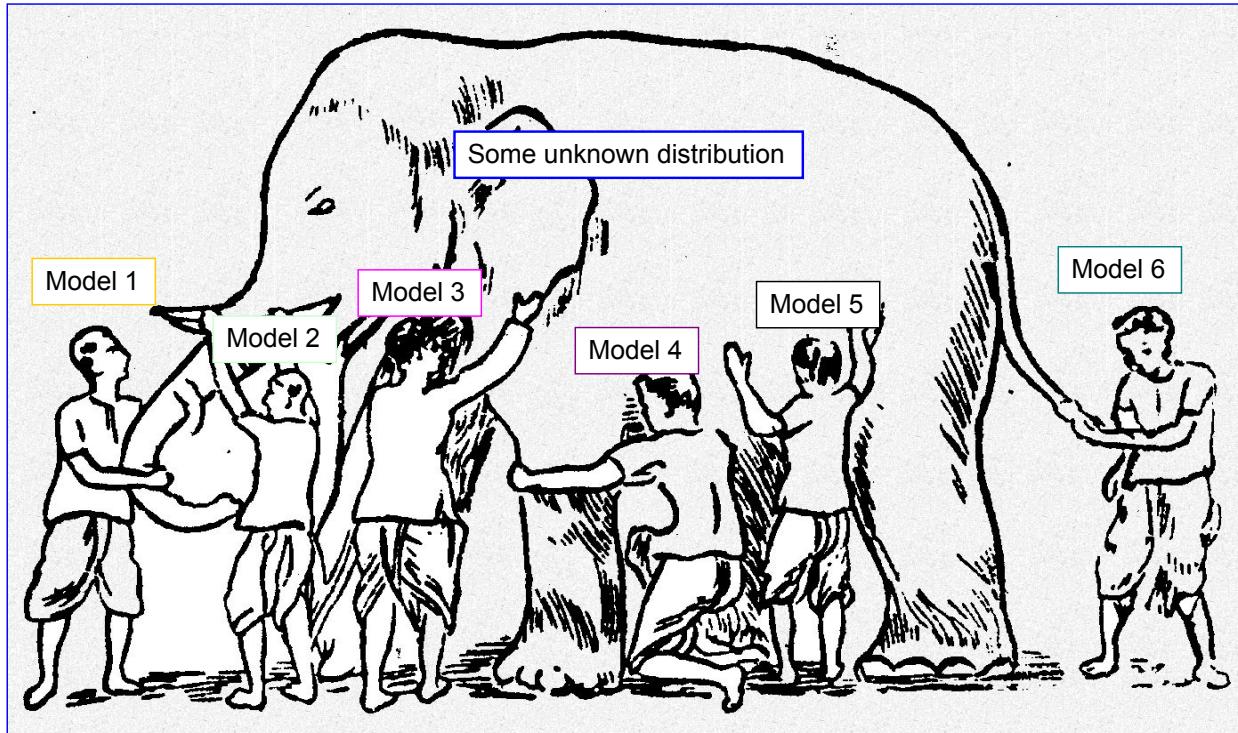
Why do ensembles work?

- Suppose there are 25 base classifiers
 - Each classifier has error rate, $\varepsilon = 0.35$
 - Assume classifiers are independent
 - i.e., probability that a classifier makes a mistake does not depend on whether other classifiers made a mistake
 - Note: in practice they are not independent!
- Probability that the ensemble classifier makes a wrong prediction
 - The ensemble makes a wrong prediction if the majority of the classifiers makes a wrong prediction
 - The probability that 13 or more classifiers err is

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} \approx 0.06 \ll \varepsilon$$

from T. Holloway, Introduction to Ensemble Learning, 2007.

Why Ensemble Works? (2)

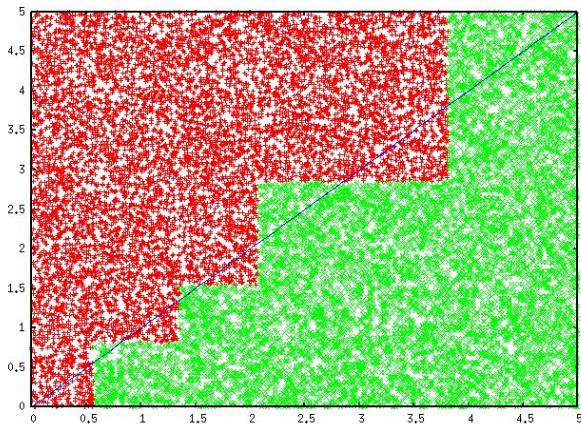


Ensemble gives the global picture!

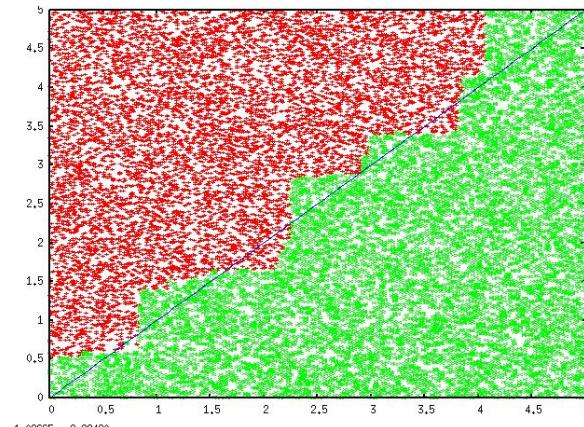
Why Ensemble Works?

- Overcome limitations of single hypothesis

- The target function may not be implementable with individual classifiers, but may be approximated by model averaging

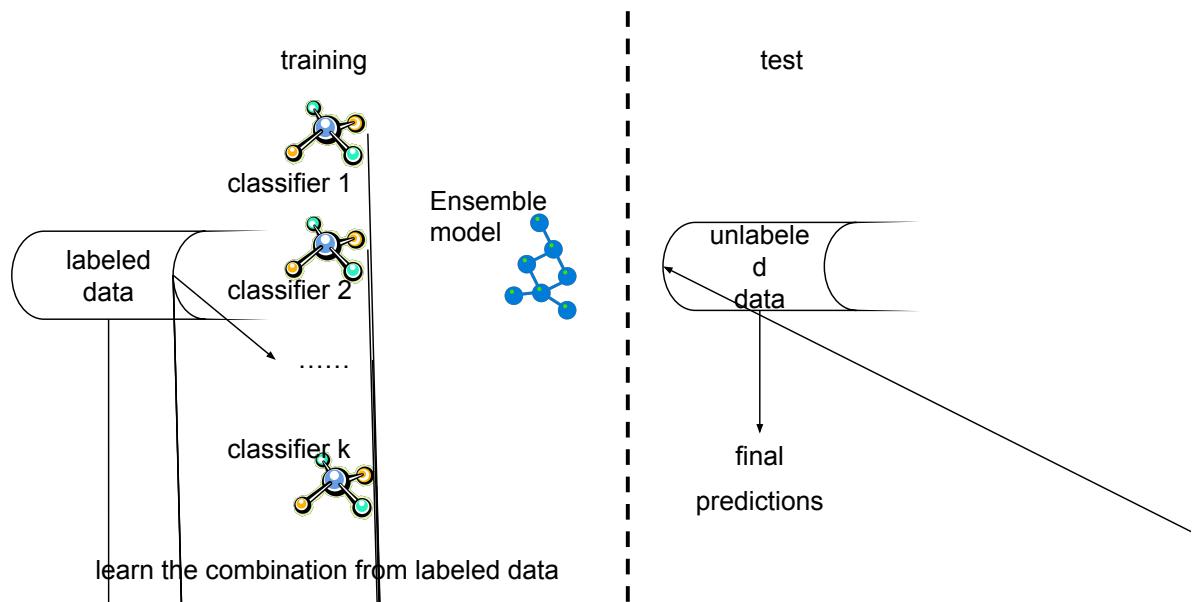


Decision Tree



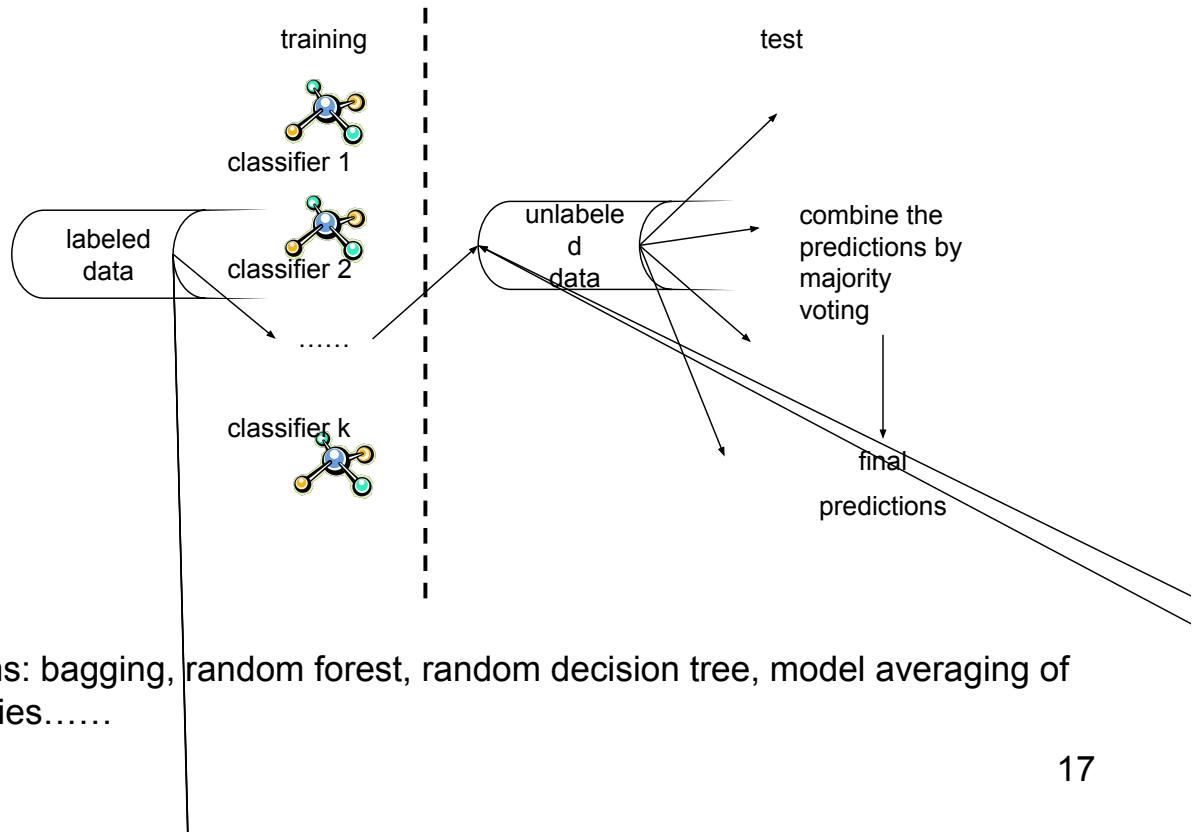
Model Averaging

Ensemble of Classifiers—Learn to Combine



Algorithms: boosting, stacked generalization, rule ensemble, Bayesian model averaging.....

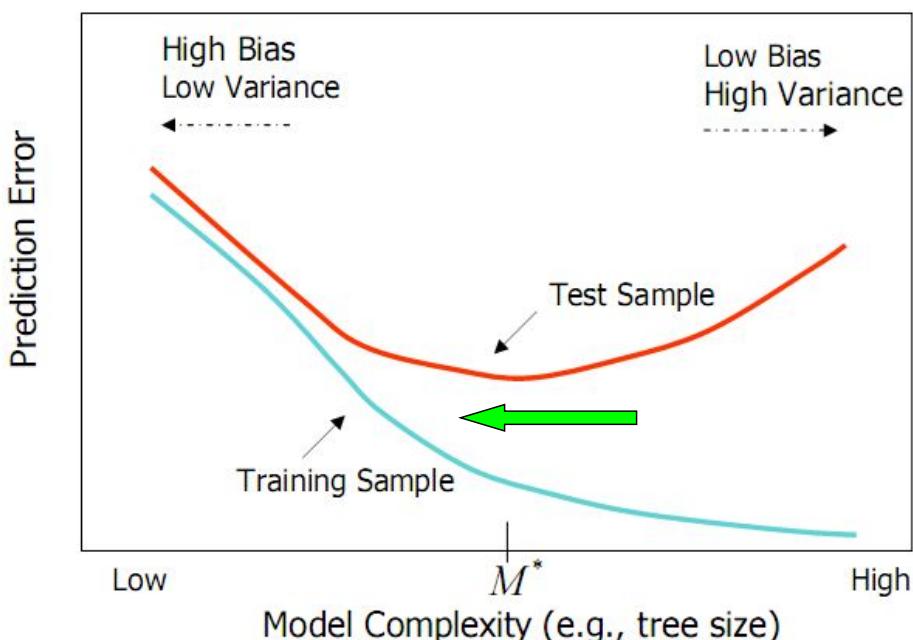
Ensemble of Classifiers—Consensus



17

Bias and Variance

- Ensemble methods
 - Combine learners to reduce variance



Generating Base Classifiers

- Sampling training examples
 - Train k classifiers on k subsets drawn from the training set
- Using different learning models
 - Use all the training examples, but apply different learning algorithms
- Sampling features
 - Train k classifiers on k subsets of features drawn from the feature space
- Learning “randomly”
 - Introduce randomness into learning procedures

19

Bagging

Bagging or [bootstrap aggregation](#) averages a given procedure over many samples, to reduce its variance — a poor man’s Bayes. See



pp 246.

Suppose $C(S, x)$ is a classifier, such as a tree, based on our training data S , producing a predicted class label at input point x .

To bag C , we draw bootstrap samples S^{*1}, \dots, S^{*B} each of size N with replacement from the training data. Then

$$\hat{C}_{bag}(x) = \text{Majority Vote } \{C(S^{*b}, x)\}_{b=1}^B.$$

Bagging can dramatically reduce the variance of unstable procedures (like trees), leading to improved prediction. However any simple structure in C (e.g a tree) is lost.

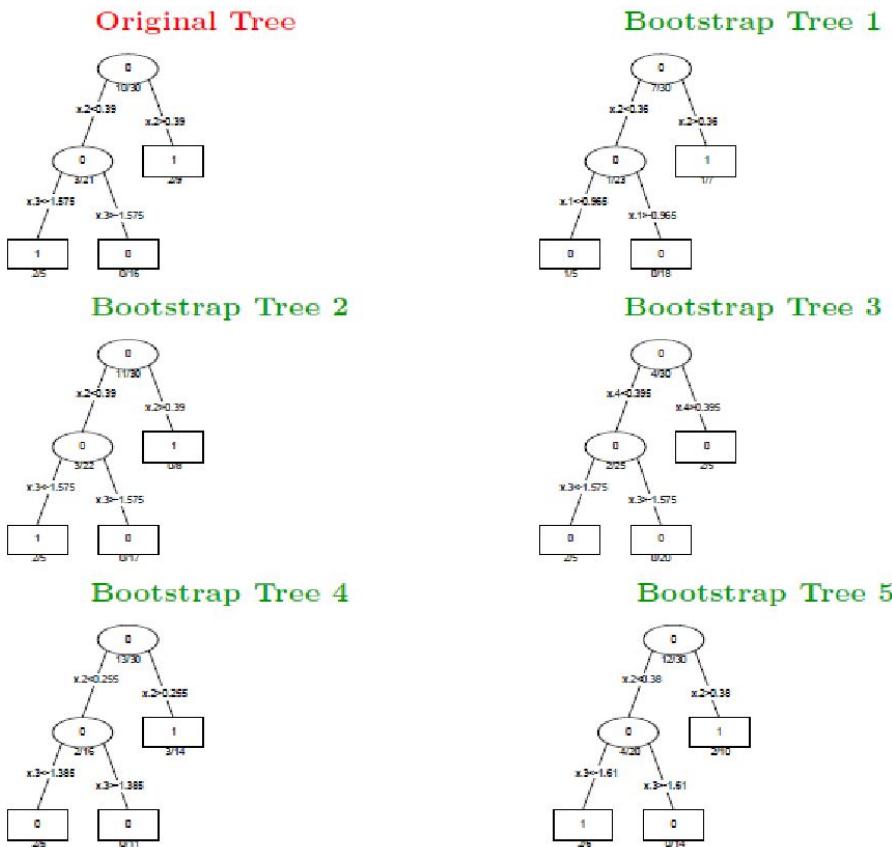
20

Generate Bootstrap Samples

- Generate new training sets using sampling with replacement (bootstrap samples)

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- some examples may appear in more than one set
 - some examples will appear more than once in a set
 - for each set, the probability that a given example appears in it is
- $$\Pr(x \in D_i) = 1 - (1 - \frac{1}{n})^n \rightarrow 0.6322$$
- i.e., less than 2/3 of the examples appear in one bootstrap sample

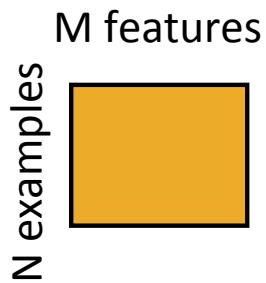


Random forest classifier

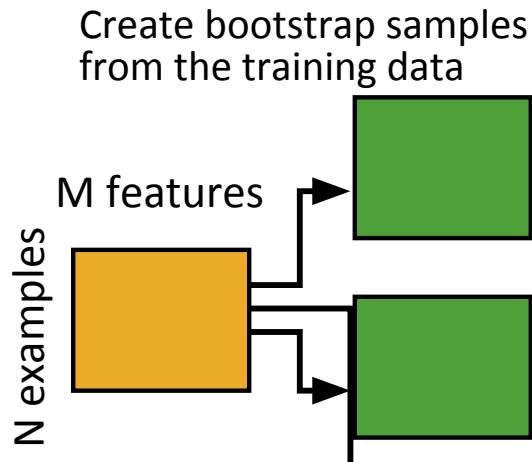
- Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Random Forest Classifier

Training Data

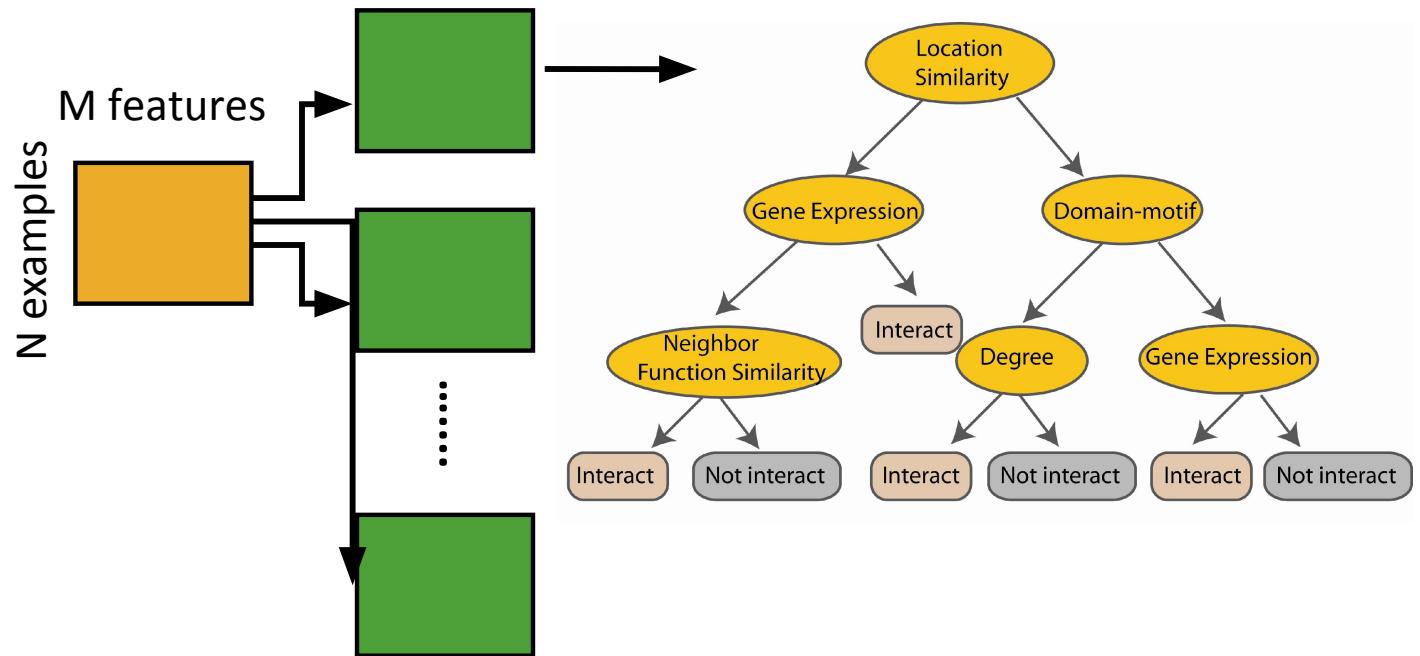


Random Forest Classifier

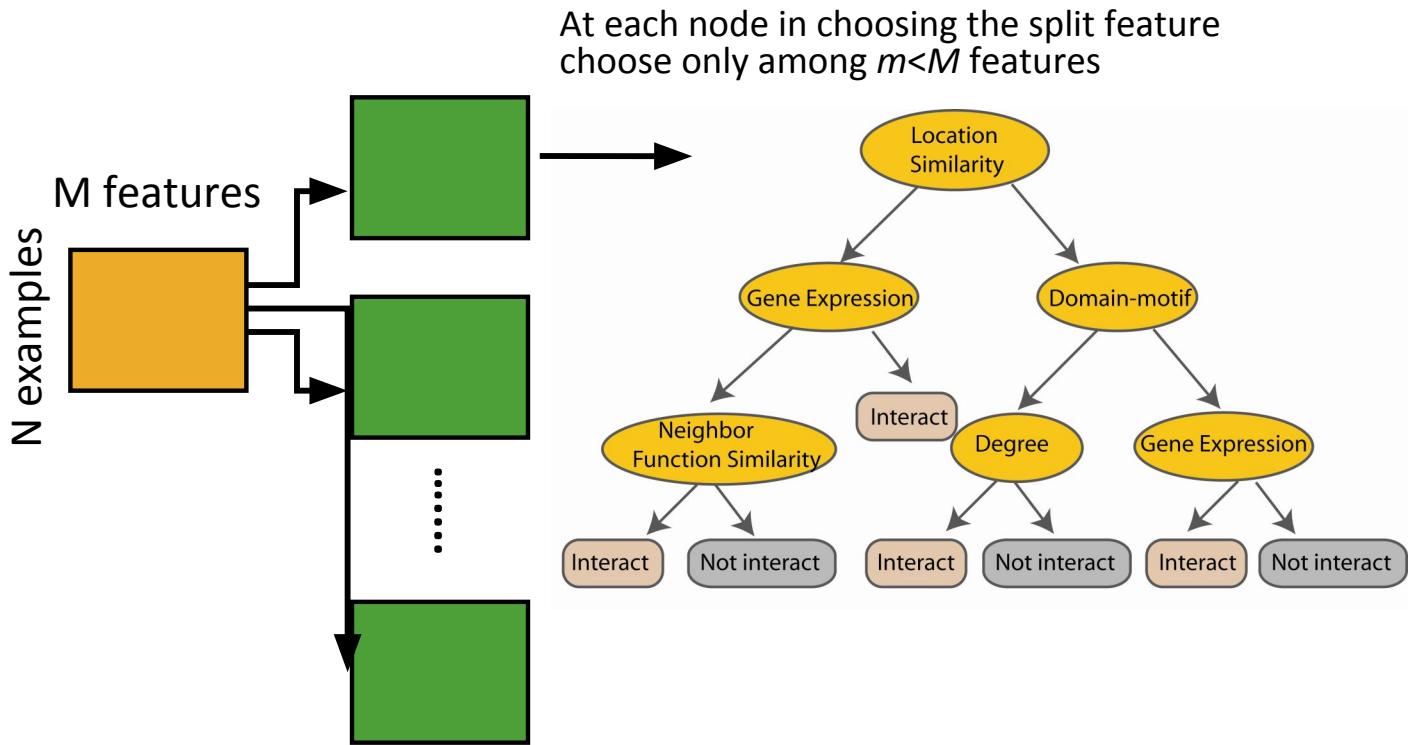


Random Forest Classifier

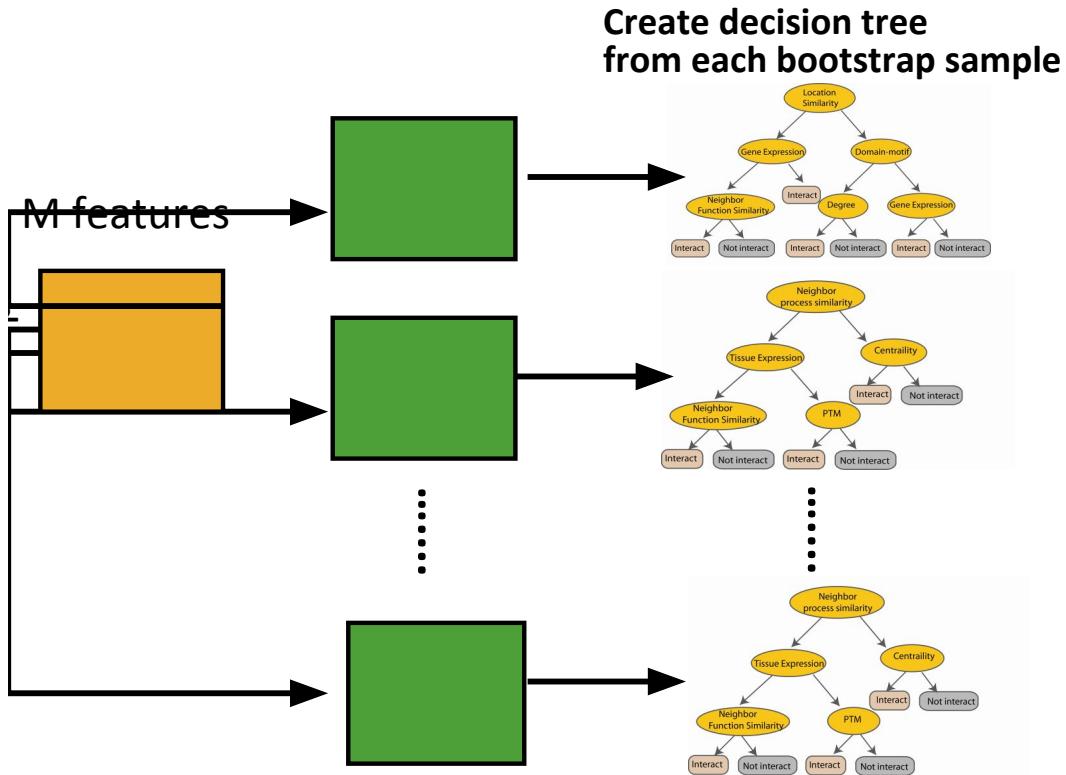
Construct a decision tree



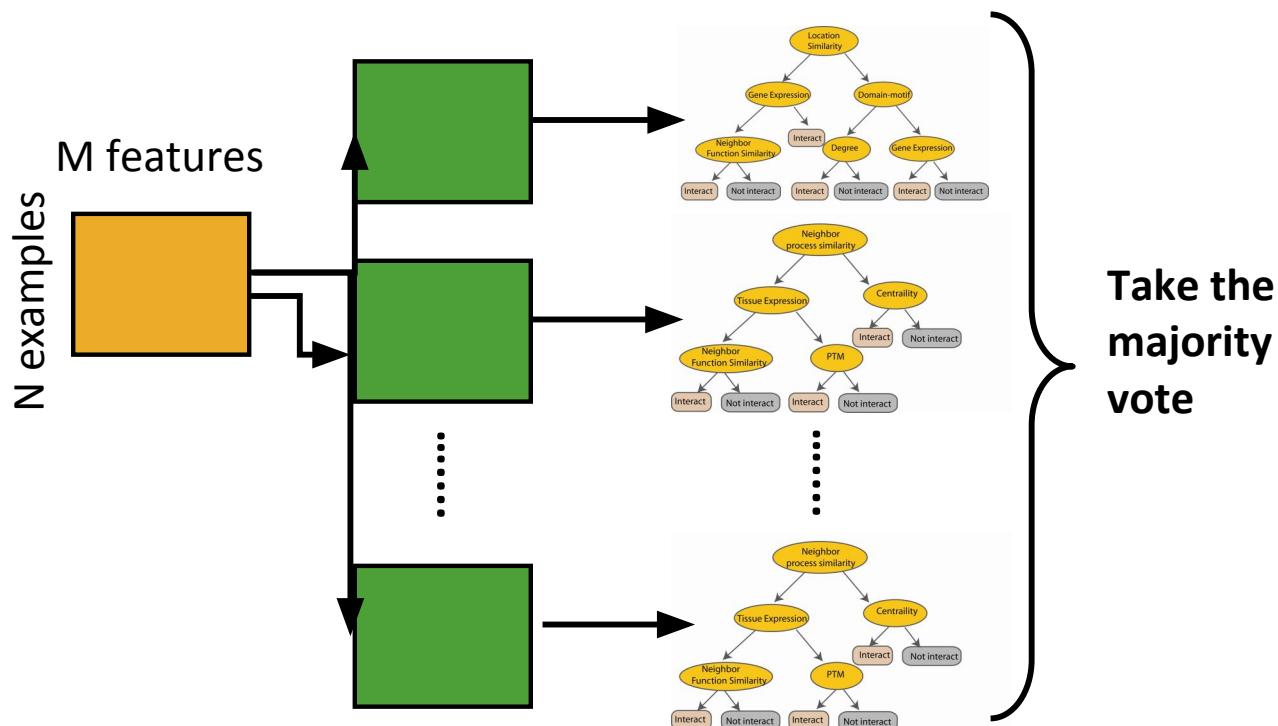
Random Forest Classifier



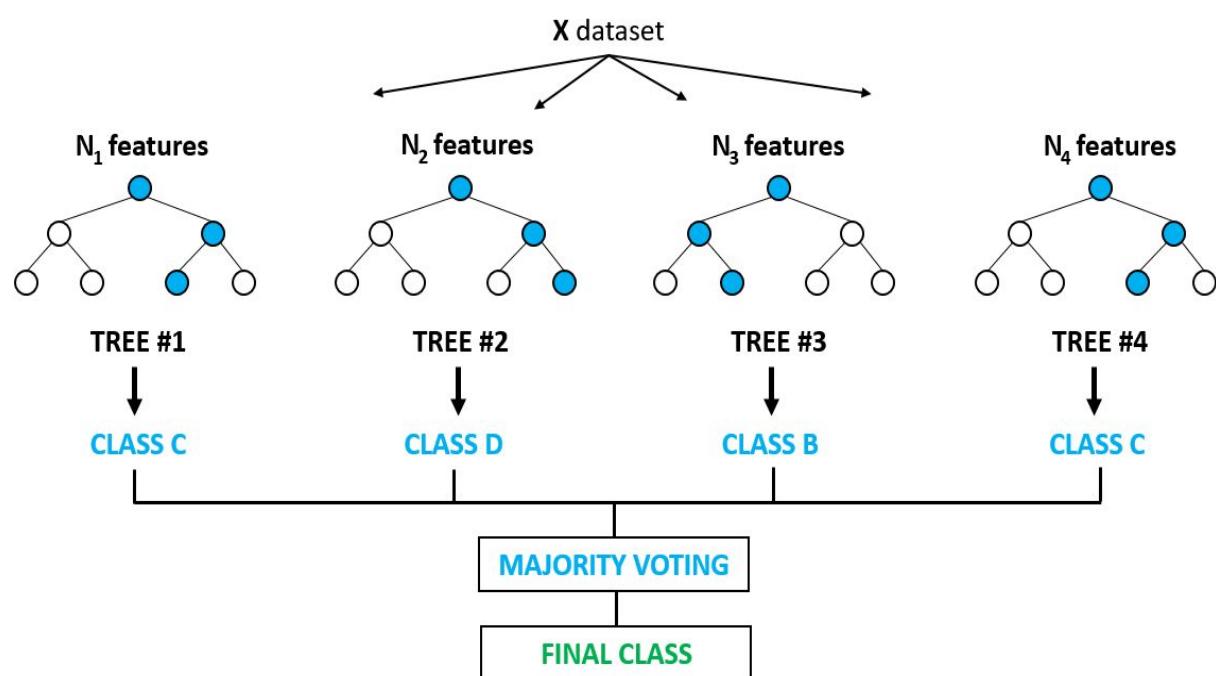
Random Forest Classifier



Random Forest Classifier



Thus,.....



Random forest

- Available package:
- http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
- To read more:
- <http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf>

Boosting

Also works by manipulating training set, but classifiers trained sequentially

Each classifier trained given knowledge of the performance of previously trained classifiers: **focus on hard examples**

Final classifier: weighted sum of component classifiers

Making weak learners stronger

- Suppose you have a weak learning module (a “base classifier”) that can always get $0.5 + \epsilon$ correct when given a two-way classification task
 - That seems like a weak assumption but beware!
- Can you apply this learning module many times to get a strong learner that can get close to zero error rate on the training data?
 - Theorists showed how to do this and it actually led to an effective new learning procedure (Freund & Shapire, 1996)

33

Boosting (ADABoost)

- First train the base classifier on all the training data with equal importance weights on each case.
- Then re-weight the training data to emphasize the hard cases and train a second model.
 - How do we re-weight the data?
- Keep training new models on re-weighted data
- Finally, use a weighted committee of all the models for the test data.
 - How do we weight the models in the committee?

34

Boosting

Training data:

$$\{(x^1, \hat{y}^1), \dots, (x^n, \hat{y}^n), \dots, (x^N, \hat{y}^N)\}$$

$\hat{y} = \pm 1$ (binary classification)

- Guarantee:

- If your ML algorithm can produce classifier with error rate smaller than 50% on training data
- You can obtain 0% error rate classifier after boosting.

- Framework of boosting

- Obtain the first classifier $f_1(x)$
- Find another function $f_2(x)$ to help $f_1(x)$
 - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
 - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
- Obtain the second classifier $f_2(x)$
- Finally, combining all the classifiers

- The classifiers are learned sequentially.

• Slides courtesy: Dr. Hung-yi Lee, NTU, Taiwan

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
 - Re-sampling your training data to form a new set
 - Re-weighting your training data to form a new set
 - In real implementation, you only have to change the cost/objective function

$$(x^1, \hat{y}^1, u^1) \quad u^1 = 1 \quad 0.4$$

$$(x^2, \hat{y}^2, u^2) \quad u^2 = 1 \quad 2.1$$

$$(x^3, \hat{y}^3, u^3) \quad u^3 = 1 \quad 0.7$$

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

$$L(f) = \sum_n u^n l(f(x^n), \hat{y}^n)$$

Idea of Adaboost

- Idea: training $f_2(x)$ on the new training set that fails $f_1(x)$

- How to find a new training set that fails $f_1(x)$?

ε_1 : the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \quad Z_1 = \sum_n u_1^n \quad \varepsilon_1 < 0.5$$

Changing the example weights from u_1^n to u_2^n such that

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

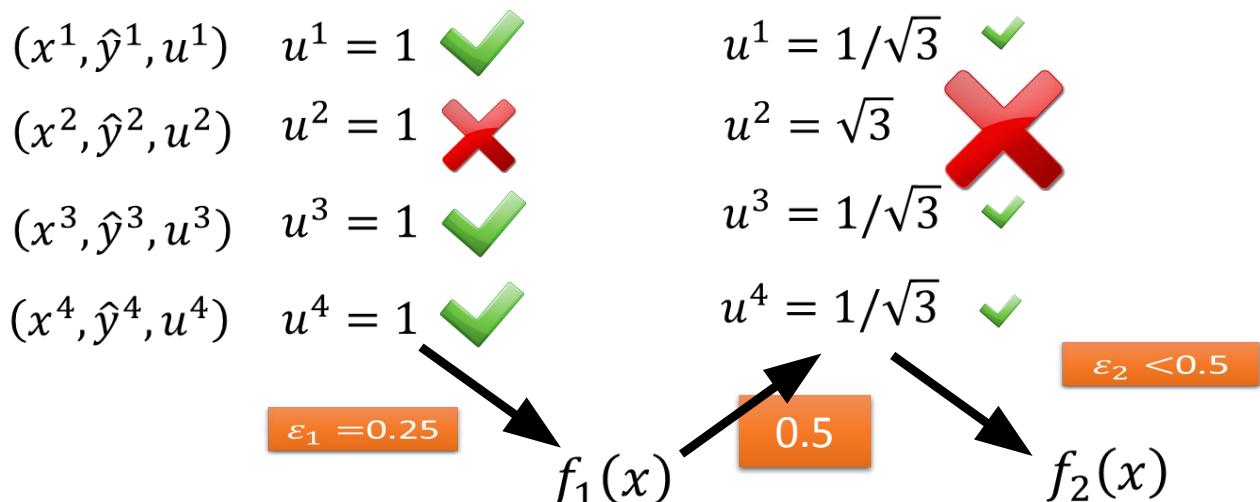
The performance of f_1 for new weights would be random.

Training $f_2(x)$ based on the new weights u_2^n

Re-weighting Training Data

- Idea: training $f_2(x)$ on the new training set that fails $f_1(x)$

- How to find a new training set that fails $f_1(x)$?



Re-weighting Training Data

- Idea: training $f_2(x)$ on the new training set that fails $f_1(x)$
- How to find a new training set that fails $f_1(x)$?

If x^n misclassified by f_1 ($f_1(x^n) \neq \hat{y}^n$)
 $u_2^n \leftarrow u_1^n$ multiplying d_1 increase
 If x^n correctly classified by f_1 ($f_1(x^n) = \hat{y}^n$)
 $u_2^n \leftarrow u_1^n$ divided by d_1 decrease

f_2 will be learned based on example weights u_2^n

What is the value of d_1 ?

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \quad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5 \quad \begin{array}{ll} f_1(x^n) \neq \hat{y}^n & u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \\ f_1(x^n) = \hat{y}^n & u_2^n \leftarrow u_1^n \text{ divided by } d_1 \end{array}$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 = \sum_{f_1(x^n) \neq \hat{y}^n} u_2^n + \sum_{f_1(x^n) = \hat{y}^n} u_2^n$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \quad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5 \quad \begin{array}{ll} f_1(x^n) \neq \hat{y}^n & u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \\ f_1(x^n) = \hat{y}^n & u_2^n \leftarrow u_1^n \text{ divided by } d_1 \end{array}$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2 \quad \frac{\sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 \quad \frac{1}{d_1} \sum_{f_1(x^n) = \hat{y}^n} u_1^n = d_1 \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n}{Z_1} \quad Z_1(1 - \varepsilon_1) \quad Z_1 \varepsilon_1$$

$$\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n = Z_1 \varepsilon_1 \quad Z_1(1 - \varepsilon_1) / d_1 = Z_1 \varepsilon_1 d_1$$

$$d_1 = \sqrt{(1 - \varepsilon_1) / \varepsilon_1} > 1$$

Algorithm for AdaBoost

- Giving training data $\{(x^1, \hat{y}^1, u_1^1), \dots, (x^n, \hat{y}^n, u_1^n), \dots, (x^N, \hat{y}^N, u_1^N)\}$
 - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For $t = 1, \dots, T$:
 - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \dots, u_t^N\}$
 - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \dots, u_t^N\}$
 - For $n = 1, \dots, N$:
 - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
 - $u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$ $d_t = \sqrt{(1 - \varepsilon_t) / \varepsilon_t}$
 - Else:
 - $u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t)$ $\alpha_t = \ln \sqrt{(1 - \varepsilon_t) / \varepsilon_t}$

$$\boxed{\begin{array}{l} \bullet \text{ If } x^n \text{ is misclassified by } f_t(x): \hat{y}^n \neq f_t(x^n) \\ \bullet u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t) \quad d_t = \sqrt{(1 - \varepsilon_t) / \varepsilon_t} \\ \bullet \text{ Else:} \\ \bullet u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t) \quad \alpha_t = \ln \sqrt{(1 - \varepsilon_t) / \varepsilon_t} \end{array}}$$

$$u_{t+1}^n \leftarrow u_t^n \times \exp(-\alpha_t)$$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \dots, f_t(x), \dots, f_T(x)$

- How to aggregate them?

- Uniform weight:

- $H(x) = \text{sign}(\sum_{t=1}^T f_t(x))$

- Non-uniform weight:

- $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t f_t(x))$

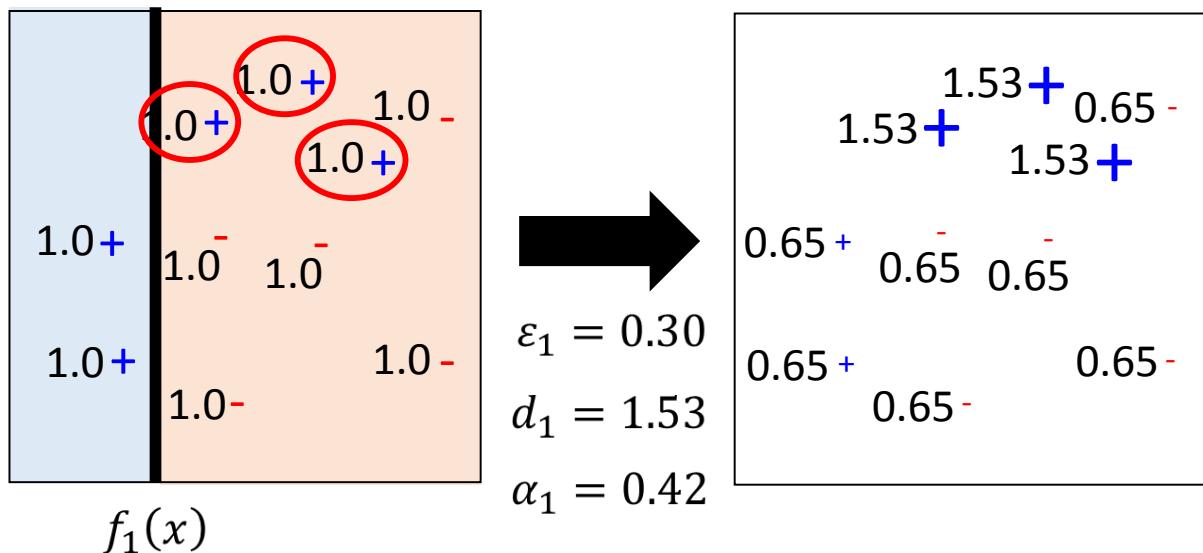
Smaller error ε_t ,
larger weight for
final voting

$$\alpha_t = \ln \sqrt{(1 - \varepsilon_t) / \varepsilon_t} \quad \varepsilon_t = 0.1 \quad \varepsilon_t = 0.4$$

$$u_{t+1}^n = u_t^n \times \exp(-\hat{y}^n f_t(x^n) \alpha_t) \quad \alpha_t = 1.10 \quad \alpha_t = 0.20$$

Toy Example T=3, weak classifier = decision stump

- t=1

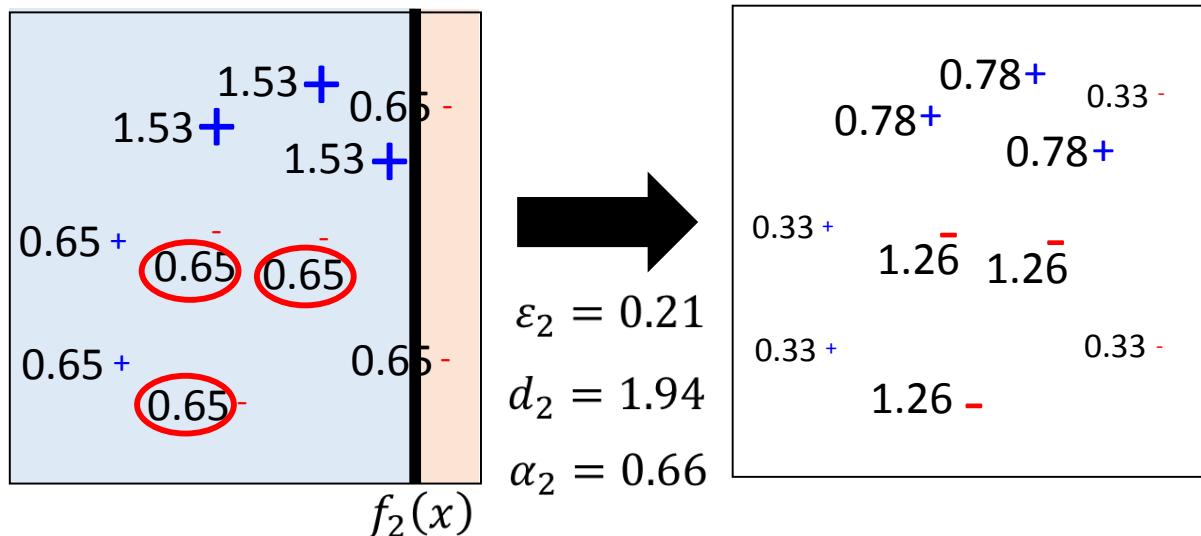
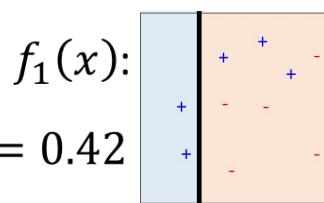


Toy Example

T=3, weak classifier = decision stump

- t=2

$$\alpha_1 = 0.42$$

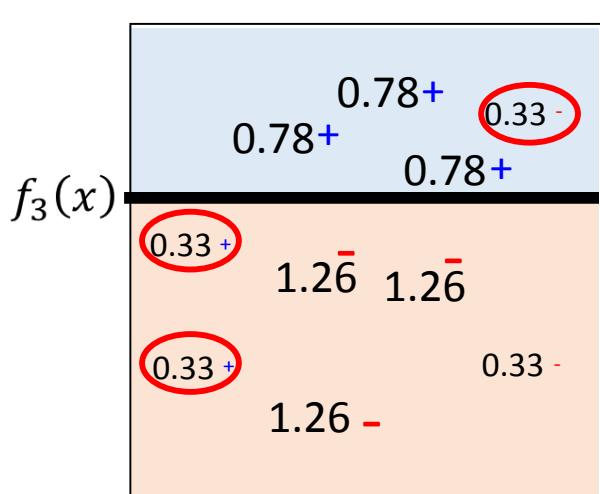
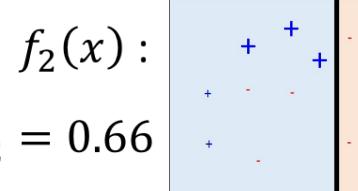
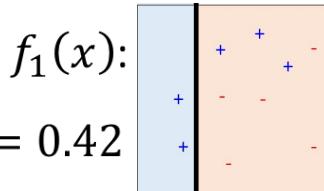


Toy Example

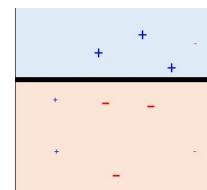
T=3, weak classifier = decision stump

- t=3

$$\alpha_1 = 0.42$$

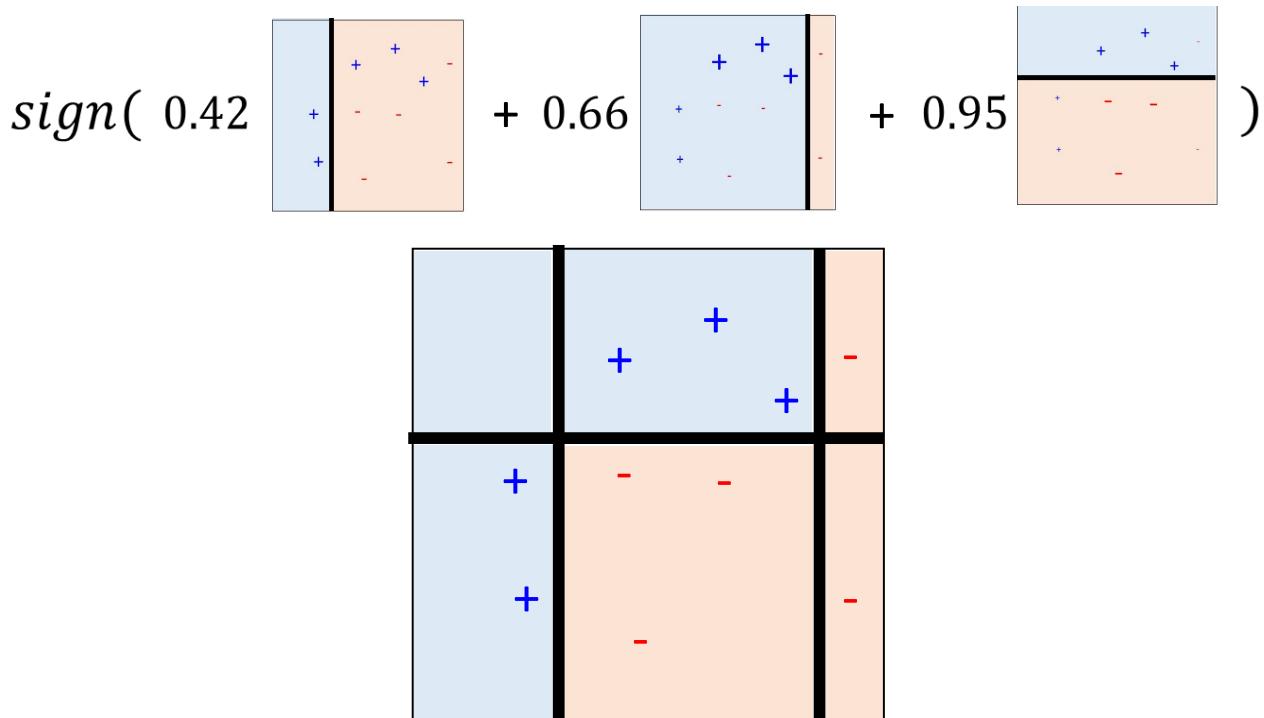


$$\alpha_3 = 0.95$$



Toy Example

- Final Classifier: $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t f_t(x))$



Combining Predictions

- voting
 - each ensemble member votes for one of the classes
 - predict the class with the highest number of vote (e.g., bagging)
- weighted voting
 - make a *weighted* sum of the votes of the ensemble members
 - weights typically depend
 - on the classifiers confidence in its prediction (e.g., the estimated probability of the predicted class)
 - on error estimates of the classifier (e.g., boosting)
- stacking
 - Why not use a classifier for making the final decision?
 - training material are the class labels of the training data and the (cross-validated) predictions of the ensemble members

Stacking

- Basic Idea:
 - learn a function that combines the predictions of the individual classifiers
- Algorithm:
 - train n different classifiers $C_1 \dots C_n$ (the *base classifiers*)
 - obtain predictions of the classifiers for the training examples
 - better do this with a cross-validation!
 - form a new data set (the *meta data*)
 - **classes**
 - the same as the original dataset
 - **attributes**
 - one attribute for each base classifier
 - value is the prediction of this classifier on the example
 - train a separate classifier M (the *meta classifier*)

Stacking (2)

- Example:

Attributes		Class
x_{11}	\dots	x_{1n_a}
x_{21}	\dots	x_{2n_a}
\dots	\dots	\dots
$x_{n_e 1}$	\dots	$x_{n_e n_a}$

training set

C_1	C_2	\dots	C_{n_c}
t	t	\dots	f
f	t	\dots	t
\dots	\dots	\dots	\dots
f	f	\dots	t

predictions of the classifiers

C_1	C_2	\dots	C_{n_c}	Class
t	t	\dots	f	t
f	t	\dots	t	f
\dots	\dots	\dots	\dots	\dots
f	f	\dots	t	t

training set for stacking

- Using a stacked classifier:

- try each of the classifiers $C_1 \dots C_n$
- form a feature vector consisting of their predictions
- submit this feature vectors to the meta classifier M

Mathematics of Boosting

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t f_t(x) \right) \quad \alpha_t = \ln \sqrt{(1 - \varepsilon_t) / \varepsilon_t}$$

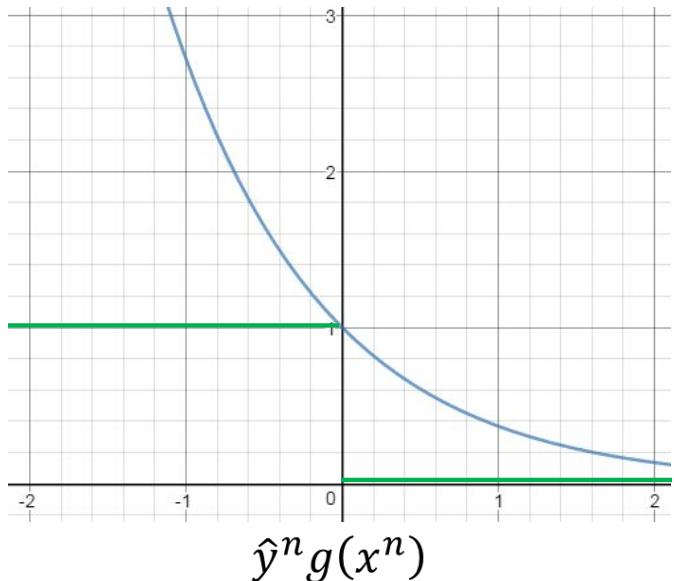
As we have more and more f_t (T increases), $H(x)$ achieves smaller and smaller error rate on training data.

Error Rate of Final Classifier

- Final classifier: $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t f_t(x))$
• $\alpha_t = \ln \sqrt{(1 - \varepsilon_t) / \varepsilon_t} \quad g(x)$

Training Data Error Rate

$$\begin{aligned} &= \frac{1}{N} \sum_n \delta(H(x^n) \neq \hat{y}^n) \\ &= \frac{1}{N} \sum_n \delta(\hat{y}^n g(x^n) < 0) \\ &\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n)) \end{aligned}$$



Training Data Error Rate

$$\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$$

$$g(x) = \sum_{t=1}^T \alpha_t f_t(x)$$

$$\alpha_t = \ln \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

Z_t : the summation of the weights of training data for training f_t

What is $Z_{T+1} = ?$ $Z_{T+1} = \sum_n u_{T+1}^n$

$$\left. \begin{array}{l} u_1^n = 1 \\ u_{t+1}^n = u_t^n \times \exp(-\hat{y}^n f_t(x^n) \alpha_t) \end{array} \right\} u_{T+1}^n = \prod_{t=1}^T \exp(-\hat{y}^n f_t(x^n) \alpha_t)$$

$$\begin{aligned} Z_{T+1} &= \sum_n \prod_{t=1}^T \exp(-\hat{y}^n f_t(x^n) \alpha_t) \\ &= \sum_n \exp \left(-\hat{y}^n \sum_{t=1}^T f_t(x^n) \alpha_t \right) \end{aligned}$$

Training Data Error Rate

$$\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$$

$$g(x) = \sum_{t=1}^T \alpha_t f_t(x)$$

$$\alpha_t = \ln \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

$Z_1 = N$ (equal weights)

$$Z_t = \underline{Z_{t-1} \varepsilon_t \exp(\alpha_t)} + \underline{Z_{t-1} (1 - \varepsilon_t) \exp(-\alpha_t)}$$

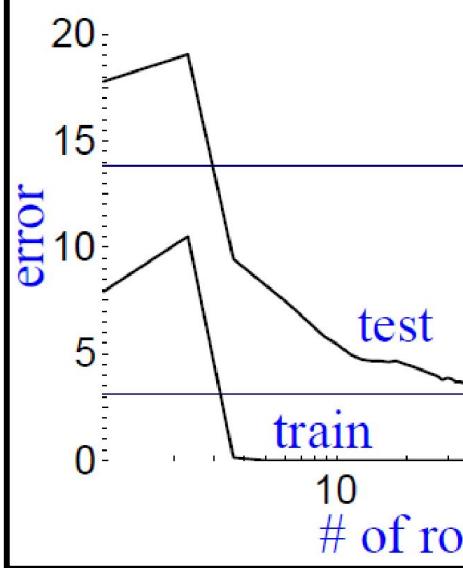
Misclassified portion in Z_{t-1} Correctly classified portion in Z_{t-1}

$$= Z_{t-1} \varepsilon_t \sqrt{(1 - \varepsilon_t)/\varepsilon_t} + Z_{t-1} (1 - \varepsilon_t) \sqrt{\varepsilon_t/(1 - \varepsilon_t)}$$

$$= Z_{t-1} \times 2 \sqrt{\varepsilon_t (1 - \varepsilon_t)} \quad Z_{T+1} = N \prod_{t=1}^T 2 \sqrt{\varepsilon_t (1 - \varepsilon_t)}$$

$$\text{Training Data Error Rate} \leq \prod_{t=1}^T \underline{2 \sqrt{\varepsilon_t (1 - \varepsilon_t)}} < 1$$

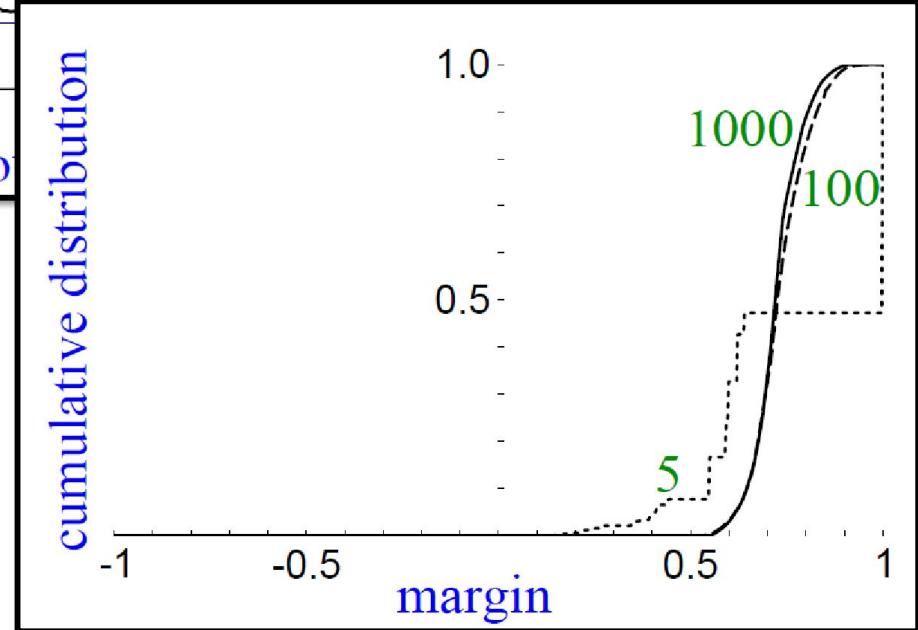
Smaller and smaller



Even though the training error is 0,
the testing error still decreases?

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t f_t(x) \right)$$

$\frac{g(x)}{\text{Margin}} = \hat{y}g(x)$



Large Margin?

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t f_t(x) \right)$$

$\frac{g(x)}{\text{Margin}} = \hat{y}g(x)$

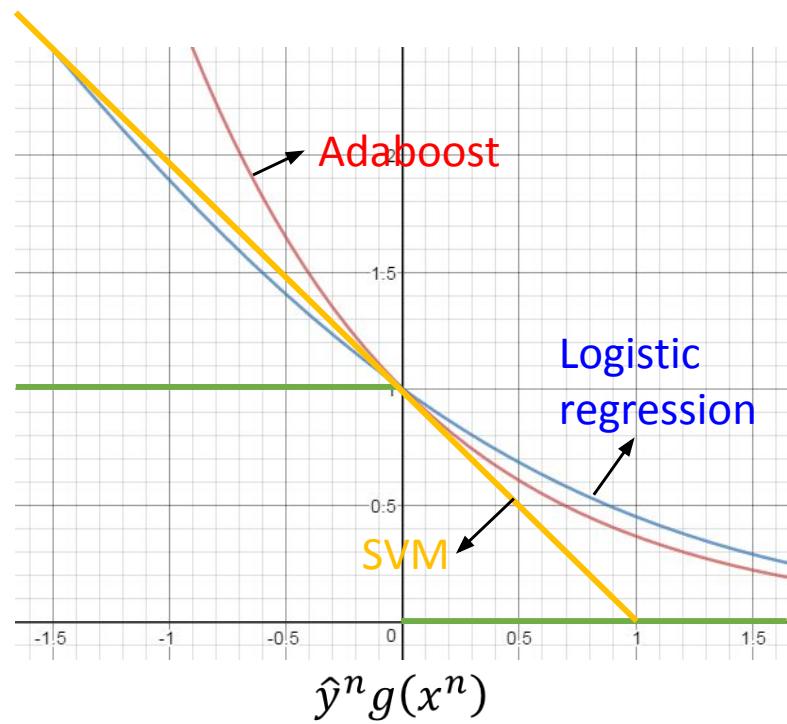
Training Data Error Rate =

$$= \frac{1}{N} \sum_n \delta(H(x^n) \neq \hat{y}^n)$$

$$\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n))$$

$$= \prod_{t=1}^T 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Getting smaller and
smaller as T increase



To learn more ...

- Introduction of Adaboost:
 - Freund; Schapire (1999). "A Short Introduction to Boosting"
- Multiclass/Regression
 - Y. Freund, R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.
 - Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory, pages 80–91, 1998.
- Gentle Boost
 - Schapire, Robert; Singer, Yoram (1999). "Improved Boosting Algorithms Using Confidence-rated Predictions".

General Formulation of Boosting

- Initial function $g_0(x) = 0$
- For $t = 1$ to T :
 - Find a function $f_t(x)$ and α_t to improve $g_{t-1}(x)$
 - $g_{t-1}(x) = \sum_{i=1}^{t-1} \alpha_i f_i(x)$
 - $g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$
 - Output: $H(x) = \text{sign}(g_T(x))$

What is the learning target of $g(x)$?

$$\text{Minimize } L(g) = \sum_n l(\hat{y}^n, g(x^n)) = \sum_n \exp(-\hat{y}^n g(x^n))$$

Gradient Boosting

- Find $g(x)$, minimize $L(g) = \sum_n \exp(-\hat{y}^n g(x^n))$
 - If we already have $g(x) = g_{t-1}(x)$, how to update $g(x)$?

Gradient Descent:

$$g_t(x) = g_{t-1}(x) - \eta \frac{\partial L(g)}{\partial g(x)} \Big|_{g(x) = g_{t-1}(x)}$$

$\sum_n \exp(-\hat{y}^n g_{t-1}(x^n))(\hat{y}^n)$

$g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$

Same direction

Gradient Boosting

$$f_t(x) \longleftrightarrow \sum_n \exp(-\hat{y}^n g_t(x^n))(\hat{y}^n)$$

Same direction

We want to find $f_t(x)$ maximizing

$$\sum_n \frac{\exp(-\hat{y}^n g_{t-1}(x^n))}{\text{example weight } u_t^n} (\hat{y}^n) f_t(x^n)$$

Minimize Error

Same sign

$$u_t^n = \exp(-\hat{y}^n g_{t-1}(x^n)) = \exp\left(-\hat{y}^n \sum_{i=1}^{t-1} \alpha_i f_i(x^n)\right)$$

$$= \prod_{i=1}^{t-1} \exp(-\hat{y}^n \alpha_i f_i(x^n))$$

Exactly the weights we obtain in Adaboost

Gradient Boosting

- Find $g(x)$, minimize $L(g) = \sum_n \exp(-\hat{y}^n g(x^n))$

$$g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$$

α_t is something like
learning rate

Find α_t minimizing $L(g_{t+1})$

$$\begin{aligned} L(g) &= \sum_n \exp(-\hat{y}^n (g_{t-1}(x) + \alpha_t f_t(x))) \\ &= \sum_n \exp(-\hat{y}^n g_{t-1}(x)) \exp(-\hat{y}^n \alpha_t f_t(x)) \\ &= \sum_{\hat{y}^n \neq f_t(x)} \exp(-\hat{y}^n g_{t-1}(x^n)) \exp(\alpha_t) \\ &\quad + \sum_{\hat{y}^n = f_t(x)} \exp(-\hat{y}^n g_{t-1}(x^n)) \exp(-\alpha_t) \end{aligned}$$

Find α_t
such that

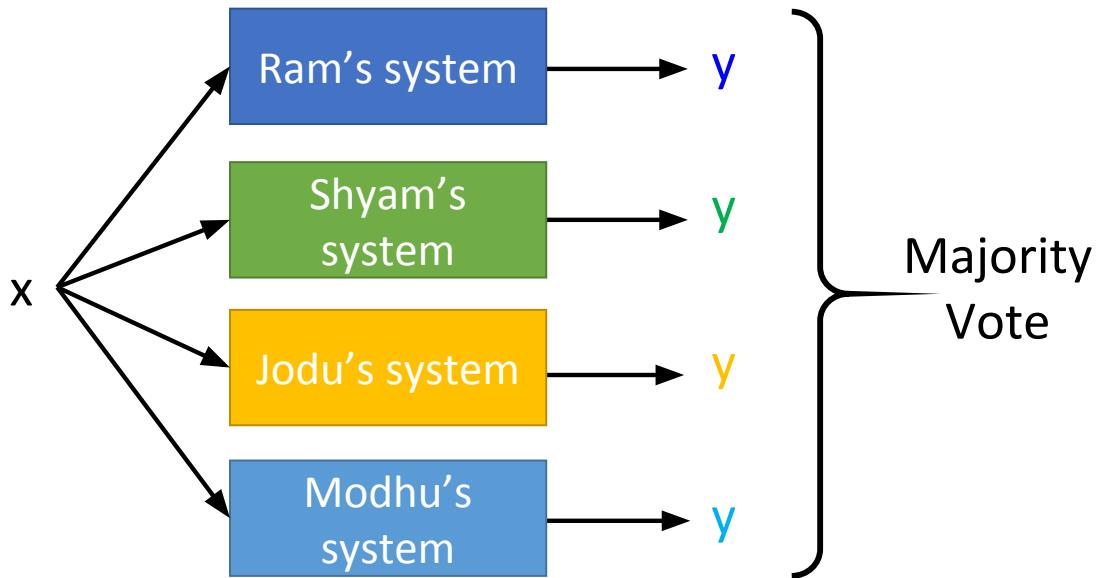
$$\frac{\partial L(g)}{\partial \alpha_t} = 0$$

$$\alpha_t = \ln \sqrt{(1 - \varepsilon_t) / \varepsilon_t}$$

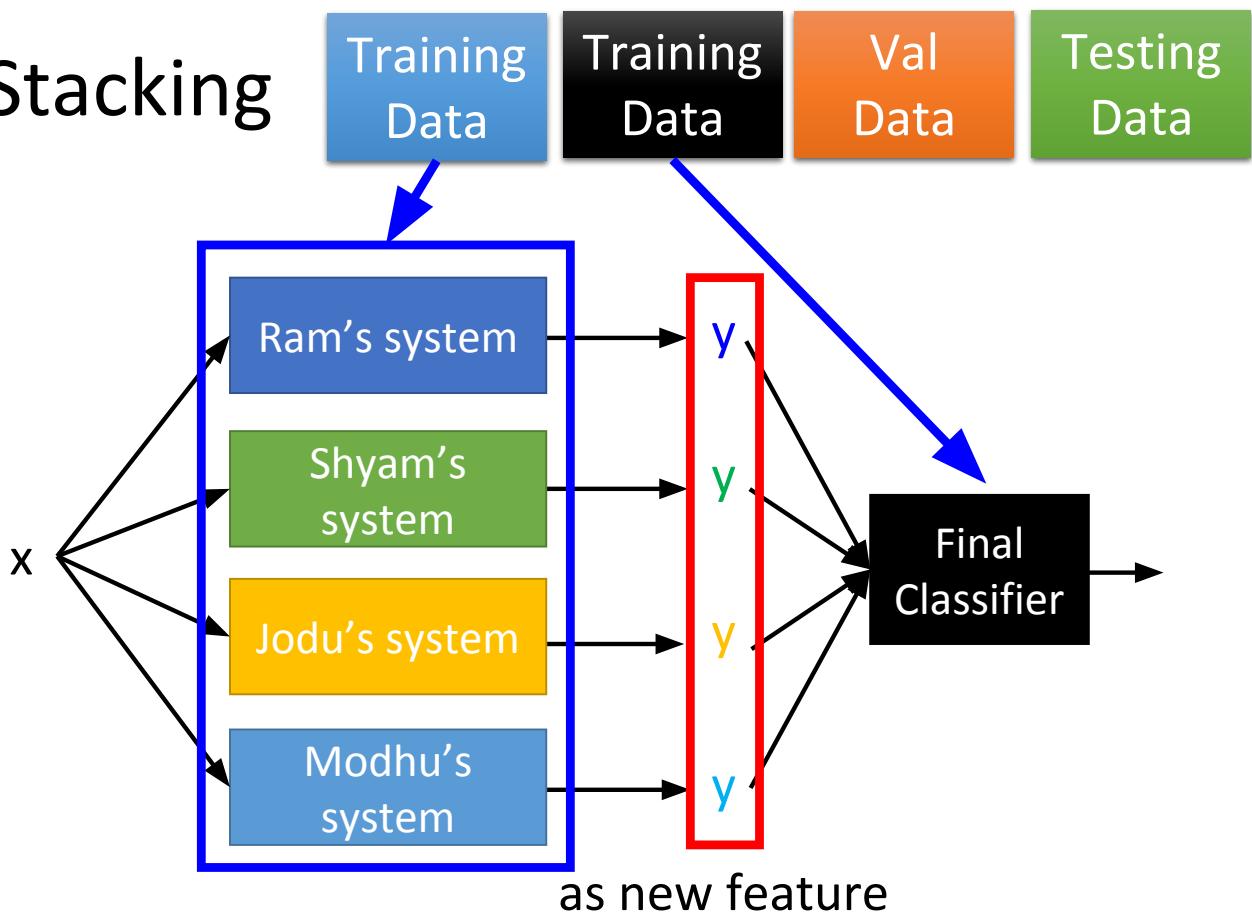
Adaboost
!

Ensemble: Stacking

Voting



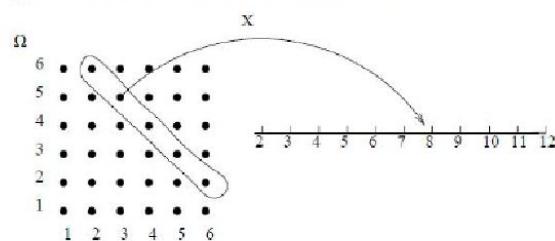
Stacking



Statistical Pattern Recognition – A Primer

Quick Review: Discrete/Continuous Random Variables

- A Random Variable is a function $X : \Omega \mapsto \mathbb{R}$
- The set of all possible values a random variable X can take is called its **range**
- **Discrete** random variables can only take isolated values (probability of a random variable taking a particular value reduces to counting)
- Discrete Example: Sum of two fair dice



- Continuous Example: Speed of a car

Discrete Distributions

- Assume X is a discrete random variable. We would like to specify probabilities of events $\{X = x\}$
- If we can specify the probabilities involving X , we can say that we have specified the probability distribution of X
- For a countable set of values x_1, x_2, \dots, x_n , we have $\mathbb{P}(X = x_i) > 0, i = 1, 2, \dots, n$ and $\sum_i \mathbb{P}(X = x_i) = 1$
- We can then define the **probability mass function** f of X by $f(X) = \mathbb{P}(X = x)$
- Sometimes write as f_X

Probability Mass Function

- Example: Toss a die and let X be its face value. X is discrete with range $\{1, 2, 3, 4, 5, 6\}$. The pmf is

x	1	2	3	4	5	6	Σ
$f(x)$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	1

- Another example: Toss two dice and let X be the largest face value. The pmf is

x	1	2	3	4	5	6	Σ
$f(x)$	$\frac{1}{36}$	$\frac{3}{36}$	$\frac{5}{36}$	$\frac{7}{36}$	$\frac{9}{36}$	$\frac{11}{36}$	1

Probability Density Functions

- A random variable X taking values in set \mathcal{X} is said to have a continuous distribution if $\mathbb{P}(X = x) = 0$ for all $x \in \mathcal{X}$
- The probability density function of a continuous random variable X satisfies
 - $f(x) \geq 0 \forall x$
 - $\int_{-\infty}^{\infty} f(x)dx = 1$
 - $\mathbb{P}(a \leq X \leq b) = \int_a^b f(x)dx \forall a, b$
- Probabilities correspond to areas under the curve $f(x)$
- Reminder: No longer need to have
 $\mathbb{P}(a \leq X \leq b) = \int_a^b f(x)dx \leq 1$ but must have
 $\int_{-\infty}^{\infty} f(x)dx = 1$



Expectations

The average value of a function $f(x)$ under a probability distribution $p(x)$ is called the **expectation** of $f(x)$.

Average is weighted by the relative probabilities of different values of x .

$$\mathbb{E}[f] = \sum_x p(x)f(x)$$

$$\mathbb{E}[f] = \int p(x)f(x) dx$$

$$\mathbb{E}[f] \simeq \frac{1}{N} \sum_{n=1}^N f(x_n)$$

Approximate Expectation
(discrete and continuous)

Now we are going to look at concepts: variance and co-variance , of 1 or more random variables, using the concept of expectation.

Variance and Covariance

The variance of $f(x)$ provides a measure for how much $f(x)$ varies around its mean $E[f(x)]$.

$$\text{var}[f] = \mathbb{E} \left[(f(x) - \mathbb{E}[f(x)])^2 \right] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$$

Given a set of N points $\{x_i\}$ in the 1D-space, the variance of the corresponding random variable x is $\text{var}[x] = E[(x-\mu)^2]$ where $\mu = E[x]$.

You can estimate the expected value as

$$\text{var}(x) = E[(x-\mu)^2] \approx 1/N \sum_{x_i} (x_i - \mu)^2$$

Remember the definition of expectation: $E[f] \approx \frac{1}{N} \sum_{n=1}^N f(x_n)$

Variance and Covariance

The variance of x provides a measure for how much x varies around its mean $\mu = E[x]$.

$$\text{var}(x) = E[(x-\mu)^2]$$

$$\text{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$

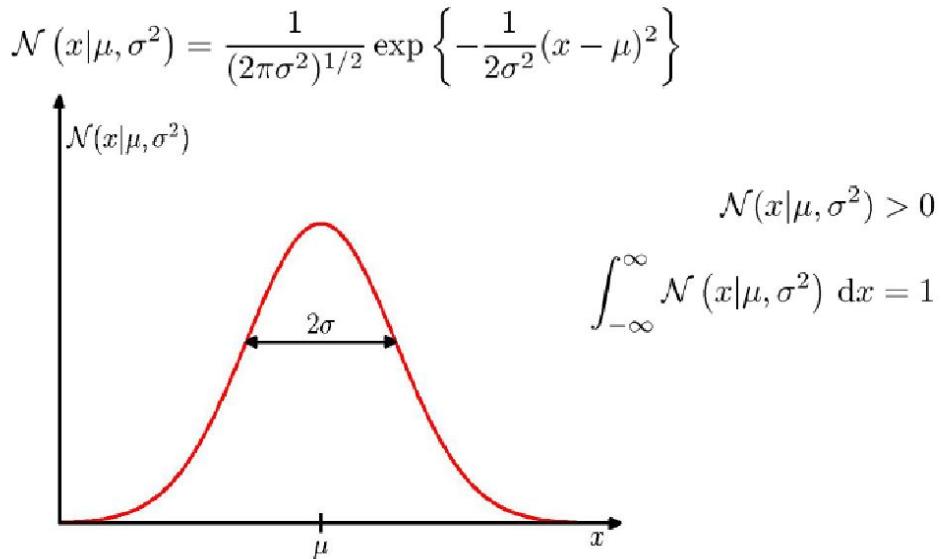
Co-variance of two random variables x and y measures the extent to which they vary together.

$$\begin{aligned} \text{cov}[x, y] &= \mathbb{E}_{x,y} [\{x - \mathbb{E}[x]\} \{y - \mathbb{E}[y]\}] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

: two random variables x, y

: two vector random variables \mathbf{x}, \mathbf{y} – covariance is a matrix

The Gaussian Distribution



Why Gaussians?

- Gaussian distributions are widely used in machine learning:
 - Central Limit Theorem!

$$\begin{aligned} \bar{X}_n &= X_1 + X_2 + \cdots + X_n \\ \sqrt{n}\bar{X}_n &\xrightarrow{d} \mathcal{N}(x; \mu, \sigma^2) \end{aligned}$$

- Actually, there are a set of "Central Limit Theorems" (e.g. corresponding to p -Stable Distributions)

Why Gaussians?

- Gaussian distributions are widely used in machine learning:
 - Central Limit Theorem!
 - Gaussians are convenient computationally;
 - Mixtures of Gaussians (just covered in class) are sufficient to approximate a wide range of distributions;
 - Closely related to squared loss (have seen earlier in class), an important error measure in statistics.

Expectations

$$\begin{aligned}\mathbb{E}[f] &= \sum_x p(x)f(x) & \mathbb{E}[f] &= \int p(x)f(x) dx & \mathbb{E}[f] &\simeq \frac{1}{N} \sum_{n=1}^N f(x_n) \\ \text{var}[f] &= \mathbb{E} \left[(f(x) - \mathbb{E}[f(x)])^2 \right] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2\end{aligned}$$

For normally distributed x:

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x dx = \mu$$

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x^2 dx = \mu^2 + \sigma^2$$

$$\text{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2$$

- Assume we have a d-dimensional input (e.g. 2D), \mathbf{x} .
- We will see how can we characterize $p(\mathbf{x})$, assuming \mathbf{x} is normally distributed.
 - For 1-dimension it was the **mean** (μ) and **variance** (σ^2)
 - Mean = $E[\mathbf{x}]$
 - Variance = $E[(\mathbf{x} - \mu)^2]$
 - For d-dimensions, we need
 - the d-dimensional **mean vector**
 - $d \times d$ dimensional **covariance matrix**
 - If $\mathbf{x} \sim N_d(\mu, \Sigma)$ then each dimension of \mathbf{x} is univariate normal
 - Converse is not true

Covariance

- The covariance of X and Y is defined as:

$$C_{XY} = E((X - \mu_X)(Y - \mu_Y))$$

- In practice, we can approximate $\text{Cov}(X, Y)$ by the **sample covariance**:

$$\bar{s} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

Covariance Matrix – 2 variables

- The covariance matrix of X and Y is given by:

$$\Sigma = C_{XY} = \begin{bmatrix} Cov(X, X) & Cov(X, Y) \\ Cov(Y, X) & Cov(Y, Y) \end{bmatrix}$$

where $Cov(X, X) = \text{Var}(X)$ and $Cov(Y, Y) = \text{Var}(Y)$

and $Cov(X, Y) = Cov(Y, X)$ (i.e., C_{XY} is **symmetric**)

15

Covariance Matrix – n variables

- The covariance matrix of **n** variables is given by:

$$C_X = \begin{bmatrix} Cov(X_1, X_1) & Cov(X_1, X_2) & \dots & Cov(X_1, X_n) \\ Cov(X_2, X_1) & Cov(X_2, X_2) & \dots & Cov(X_2, X_n) \\ \dots & \dots & \dots & \dots \\ Cov(X_n, X_1) & Cov(X_n, X_2) & \dots & Cov(X_n, X_n) \end{bmatrix}$$

where $Cov(X_i, X_j) = Cov(X_j, X_i)$ and $Cov(X_i, X_i) \geq 0$
(i.e., **symmetric matrix**)

- In practice, we can approximate C_X by the **sample covariance matrix**.

16

Normal Distribution & Multivariate Normal Distribution

- For a single variable, the normal density function is:

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

- For variables in higher dimensions, this generalizes to:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\}$$

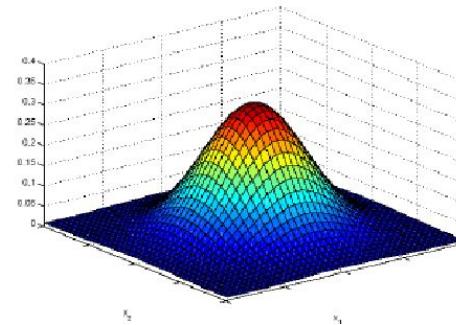
- where the mean $\boldsymbol{\mu}$ is now a d-dimensional vector.

$\boldsymbol{\Sigma}$ is a $d \times d$ covariance matrix

$|\Sigma|$ is the determinant of Σ :

$$\boldsymbol{\mu} = E[\mathbf{x}]$$

$$\boldsymbol{\Sigma} = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T].$$



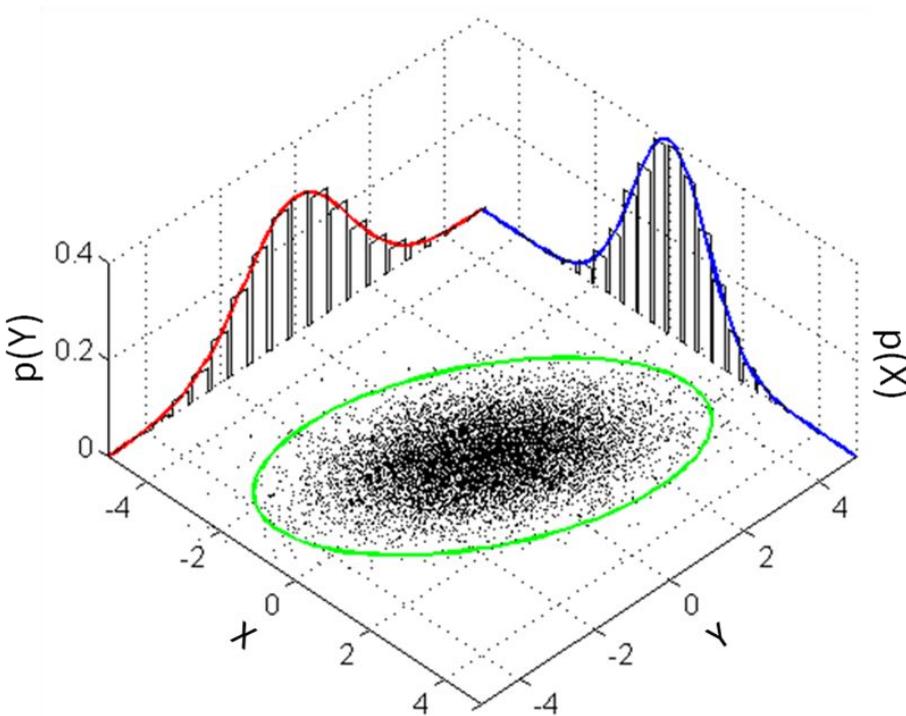
Multivariate Parameters: Mean, Covariance

$$\text{Mean: } E[\mathbf{x}] = \boldsymbol{\mu} = [\mu_1, \dots, \mu_d]^T$$

$$\boldsymbol{\Sigma} \equiv \text{Cov}(\mathbf{x}) = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix} = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = E[\mathbf{x}\mathbf{x}^T] - \boldsymbol{\mu}\boldsymbol{\mu}^T$$

$$\sigma_{ij} \equiv \text{Cov}(x_i, x_j) \equiv E[(x_i - \mu_i)(x_j - \mu_j)]$$

Bivariate Normal Distribution with Non-diagonal Covariance Matrix



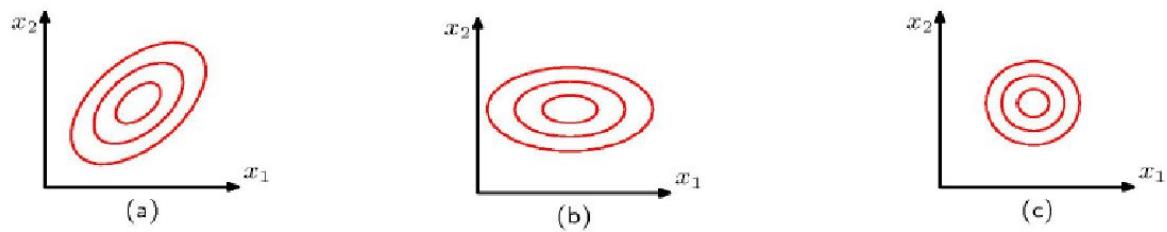
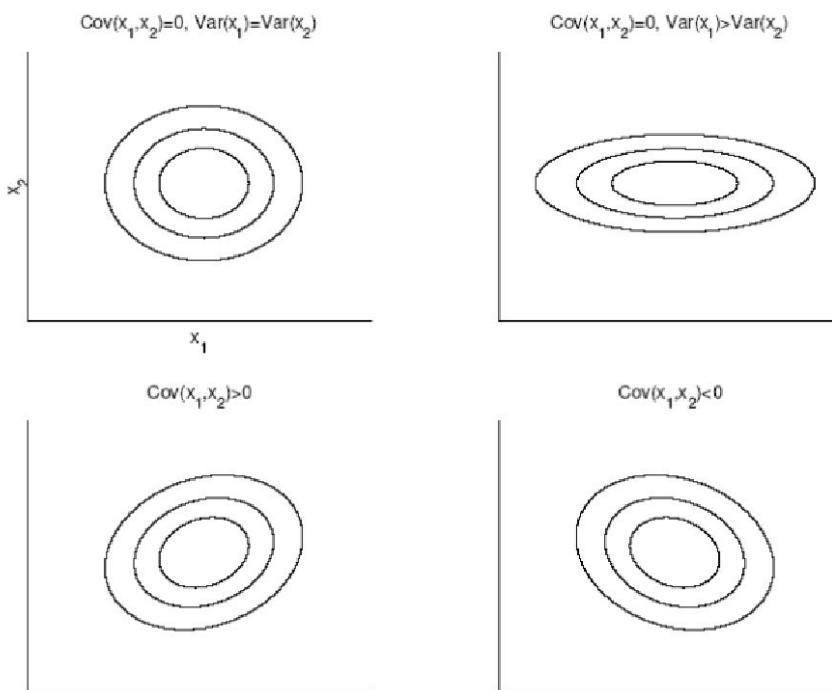
- **Variance:** How much X varies around the expected value
- **Covariance** is the measure the strength of the **linear relationship** between two random variables
 - covariance becomes more positive for each pair of values which differ from their mean in the same direction
 - covariance becomes more negative with each pair of values which differ from their mean in opposite directions.
 - if two variables are independent, then their covariance/correlation is zero (converse is not true).

- **Correlation** is a dimensionless measure of linear dependence.
 - range between -1 and +1

$$\text{Covariance: } \sigma_{ij} = \text{Cov}(x_i, x_j) = E[(x_i - \mu_i)(x_j - \mu_j)]$$

$$\text{Correlation: } \text{Corr}(x_i, x_j) = \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

Covariance Matrices



Contours of constant probability density for a 2D Gaussian distributions with

- general covariance matrix
- diagonal covariance matrix (covariance of x_1, x_2 is 0)
- Σ proportional to the identity matrix (covariances are 0, variances of each dimension are the same)



Shape and orientation of the hyper-ellipsoid centered at μ is defined by Σ

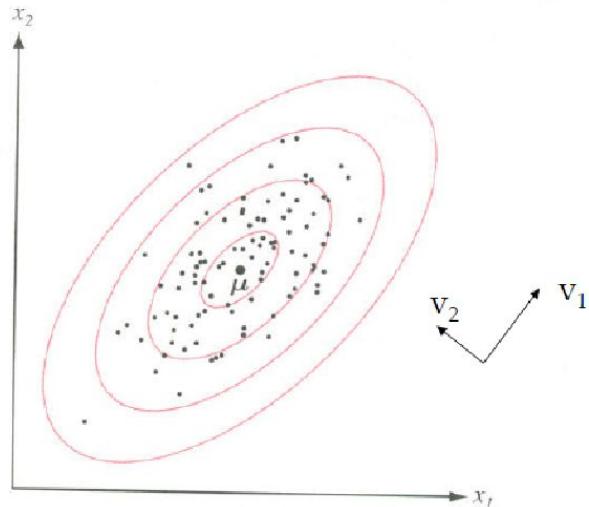


FIGURE 2.9. Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean μ . The ellipses show lines of equal probability density of the Gaussian.



Properties of Σ

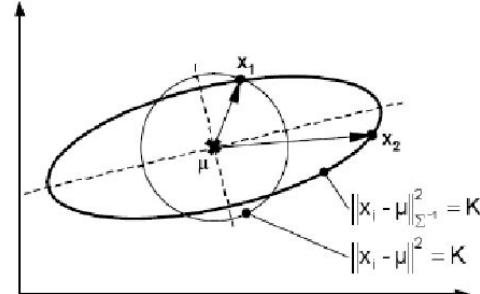
- A small value of $|\Sigma|$ (determinant of the covariance matrix) indicates that samples are close to μ
- Small $|\Sigma|$ may also indicate that there is a **high correlation** between variables
- If some of the variables are **linearly dependent**, or if the variance of one variable is 0, then Σ is **singular** and $|\Sigma|$ is 0.
 - Dimensionality should be reduced to get a positive definite matrix

Points that are the same distance from μ

- The quadratic term is called the Mahalanobis distance, a very important distance in Statistical PR

Mahalanobis Distance

$$\|x - y\|_{\Sigma^{-1}}^2 = (x - y)^T \Sigma^{-1} (x - y)$$



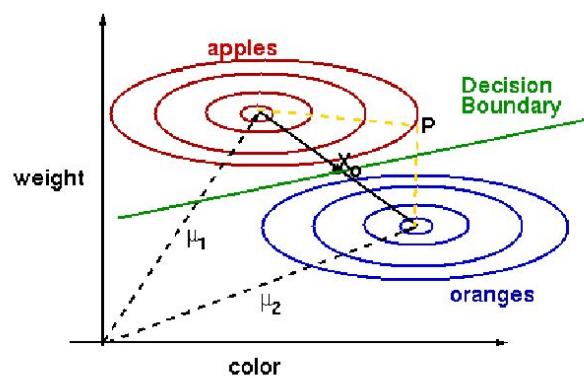
- The Mahalanobis distance is a vector distance that uses a Σ^{-1} norm

- Σ^{-1} can be thought of as a stretching factor on the space
- Note that for an identity covariance matrix ($\Sigma=I$), the Mahalanobis distance becomes the familiar Euclidean distance

- The ellipse consists of points that are equi-distant to the center w.r.t. Mahalanobis distance.
- The circle consists of points that are equi-distant to the center w.r.t. The Euclidian distance.



Why Mahalanobis Distance

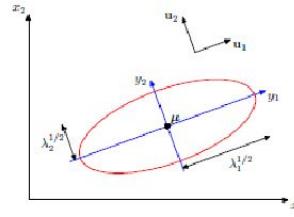


It takes into account the covariance of the data.

- Point P is at closer (Euclidean) to the mean for the orange class, but using the Mahalanobis distance, it is found to be closer to 'apple' class.

Density contours

- What are the constant density contours?



$$\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right) = \text{const}$$

$$(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = \text{const}$$

- This is a quadratic form, whose solution is an ellipsoid (in 2D, simply an ellipse)

Properties of the covariance

- Consider the eigenvector equation: $\boldsymbol{\Sigma}\mathbf{u} = \lambda\mathbf{u}$
- As a covariance matrix, $\boldsymbol{\Sigma}$ is symmetric $d \times d$ matrix. Therefore, we have d solutions $\{\lambda_i, \mathbf{u}_i\}_{i=1}^d$ where the eigenvalues λ_i are real, and the eigenvectors \mathbf{u}_i are orthonormal, i.e., inner product

$$\mathbf{u}_j^T \mathbf{u}_i = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

- The covariance matrix $\boldsymbol{\Sigma}$ then may be written as:
- $$\boldsymbol{\Sigma} = \sum_i \lambda_i \mathbf{u}_i \mathbf{u}_i^T$$
- Thus, the inverse covariance may be written as:
- $$\boldsymbol{\Sigma}^{-1} = \sum_i \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T$$

Continued..

- The quadratic form $(x - \mu)^T \Sigma^{-1} (x - \mu)$ becomes:

$$\sum_i \frac{y_i^2}{\lambda_i}$$

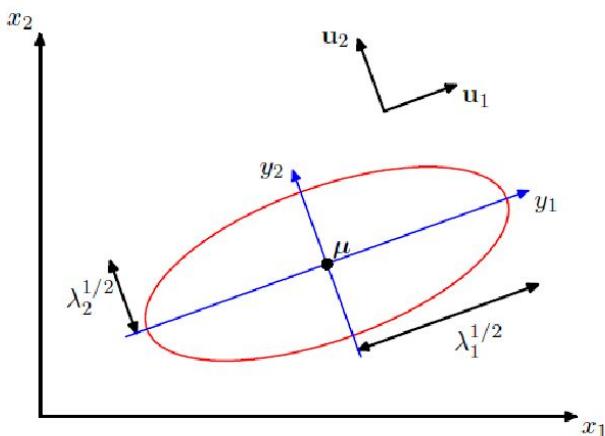
where $y_i = u_i^T (x - \mu)$

- $\{y_i\}$ may be interpreted as a new coordinate system defined by the orthonormal vectors u_i that are shifted and rotated with respect to the original coordinate system
- Stack the d transposed orthonormal eigenvectors of Σ into $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^T \\ \dots \\ \mathbf{u}_d^T \end{bmatrix}$. Then, $\mathbf{y} = \mathbf{U}(\mathbf{x} - \mu)$ defines rotation (and possibly reflection) of \mathbf{x} , shifted so that μ becomes origin.

Geometry of the Gaussian



- $\sqrt{\lambda_i}$ gives scaling along \mathbf{u}_i
- Example in 2D:



Geometry Continued ...

- The determinant of the covariance matrix may be written as the product of its eigenvalues i.e. $|\Sigma|^{\frac{1}{2}} = \prod_j \lambda_j^{\frac{1}{2}}$
- Thus, in the y_i coordinate system, the Gaussian distribution takes the form:

$$p(y) = \prod_j \frac{1}{(2\pi\lambda_j)^{\frac{1}{2}}} \exp\left(-\frac{y_j^2}{2\lambda_j}\right)$$

- which is the product of d independent univariate Gaussians
- The eigenvectors thus define a new set of shifted and rotated coordinates w.r.t which the joint probability distribution factorizes into a product of independent distributions

Density Contours are Ellipsoids

- We saw that: $(x - \mu)^T \Sigma^{-1} (x - \mu) = \text{const}^2$
- Recall that $\Sigma^{-1} = \sum_i \frac{1}{\lambda_i} u_i u_i^T$
- Thus we have:

$$\sum_i \frac{y_i^2}{\lambda_i} = \text{const}^2$$

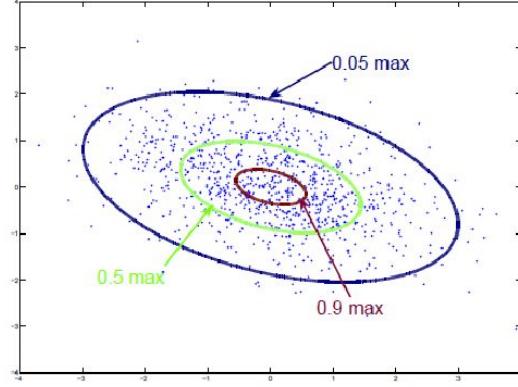
where $y_i = u_i^T (x - \mu)$

- Recall the expression for an ellipse in 2D: $\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1$

Intuition so far

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

- Falls off exponentially as a function of (squared) Euclidean distance to the mean $\|\mathbf{x} - \boldsymbol{\mu}\|^2$;
- the covariance matrix $\boldsymbol{\Sigma}$ determines the shape of the density;
- Determinant $|\boldsymbol{\Sigma}|$ measures the “spread” (analogous to σ^2).
- \mathcal{N} is the joint density of coordinates x_1, \dots, x_d .



Log-likelihood

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

- Take the log, for a single example \mathbf{x} :

$$\log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{d}{2} \log 2\pi - \frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

- Can ignore terms independent of parameters:

$$\log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) + \text{const}$$

Log-likelihood (contd)

$$\log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) + \text{const}$$

- Given a set \mathbf{X} of n i.i.d. vectors, we have

$$\log \mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{n}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) + \text{const}$$

- We are now ready to compute ML estimates for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

ML for parameters

$$\log \mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{n}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) + \text{const}$$

- To find ML estimate, we use the rule

$$\frac{\partial}{\partial \mathbf{a}} \mathbf{a}^T \mathbf{b} = \frac{\partial}{\partial \mathbf{a}} \mathbf{b}^T \mathbf{a} = \mathbf{b},$$

and set derivative w.r.t. $\boldsymbol{\mu}$ to zero:

$$\frac{\partial}{\partial \boldsymbol{\mu}} \log \mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^n \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) = 0,$$

which yields $\hat{\boldsymbol{\mu}}_{ML} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$.

Estimating the Covariance:

$$\begin{aligned}
\ell &= \sum_{i=1}^m \left(\frac{1}{2} \log |\Sigma^{-1}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right) \\
&= \frac{m}{2} \log |\Sigma^{-1}| - \frac{1}{2} \sum_{i=1}^m (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
&= \frac{m}{2} \log |\Sigma^{-1}| - \frac{1}{2} \sum_{i=1}^m \text{tr} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
&= \frac{m}{2} \log |\Sigma^{-1}| - \frac{1}{2} \sum_{i=1}^m \text{tr} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}
\end{aligned}$$

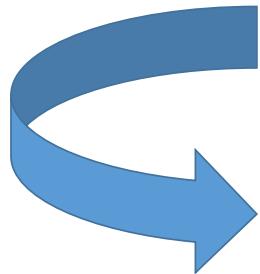
We've made the steps easy to follow.

- The first step comes from observing that the first term is independent of the summation variable, i
- The next step uses the trace trick: we can throw in the trace of a real number because the value is preserved.
- The next step uses the **cyclic permutation property** of the trace operator: $\text{tr } ABC = \text{tr } CAB$

$$\log \det X^{-1} = \log(\det X)^{-1} = -\log \det X$$

$$\frac{\partial}{\partial X_{ij}} \log \det X^{-1} = -\frac{\partial}{\partial X_{ij}} \log \det X = -\frac{1}{\det X} \frac{\partial \det X}{\partial X_{ij}} = -\frac{1}{\det X} \text{adj}(X)_{ji} = -(X^{-1})_{:}$$

Thus, $\nabla_A \log |A| = (A^{-1})^T$



$$\begin{aligned}
\text{Using... } \Lambda &= \Sigma^{-1} & \nabla_\Lambda \ell &= \nabla_\Lambda \left(\frac{m}{2} \log |\Lambda| - \frac{1}{2} \sum_{i=1}^m \text{tr} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \Lambda \right) \\
&&&= \frac{m}{2} \Lambda^{-1} - \frac{1}{2} \sum_{i=1}^m (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T = 0 \\
&\Rightarrow m \Lambda^{-1} &&= \sum_{i=1}^m (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \\
&\Rightarrow m \Sigma &&= \sum_{i=1}^m (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \\
&\Rightarrow \boxed{\Sigma = \frac{1}{m} \sum_{i=1}^m (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T}
\end{aligned}$$

ML for parameters (contd)

- A somewhat lengthier derivation produces ML estimate for the covariance:

$$\hat{\Sigma}_{ML} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T.$$

- Note: the $\boldsymbol{\mu}$ above is the ML estimate $\hat{\boldsymbol{\mu}}_{ML}$.
- Thus ML estimates for the mean is the *sample mean* of the data, and ML estimate for the covariance is the *sample covariance* of the data.

Generative Machine Learning:

Gaussian Discriminant Analysis (GDA)

Naïve Bayes Classification



Thomas Bayes
1702 - 1761

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:
$$P(C | A) = \frac{P(A, C)}{P(A)}$$

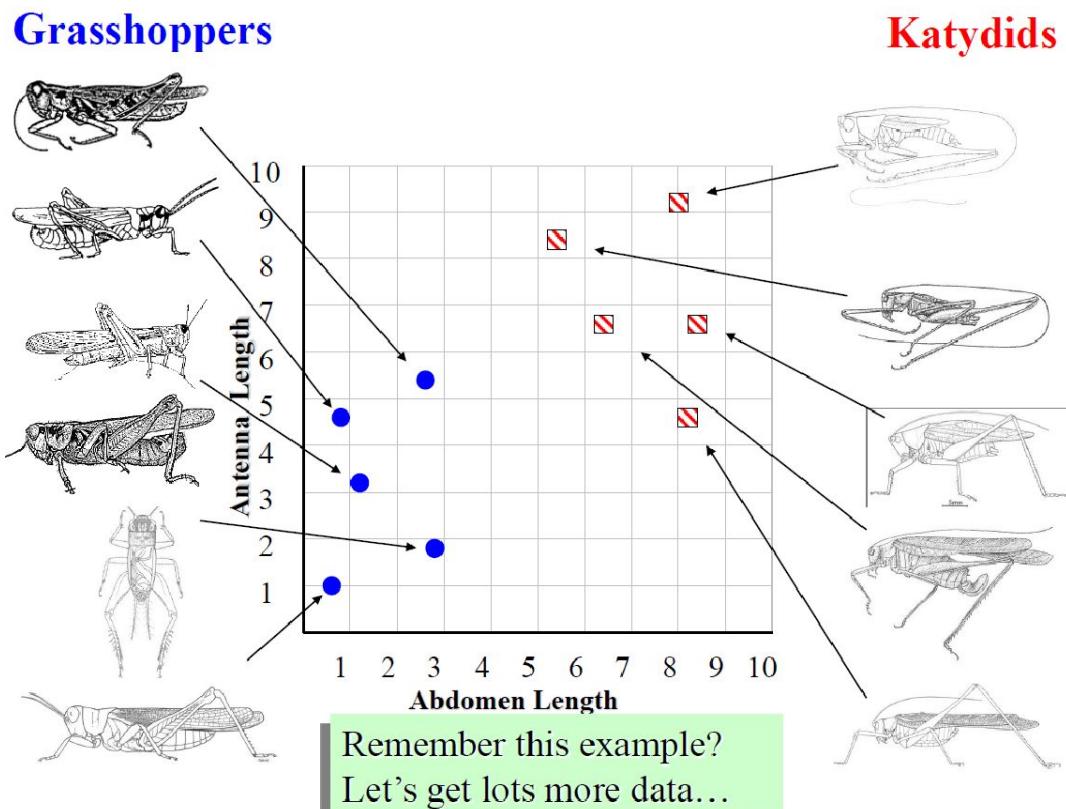
$$P(A | C) = \frac{P(A, C)}{P(C)}$$
- Bayes theorem:

$$P(C | A) = \frac{P(A | C)P(C)}{P(A)}$$

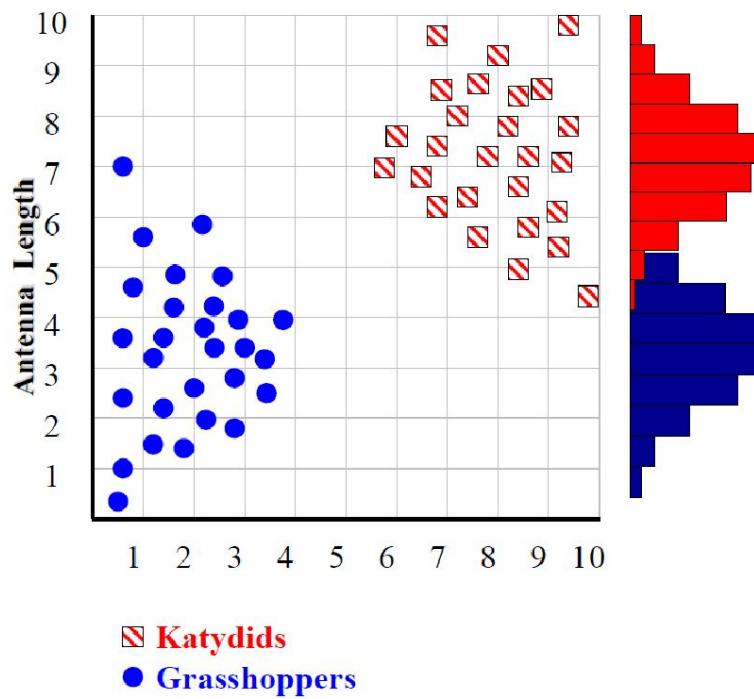
Example of Bayes Theorem

- Given:
 - A doctor knows that Cold causes fever 50% of the time
 - Prior probability of any patient having cold is 1/50,000
 - Prior probability of any patient having fever is 1/20
- If a patient has fever, what's the probability he/she has cold?

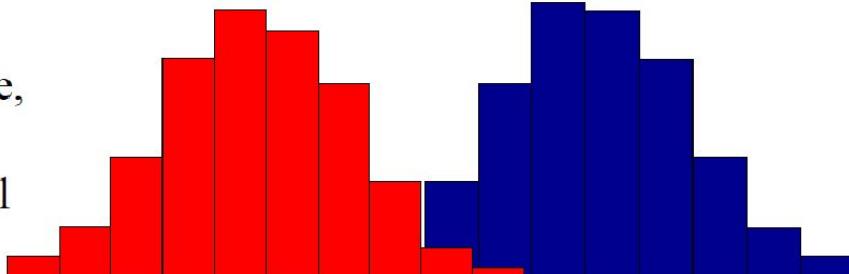
$$P(C | F) = \frac{P(F | C)P(C)}{P(F)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$



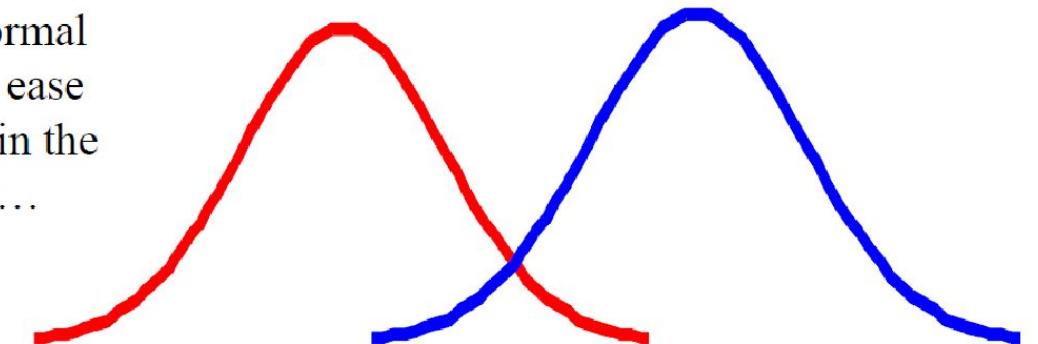
With a lot of data, we can build a histogram. Let us just build one for “Antenna Length” for now...



We can leave the histograms as they are, or we can summarize them with two normal distributions.

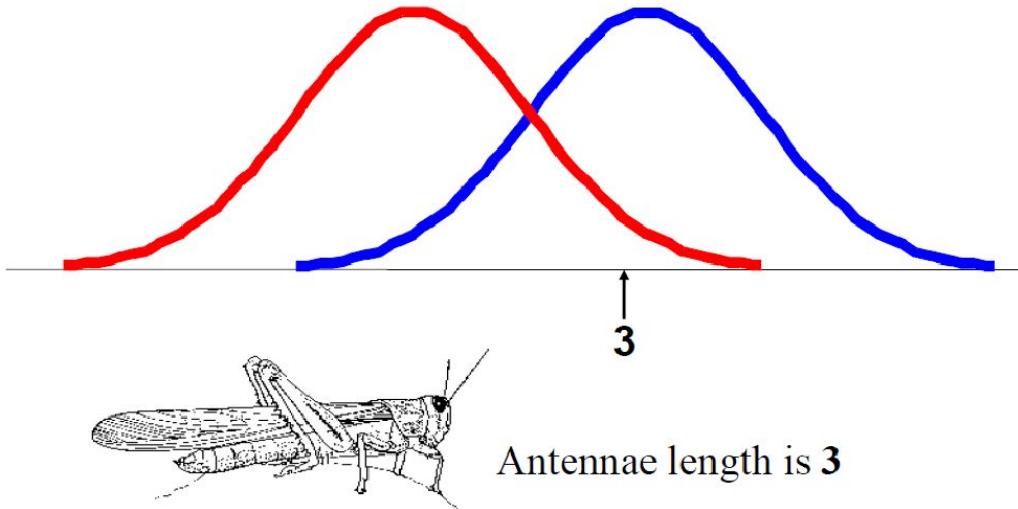


Let us use two normal distributions for ease of visualization in the following slides...



- We want to classify an insect we have found. Its antennae are 3 units long. How can we classify it?
- We can just ask ourselves, given the distributions of antennae lengths we have seen, is it more *probable* that our insect is a **Grasshopper** or a **Katydid**.
- There is a formal way to discuss the most *probable* classification...

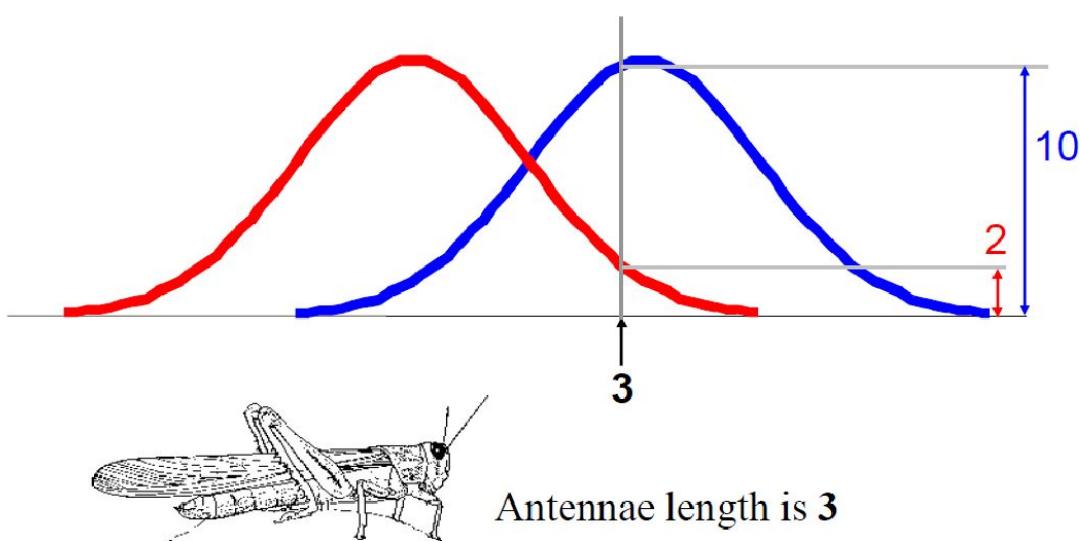
$p(c_j | d)$ = probability of class c_j , given that we have observed d



$p(c_j | d)$ = probability of class c_j , given that we have observed d

$$P(\text{Grasshopper} | 3) = 10 / (10 + 2) = 0.833$$

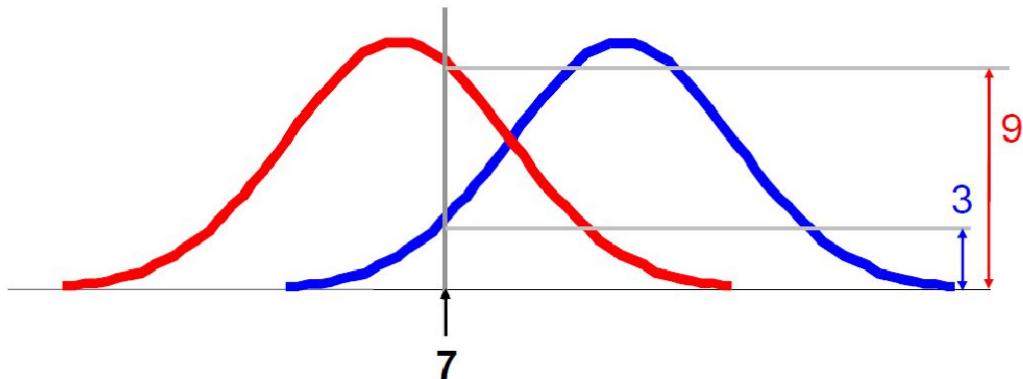
$$P(\text{Katydid} | 3) = 2 / (10 + 2) = 0.166$$



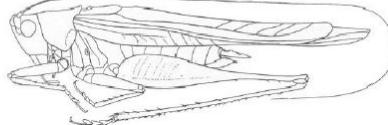
$p(c_j | d)$ = probability of class c_j , given that we have observed d

$$P(\text{Grasshopper} | 7) = 3 / (3 + 9) = 0.250$$

$$P(\text{Katydid} | 7) = 9 / (3 + 9) = 0.750$$



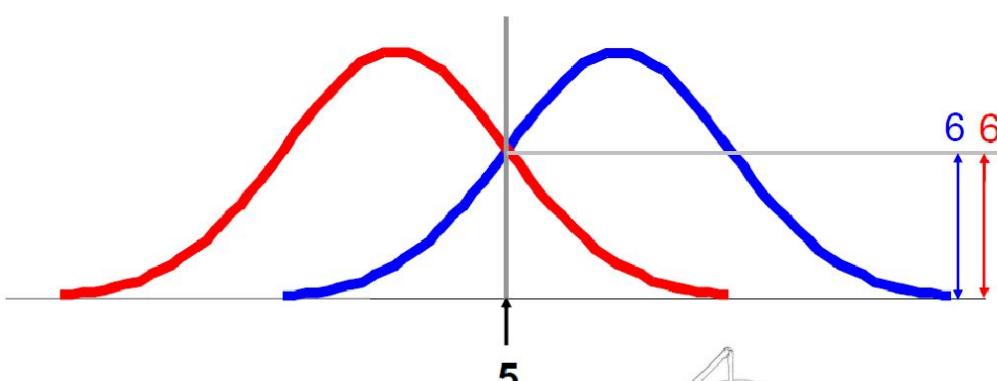
Antennae length is 7



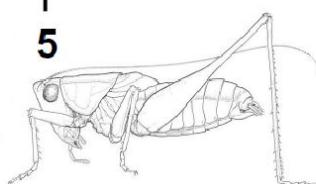
$p(c_j | d)$ = probability of class c_j , given that we have observed d

$$P(\text{Grasshopper} | 5) = 6 / (6 + 6) = 0.500$$

$$P(\text{Katydid} | 5) = 6 / (6 + 6) = 0.500$$



Antennae length is 5



Bayes Classifiers

That was a visual intuition for a simple case of the Bayes classifier, also called:

- Idiot Bayes
- Naïve Bayes
- Simple Bayes

We are about to see some of the mathematical formalisms, and more examples, but keep in mind the basic idea.

*Find out the probability of the **previously unseen instance** belonging to each class, then simply pick the most probable class.*

Bayes Classifiers

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j | d) = \frac{p(d | c_j) p(c_j)}{p(d)}$$

- $p(c_j | d)$ = probability of instance d being in class c_j ,
This is what we are trying to compute
- $p(d | c_j)$ = probability of generating instance d given class c_j ,
We can imagine that being in class c_j , causes you to have feature d with some probability
- $p(c_j)$ = probability of occurrence of class c_j ,
This is just how frequent the class c_j , is in our database
- $p(d)$ = probability of instance d occurring
This can actually be ignored, since it is the same for all classes

Assume that we have two classes

$c_1 = \text{male}$, and $c_2 = \text{female}$.

We have a person whose sex we do not know, say “*drew*” or *d*.

Classifying *drew* as male or female is equivalent to asking is it more probable that *drew* is **male** or **female**, I.e which is greater $p(\text{male} | \text{drew})$ or $p(\text{female} | \text{drew})$

(Note: ‘‘Drew can be a male or female name’’)



Drew Barrymore



Drew Carey

What is the probability of being called “*drew*” given that you are a **male**?

$$p(\text{male} | \text{drew}) = \frac{p(\text{drew} | \text{male}) p(\text{male})}{p(\text{drew})}$$

What is the probability of being a **male**?

What is the probability of being named “*drew*”?
(actually irrelevant, since it is that same for all classes)



Officer Drew

This is Officer Drew (who arrested me in 1997). Is Officer Drew a **Male** or **Female**?

Luckily, we have a small database with names and sex.

We can use it to apply Bayes rule...

$$p(c_j | d) = \frac{p(d | c_j) p(c_j)}{p(d)}$$

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male



$$p(c_j | d) = \frac{p(d | c_j) p(c_j)}{p(d)}$$

Officer Drew

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

$$p(\text{male} | \text{drew}) = \frac{1/3 * 3/8}{3/8} = 0.125$$
$$p(\text{female} | \text{drew}) = \frac{2/5 * 5/8}{3/8} = 0.250$$

Officer Drew is more likely to be a Female.



Officer Drew IS a female!

Officer Drew

$$p(\text{male} | \text{drew}) = \frac{1/3 * 3/8}{3/8} = 0.125$$

$$p(\text{female} | \text{drew}) = \frac{2/5 * 5/8}{3/8} = 0.250$$

So far we have only considered Bayes Classification when we have one attribute (the “*antennae length*”, or the “*name*”). But we may have many features.

How do we use all the features?

$$p(c_j | d) = \frac{p(d | c_j) p(c_j)}{p(d)}$$

Name	Over 170CM	Eye	Hair length	Sex
Drew	No	Blue	Short	Male
Claudia	Yes	Brown	Long	Female
Drew	No	Blue	Long	Female
Drew	No	Blue	Long	Female
Alberto	Yes	Brown	Short	Male
Karin	No	Blue	Long	Female
Nina	Yes	Brown	Short	Female
Sergio	Yes	Blue	Long	Male

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \dots * p(d_n|c_j)$$

The probability of class c_j generating instance d , equals....

The probability of class c_j generating the observed value for feature 1, multiplied by..

The probability of class c_j generating the observed value for feature 2, multiplied by..

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \dots * p(d_n|c_j)$$

$$p(\text{officer drew}|c_j) = p(\text{over_170}_\text{cm} = \text{yes}|c_j) * p(\text{eye} = \text{blue}|c_j) * \dots$$



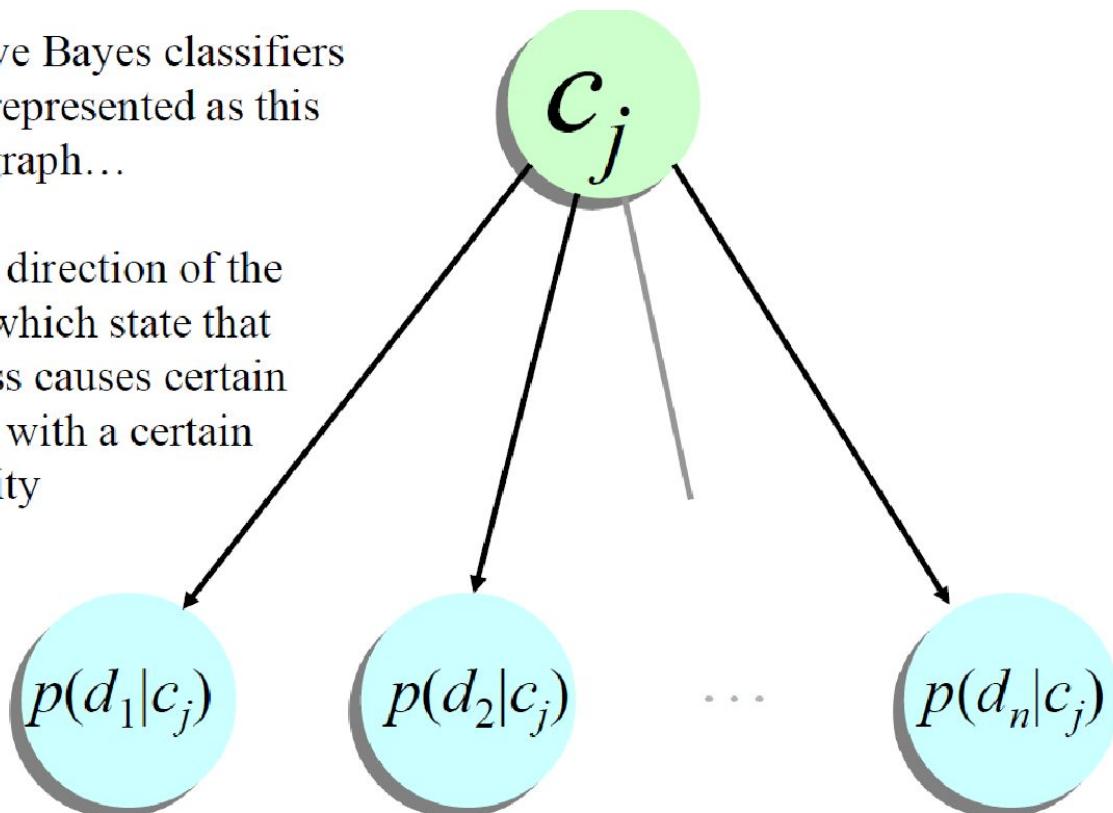
Officer Drew
is blue-eyed,
over 170_{cm}
tall, and has
long hair

$$p(\text{officer drew} | \text{Female}) = 2/5 * 3/5 * \dots$$

$$p(\text{officer drew} | \text{Male}) = 2/3 * 2/3 * \dots$$

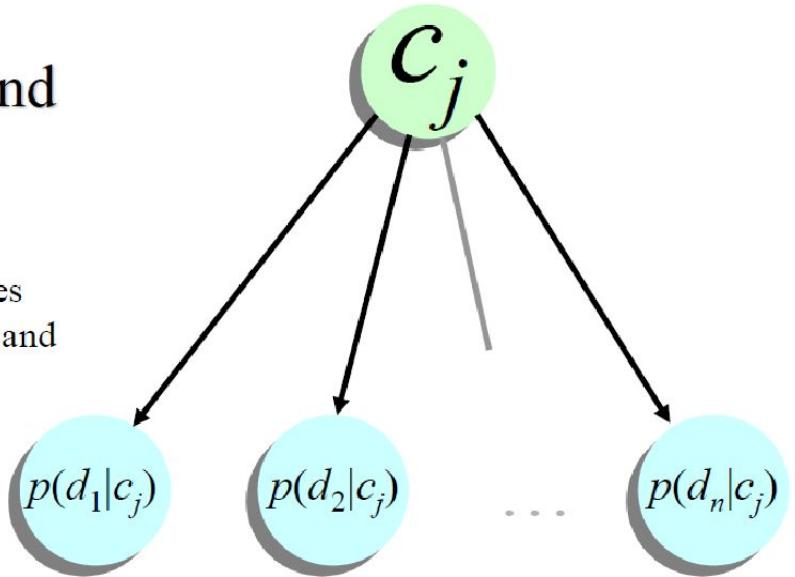
The Naive Bayes classifiers
is often represented as this
type of graph...

Note the direction of the
arrows, which state that
each class causes certain
features, with a certain
probability



Naïve Bayes is fast and space efficient

We can look up all the probabilities with a single scan of the database and store them in a (small) table...



Sex	Over190 _{cm}	
Male	Yes	0.15
	No	0.85
Female	Yes	0.01
	No	0.99

Sex	Long Hair	
Male	Yes	0.05
	No	0.95
Female	Yes	0.70
	No	0.30

Sex		
Male		
Female		

How to estimate Probability from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c / N$
 - e.g., $P(\text{No}) = 7/10$, $P(\text{Yes}) = 3/10$
- For discrete attributes:
$$P(A_i | C_k) = |A_{ik}| / N_c$$
 - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
 - Examples:
$$P(\text{Status=Married} | \text{No}) = 4/7$$

$$P(\text{Refund=Yes} | \text{Yes}) = 0$$

How to estimate Probability from Data?

- For continuous attributes:
 - Discretize the range into bins
 - one ordinal attribute per bin
 - violates independence assumption
 - Two-way split: $(A < v)$ or $(A > v)$
 - choose only one of the two splits as new attribute
 - Probability density estimation:
 - Assume attribute follows a normal distribution
 - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
 - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i | c)$

How to estimate Probability from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:
- $$P(A_i | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$
- One for each (A_i, c_j) pair
- For (Income, Class=No):
 - If Class=No
 - sample mean = 110
 - sample variance = 2975

$$P(\text{Income}=120 | \text{No}) = \frac{1}{\sqrt{2\pi(54.54)}} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a new test record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

$P(\text{Refund}=\text{Yes}|\text{No}) = 3/7$
 $P(\text{Refund}=\text{No}|\text{No}) = 4/7$
 $P(\text{Refund}=\text{Yes}|\text{Yes}) = 0$
 $P(\text{Refund}=\text{No}|\text{Yes}) = 1$
 $P(\text{Marital Status}=\text{Single}|\text{No}) = 2/7$
 $P(\text{Marital Status}=\text{Divorced}|\text{No}) = 1/7$
 $P(\text{Marital Status}=\text{Married}|\text{No}) = 4/7$
 $P(\text{Marital Status}=\text{Single}|\text{Yes}) = 2/7$
 $P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = 1/7$
 $P(\text{Marital Status}=\text{Married}|\text{Yes}) = 0$

For taxable income:

If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \times P(\text{Married}|\text{Class}=\text{No}) \times P(\text{Income}=120\text{K}|\text{Class}=\text{No}) = 4/7 \times 4/7 \times 0.0072 = 0.0024$
- $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \times P(\text{Married}|\text{Class}=\text{Yes}) \times P(\text{Income}=120\text{K}|\text{Class}=\text{Yes}) = 1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore $P(\text{No}|X) > P(\text{Yes}|X)$
=> Class = No

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original : } P(A_i | C) = \frac{N_{ic}}{N_c}$$

c: number of classes

$$\text{Laplace : } P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$$

p: prior probability

$$\text{m - estimate : } P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

m: parameter

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

$$P(A|M)P(M) > P(A|N)P(N)$$

=> Mammals

Naïve Bayes is NOT sensitive to irrelevant features...

Suppose we are trying to classify a persons sex based on several features, including eye color. (Of course, eye color is completely irrelevant to a persons gender)

$$p(\text{Jessica} | c_j) = p(\text{eye} = \text{brown} | c_j) * p(\text{wears_dress} = \text{yes} | c_j) * \dots$$

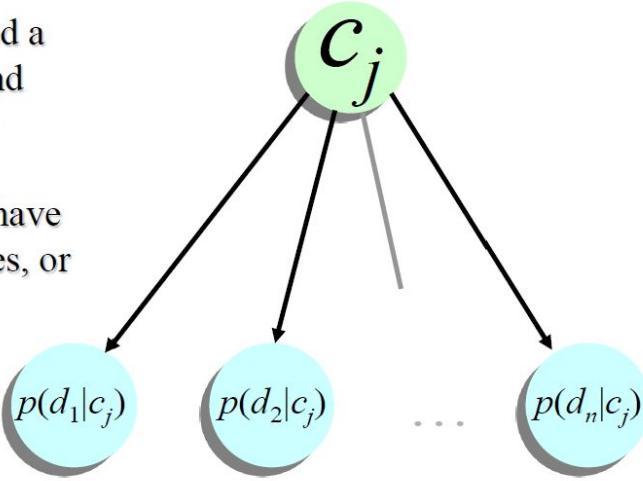
$$p(\text{Jessica} | \text{Female}) = 9,000/10,000 * 9,975/10,000 * \dots$$

$$p(\text{Jessica} | \text{Male}) = 9,001/10,000 * 2/10,000 * \dots$$

Almost the same!

However, this assumes that we have good enough estimates of the probabilities, so the more data the better.

An obvious point. I have used a simple two class problem, and two possible values for each example, for my previous examples. However we can have an arbitrary number of classes, or feature values



Animal	Mass >10 _{kg}	
Cat	Yes	0.15
	No	0.85
Dog	Yes	0.91
	No	0.09
Pig	Yes	0.99
	No	0.01

Animal	Color	
Cat	Black	0.33
	White	0.23
Dog	Brown	0.44
	Black	0.97
Pig	White	0.03
	Brown	0.90
Pig	Black	0.04
	White	0.01

Animal
Cat
Dog
Pig

Advantages/Disadvantages of Naïve Bayes

- Advantages:
 - Fast to train (single scan). Fast to classify
 - Not sensitive to irrelevant features
 - Handles real and discrete data
 - Handles streaming data well
- Disadvantages:
 - Assumes independence of features

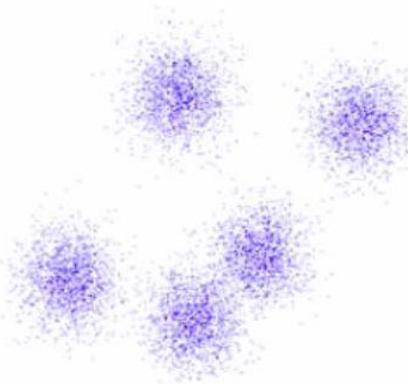
Spectral Clustering

Introduction

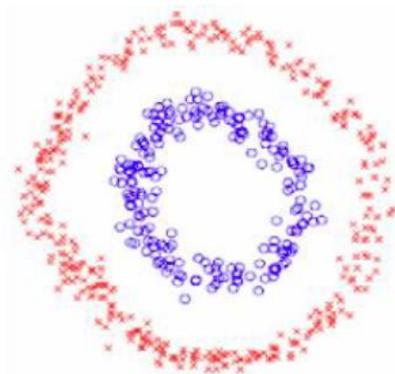
Several slides are credited to Prof. Aarti Singh, ML 10 – 701, CMU

Data Clustering

- Two different criteria
 - Compactness, e.g., k-means, mixture models
 - Connectivity, e.g., spectral clustering



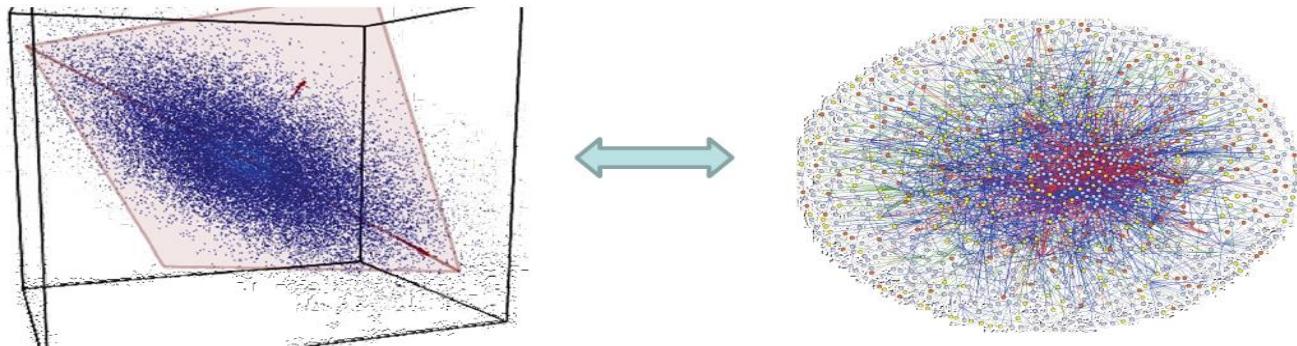
Compactness



Connectivity

Connect Points in R^d and Graph Views of Data

- Points in R^d
 - via near-neighbor graphs
- Graph
 - via matrix representations of graphs



Graph Clustering

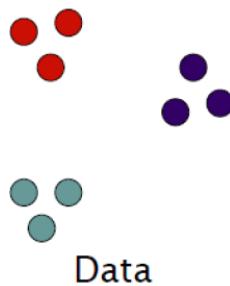
Goal: Given data points X_1, \dots, X_n and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

Similarity Graph: $G(V, E, W)$

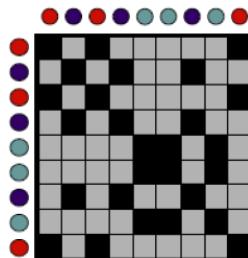
V – Vertices (Data points)

E – Edge if similarity > 0

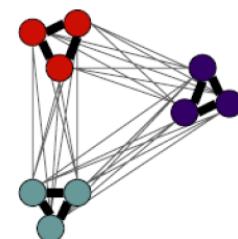
W - Edge weights (similarities)



Data



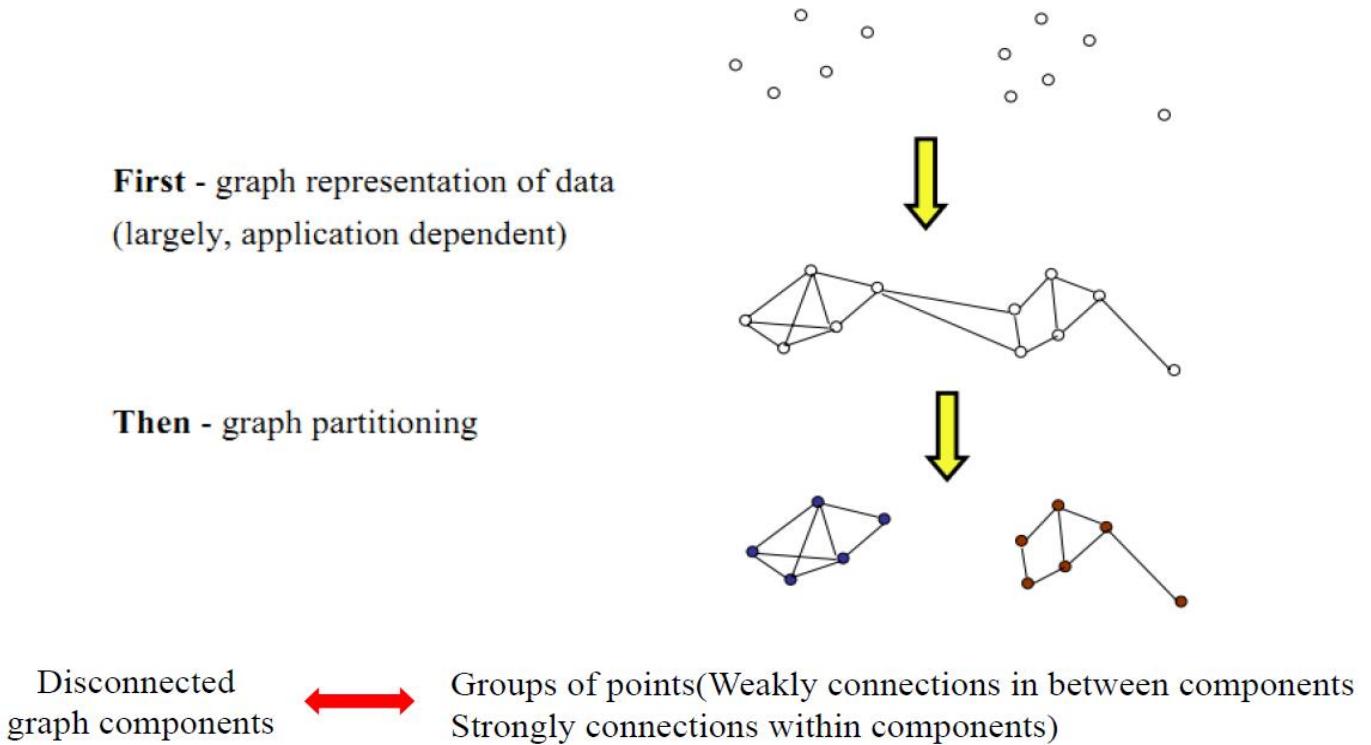
Similarities



Similarity graph

Partition the graph so that edges within a group have large weights and edges across groups have small weights.

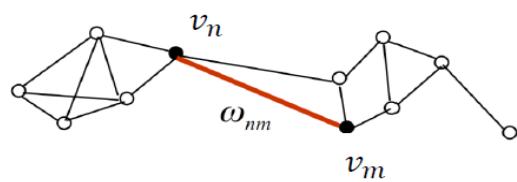
GENERAL



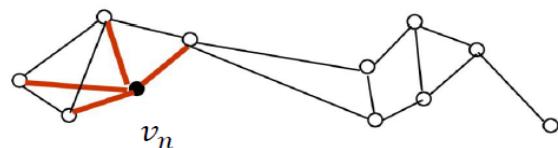
GRAPH NOTATION

$G = (V, E) :$

- Vertex set $V = \{v_1, \dots, v_n\}$
- Weighted adjacency matrix $W = (w_{ij}) \quad i, j = 1, \dots, n \quad w_{ij} \geq 0$



- Degree $d_i = \sum_{j=1}^n w_{ij}$



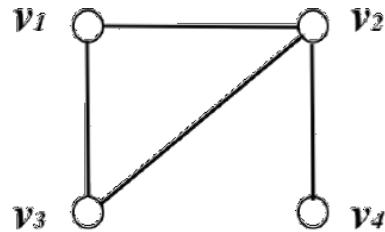
- Degree matrix Diagonal matrix with the degrees d_1, \dots, d_n on the diagonal.

Adjacency Matrices

- For a graph with n vertices, the entries of the $n \times n$ adjacency matrix are defined by:

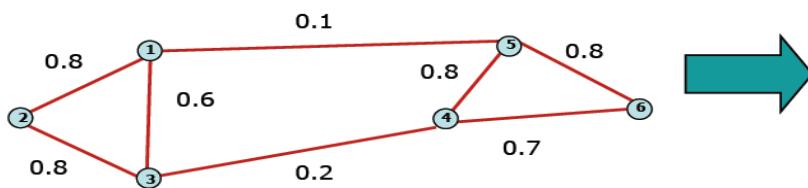
$$\mathbf{A} := \begin{cases} A_{ij} = 1 & \text{if there is an edge } e_{ij} \\ A_{ij} = 0 & \text{if there is no edge} \\ A_{ii} = 0 \end{cases}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



Weighted Matrices

- Adjacency matrix (A)
 - $n \times n$ matrix
 - $A = [w_{ij}]$: edge weight between vertex x_i and x_j



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

- Important properties:
 - Symmetric matrix
 - ⇒ Eigenvalues are real
 - ⇒ Eigenvector could span orthogonal base

Eigenvalues and Eigenvectors

- \mathbf{A} is a real-symmetric matrix: it has n real eigenvalues and its n real eigenvectors form an orthonormal basis.
- Let $\{\lambda_1, \dots, \lambda_i, \dots, \lambda_r\}$ be the set of *distinct* eigenvalues.
- The eigenspace S_i contains the eigenvectors associated with λ_i :

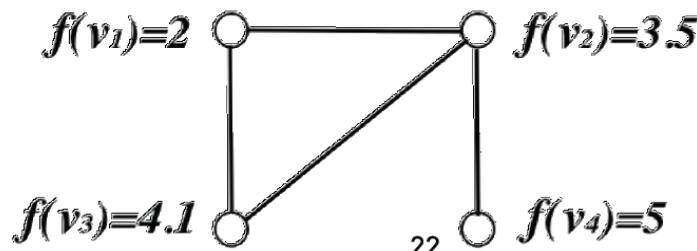
$$S_i = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \lambda_i \mathbf{x}\}$$

- For real-symmetric matrices, the algebraic multiplicity is equal to the geometric multiplicity, for all the eigenvalues.
- The dimension of S_i (geometric multiplicity) is equal to the multiplicity of λ_i .
- If $\lambda_i \neq \lambda_j$ then S_i and S_j are mutually orthogonal.

Order the eigenvalues from small to large

Functions on Graphs

- We consider real-valued functions on the set of the graph's vertices, $f : \mathcal{V} \rightarrow \mathbb{R}$. Such a function assigns a real number to each graph node.
- f is a vector indexed by the graph's vertices, hence $f \in \mathbb{R}^n$.
- Notation: $f = (f(v_1), \dots, f(v_n)) = (f(1), \dots, f(n))$.
- The eigenvectors of the adjacency matrix, $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, can be viewed as *eigenfunctions*.



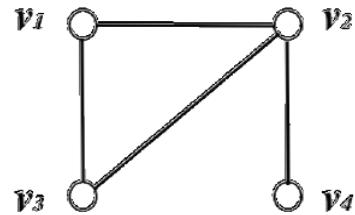
Graph (Unnormalized) Laplacian

- $\mathbf{L} = \nabla^T \nabla$
- $(\mathbf{L}\mathbf{f})(v_i) = \sum_{v_j \sim v_i} (f(v_i) - f(v_j))$
- Connection between the Laplacian and the adjacency matrices:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

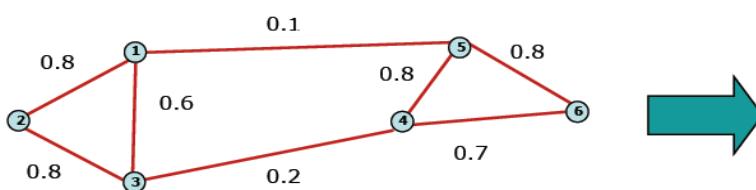
- The degree matrix: $\mathbf{D} := D_{ii} = d(v_i)$.

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$



Degree Matrix

- **Degree matrix (\mathbf{D})**
 - $n \times n$ diagonal matrix
 - $D(i,i) = \sum_j w_{ij}$: total weight of edges incident to vertex x_i

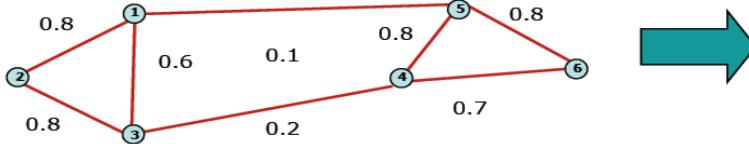


	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	0	0	0	0	0
x_2	0	1.6	0	0	0	0
x_3	0	0	1.6	0	0	0
x_4	0	0	0	1.7	0	0
x_5	0	0	0	0	1.7	0
x_6	0	0	0	0	0	1.5

- Important application:
 - Normalize adjacency matrix

Laplacian Matrix

- **Laplacian matrix (L)**
 - $n \times n$ symmetric matrix



$$L = D - A$$

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

- **Important properties:**
 - Eigenvalues are non-negative real numbers (Gershgorin circle theorem)
 - Eigenvectors are real and orthogonal
 - Eigenvalues and eigenvectors provide an insight into the connectivity of the graph...

Undirected Weighted Graphs

- We consider *undirected weighted graphs*: Each edge e_{ij} is weighted by $w_{ij} > 0$.
- The Laplacian as an operator:

$$(\mathbf{L}\mathbf{f})(v_i) = \sum_{v_j \sim v_i} w_{ij}(f(v_i) - f(v_j))$$

- As a quadratic form:

$$\mathbf{f}^\top \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{e_{ij}} w_{ij}(f(v_i) - f(v_j))^2$$

- \mathbf{L} is symmetric and positive semi-definite.
- \mathbf{L} has n non-negative, real-valued eigenvalues:
 $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

- Unnormalized Graph Laplacian

$$d_i = \sum_{j=1}^n w_{ij}$$

$$L = D - W$$

Proposition 1 (Properties of L) The matrix L satisfies the following properties:

1. For every $f \in \mathbb{R}^n$ we have

$$f'Lf = \frac{1}{2} \sum_{i,j=1}^n w_{ij}^2 (f_i - f_j)^2$$

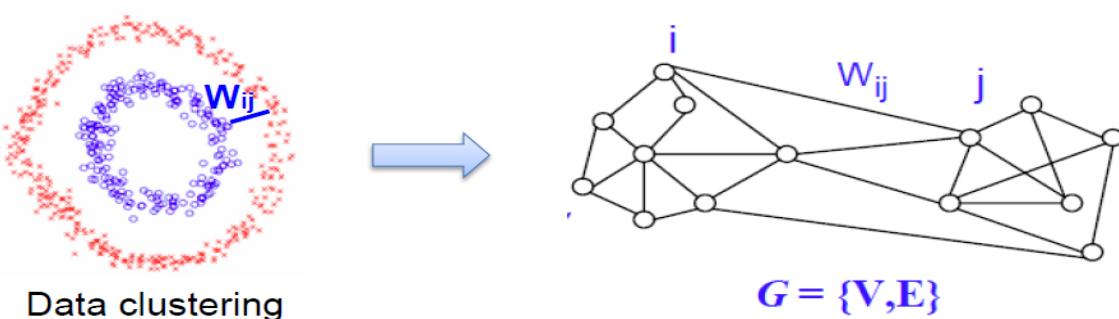
$$\begin{aligned} f'Lf &= f'Df - f'Wf = \sum_{i=1}^n d_i f_i^2 - \sum_{i,j=1}^n f_i f_j w_{ij} \\ &= \frac{1}{2} \left(\sum_{i=1}^n d_i f_i^2 - 2 \sum_{i,j=1}^n f_i f_j w_{ij} + \sum_{j=1}^n d_j f_j^2 \right) = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2 \end{aligned}$$

Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

E.g. Gaussian kernel similarity function

$$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}} \longrightarrow \text{Controls size of neighborhood}$$



SIMILARITY GRAPH

- ε -neighborhood graph

Connect all points whose pairwise distances are smaller than ε

- k -nearest neighbor graph

Connect vertex v_i with vertex v_j if v_j is among the k -nearest neighbors of v_i .

- fully connected graph

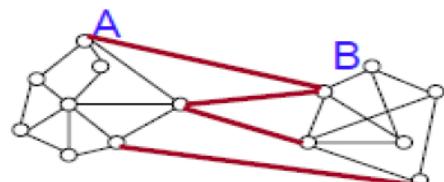
Connect all points with positive similarity with each other

All the above graphs are regularly used in spectral clustering!

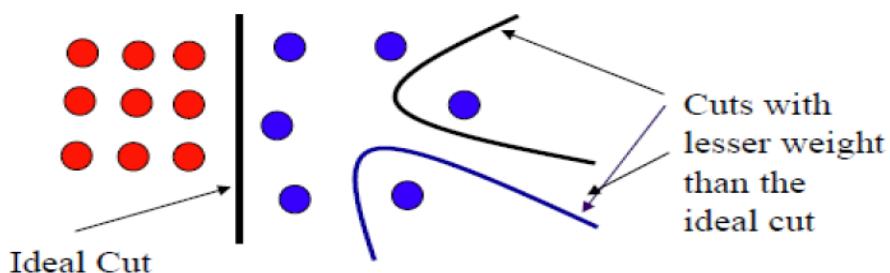
Partitioning a graph into two clusters

Min-cut: Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum.

$$\text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij}$$



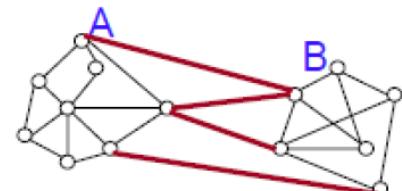
- Easy to solve $O(VE)$ algorithm
- Not satisfactory partition – often isolates vertices



Partitioning a graph into two clusters

Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum & size of A and B are very similar.

$$\text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij}$$



Normalized cut:

$$\text{Ncut}(A, B) := \text{cut}(A, B) \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right)$$

$$\text{vol}(A) = \sum_{i \in A} d_i$$

But NP-hard to solve!!

Spectral clustering is a relaxation of these.

Normalized Cut and Graph Laplacian

$$\text{Ncut}(A, B) := \text{cut}(A, B) \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right)$$

$$\text{Let } \mathbf{f} = [f_1 \ f_2 \ \dots \ f_n]^T \text{ with } f_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\text{vol}(B)} & \text{if } i \in B \end{cases}$$

$$\mathbf{f}^T \mathbf{L} \mathbf{f} = \sum_{ij} w_{ij} (f_i - f_j)^2 = \sum_{i \in A, j \in B} w_{ij} \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right)^2$$

$$\mathbf{f}^T \mathbf{D} \mathbf{f} = \sum_j d_j f_j^2 = \sum_{i \in A} \frac{d_i}{\text{vol}(A)^2} + \sum_{j \in B} \frac{d_j}{\text{vol}(B)^2} = \frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}$$

$$\text{Ncut}(A, B) = \frac{\mathbf{f}^T \mathbf{L} \mathbf{f}}{\mathbf{f}^T \mathbf{D} \mathbf{f}}$$

Normalized Cut and Graph Laplacian

$$\min \text{Ncut}(A, B) = \min \frac{\mathbf{f}^T \mathbf{L} \mathbf{f}}{\mathbf{f}^T \mathbf{D} \mathbf{f}}$$

where $\mathbf{f} = [f_1 \ f_2 \ \dots \ f_n]^T$ with $f_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\text{vol}(B)} & \text{if } i \in B \end{cases}$

Relaxation: $\min \frac{\mathbf{f}^T \mathbf{L} \mathbf{f}}{\mathbf{f}^T \mathbf{D} \mathbf{f}}$ s.t. $\mathbf{f}^T \mathbf{D} \mathbf{1} = 0$

Solution: \mathbf{f} – second eigenvector of generalized eval problem

$$\mathbf{L} \mathbf{f} = \lambda \mathbf{D} \mathbf{f}$$

Obtain cluster assignments by thresholding \mathbf{f} at 0

Approximation of Normalized cut

$$\text{Ncut}(A, B) := \text{cut}(A, B) \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right)$$

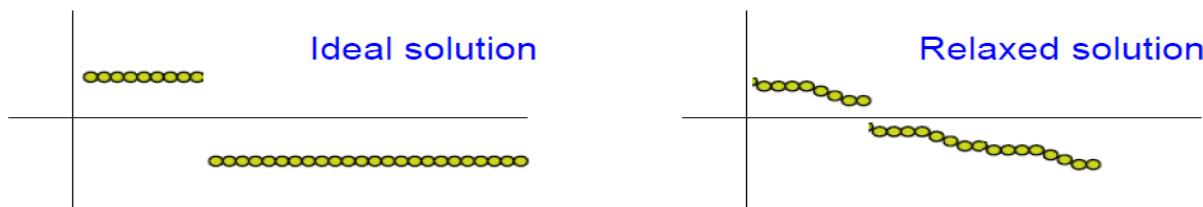
Let \mathbf{f} be the eigenvector corresponding to the second smallest eval of the generalized eval problem.

$$\mathbf{L} \mathbf{f} = \lambda \mathbf{D} \mathbf{f}$$

Equivalent to eigenvector corresponding to the second smallest eval of the normalized Laplacian $L' = D^{-1}L = I - D^{-1}W$

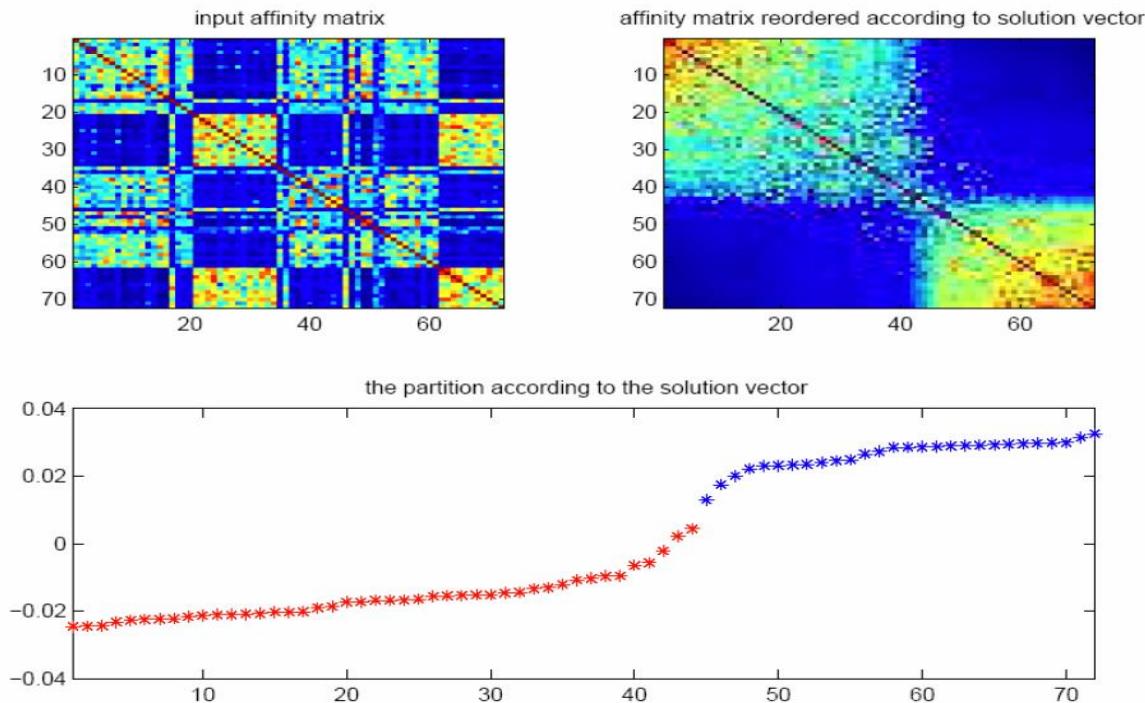
Recover binary partition as follows:

$$\begin{array}{lll} i \in A & \text{if} & f_i \geq 0 \\ i \in B & \text{if} & f_i < 0 \end{array}$$



Example

Xing et al 2001



How to partition a graph into k clusters?

Spectral Clustering Algorithm

Input: Similarity matrix W , number k of clusters to construct

- Build similarity graph
- Compute the first k eigenvectors v_1, \dots, v_k of the matrix

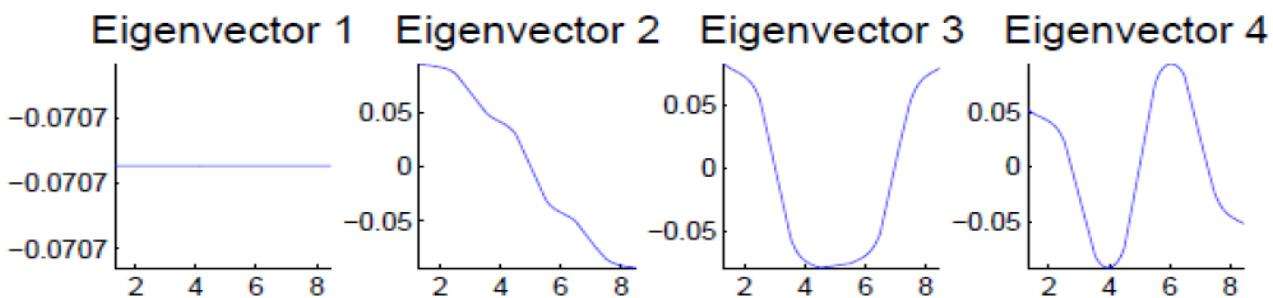
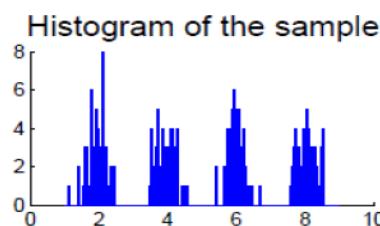
$$\begin{cases} L & \text{for unnormalized spectral clustering} \\ L' & \text{for normalized spectral clustering} \end{cases}$$

- Build the matrix $V \in \mathbb{R}^{n \times k}$ with the eigenvectors as columns
- Interpret the rows of V as new data points $Z_i \in \mathbb{R}^k$

	v_1	v_2	v_3		
Z_1	v_{11}	v_{12}	v_{13}	Dimensionality Reduction	
\vdots	\vdots	\vdots	\vdots	$n \times n \rightarrow n \times k$	
Z_n	v_{n1}	v_{n2}	v_{n3}		

- Cluster the points Z_i with the k -means algorithm in \mathbb{R}^k .

Eigenvectors of Graph Laplacian

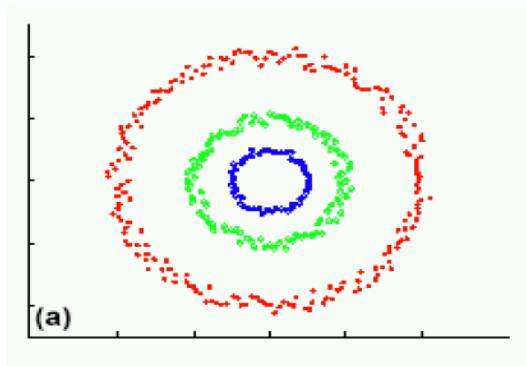


- 1st Eigenvector is the all ones vector **1** (if graph is connected)
- 2nd Eigenvector thresholded at 0 separates first two clusters from last two
- k-means clustering of the 4 eigenvectors identifies all clusters

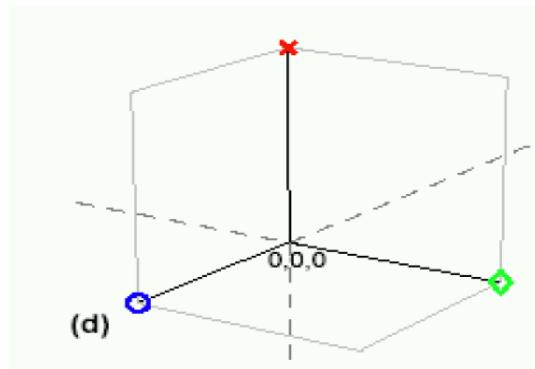
Why does it work?

Data are projected into a lower-dimensional space (the spectral/eigenvector domain) where they are easily separable, say using k-means.

Original data



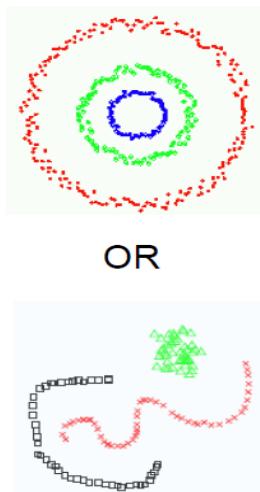
Projected data



Graph has 3 connected components – first three eigenvectors are constant (all ones) on each component.

Understanding Spectral Clustering

- If graph is connected, first Laplacian evec is constant (all 1s)
- If graph is disconnected (k connected components), Laplacian is block diagonal and first k Laplacian evecs are:



$$L = \begin{bmatrix} L_1 & & & \\ & \ddots & 0 & \\ & & L_2 & \\ & & & \ddots \\ 0 & & & L_3 \end{bmatrix}$$

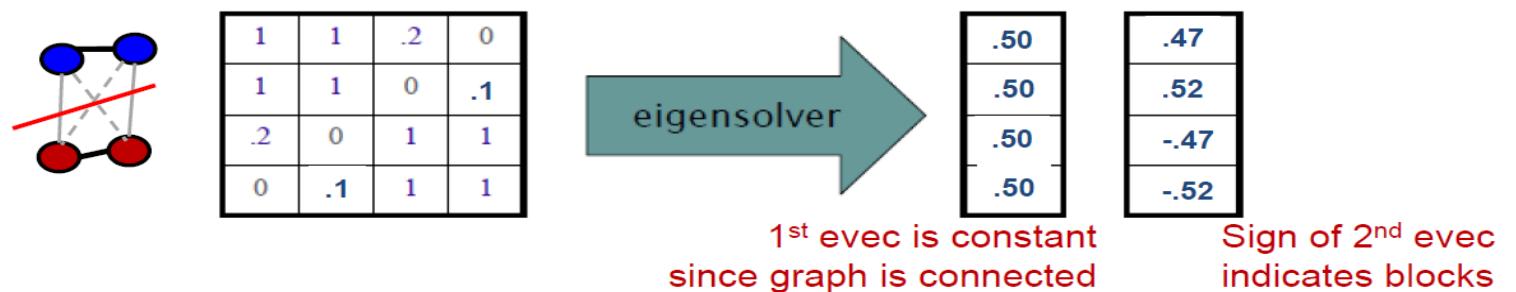
$$\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{array}$$

First three eigenvectors

Understanding Spectral Clustering

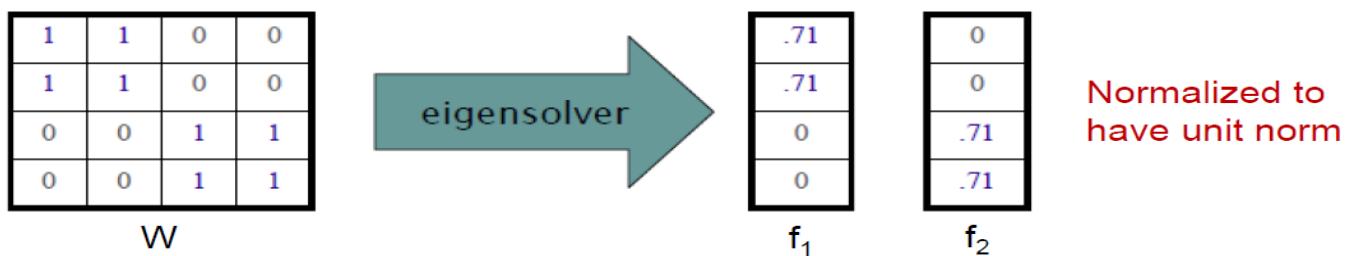
- Is all hope lost if clusters don't correspond to connected components of graph? No!
- If clusters are connected loosely (small off-block diagonal entries), then 1st Laplacian eigenvalue is all 1s, but second eigenvector gets first cut (min normalized cut)

$$\text{Ncut}(A, B) := \text{cut}(A, B) \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right)$$

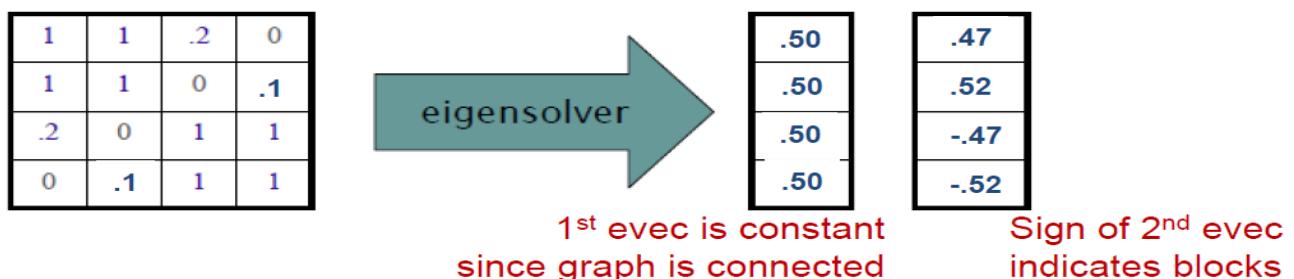


Why does it work?

Block weight matrix (disconnected graph) results in block eigenvectors:



Slight perturbation does not change span of eigenvectors significantly:



Why does it work?

Can put data points into blocks using eigenvectors:

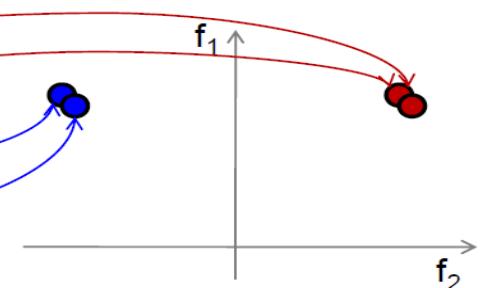
1	1	.2	0
1	1	0	.1
.2	0	1	1
0	.1	1	1

W

.50	.47
.50	.52
.50	-.47
.50	-.52

f_1

f_2



Embedding is same regardless of data ordering:

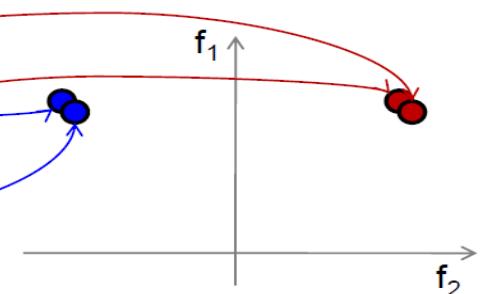
1	.2	1	0
.2	0	1	1
1	1	0	.1
0	1	.1	1

W

.50	.47
.50	-.47
.50	.52
.50	-.52

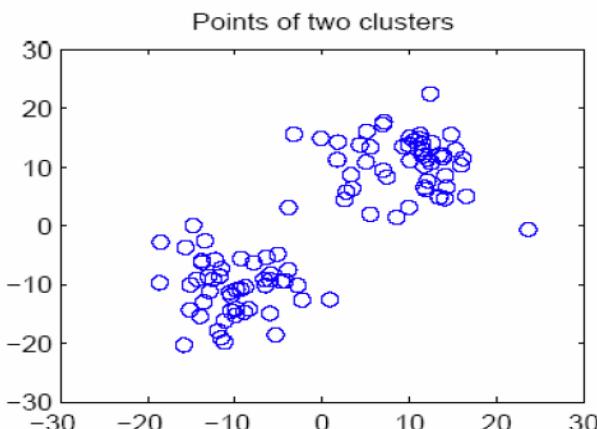
f_1

f_2

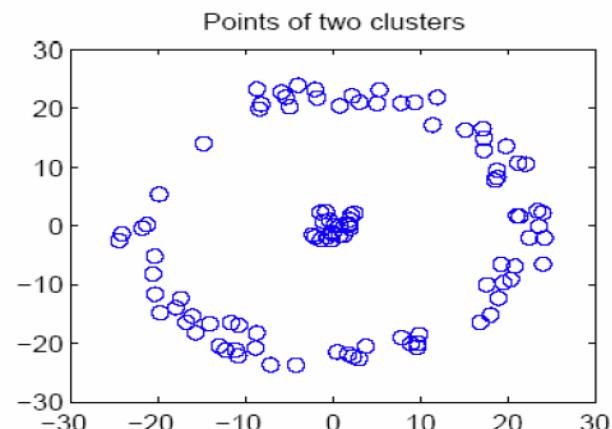


k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to **find cluster with non-convex boundaries**.



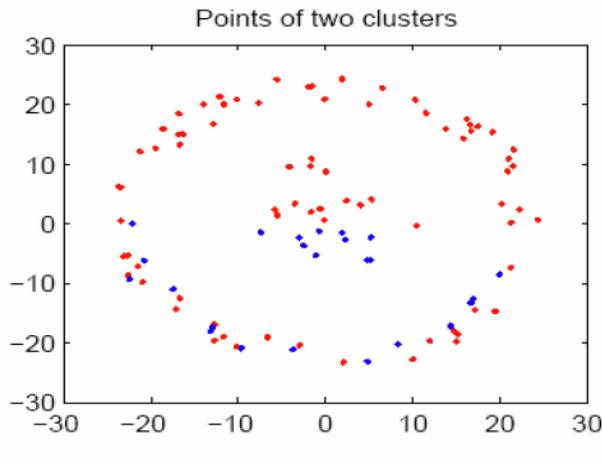
Both perform same



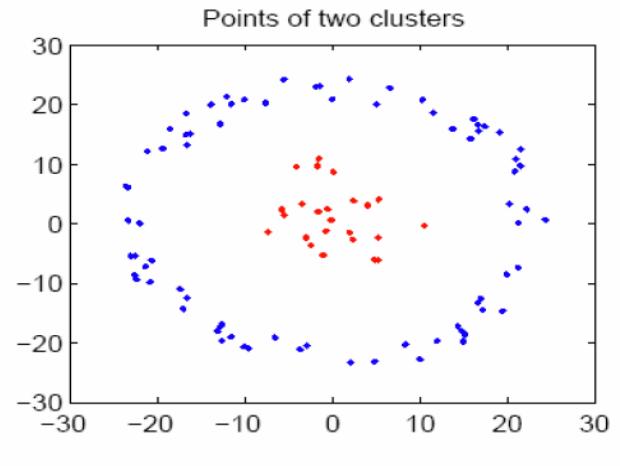
Spectral clustering is superior

k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to **find cluster with non-convex boundaries**.



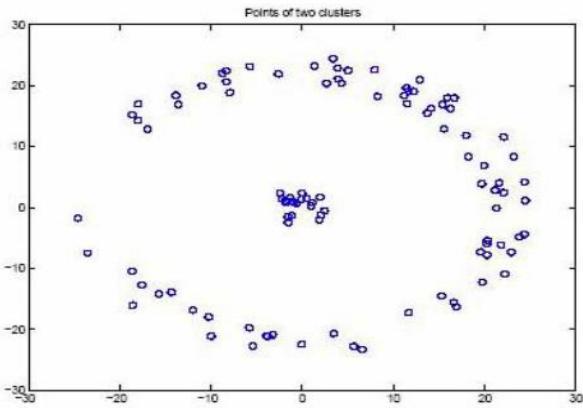
k-means output



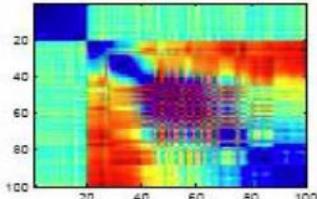
Spectral clustering output

k-means vs Spectral clustering

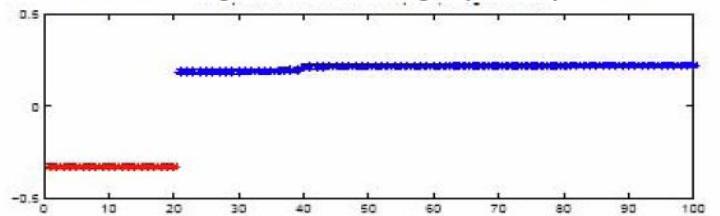
Applying k-means to laplacian eigenvectors allows us to **find cluster with non-convex boundaries**.



Similarity matrix

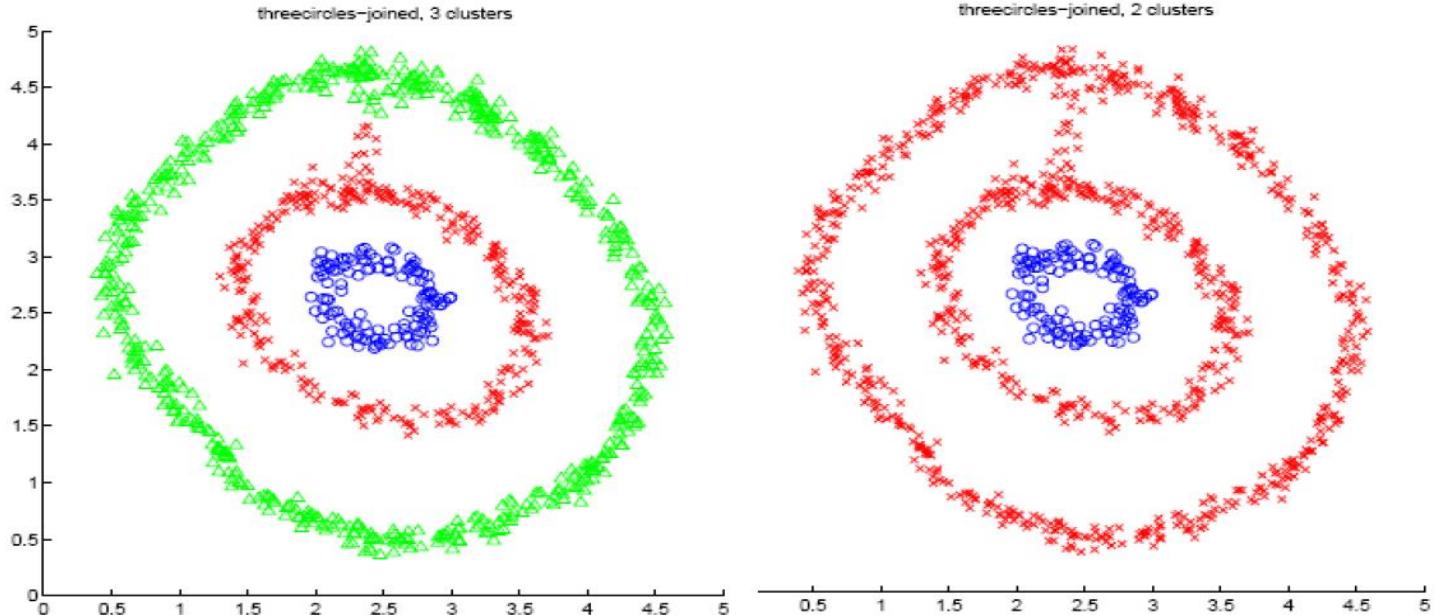


Second eigenvector of graph Laplacian



Examples (Choice of k)

Ng et al 2001



Spectral clustering summary

- Algorithms that cluster points using eigenvectors of matrices derived from the data
- Useful in hard non-convex clustering problems
- Obtain data representation in the low-dimensional space that can be easily clustered
- Variety of methods that use eigenvectors of unnormalized or normalized Laplacian, differ in how to derive clusters from eigenvectors, k-way vs repeated 2-way
- Empirically very successful

Connected Graph Laplacians

- $\mathbf{L}\mathbf{u} = \lambda\mathbf{u}$.
- $\mathbf{L}\mathbf{1}_n = \mathbf{0}$, $\lambda_1 = 0$ is the smallest eigenvalue.
- The *one* vector: $\mathbf{1}_n = (1 \dots 1)^\top$.
- $0 = \mathbf{u}^\top \mathbf{L}\mathbf{u} = \sum_{i,j=1}^n w_{ij}(u(i) - u(j))^2$.
- If any two vertices are connected by a path, then $\mathbf{u} = (u(1), \dots, u(n))$ needs to be constant at all vertices such that the quadratic form vanishes. Therefore, a graph with one connected component has the constant vector $\mathbf{u}_1 = \mathbf{1}_n$ as the only eigenvector with eigenvalue 0.

A Graph with k Connected Components

- Each connected component has an associated Laplacian.
Therefore, we can write matrix \mathbf{L} as a *block diagonal matrix*:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & \\ & \ddots & \\ & & \mathbf{L}_k \end{bmatrix}$$

- The spectrum of \mathbf{L} is given by the union of the spectra of \mathbf{L}_i .
- Each block corresponds to a connected component, hence each matrix \mathbf{L}_i has an eigenvalue 0 with multiplicity 1.
- The spectrum of \mathbf{L} is given by the union of the spectra of \mathbf{L}_i .
- The eigenvalue $\lambda_1 = 0$ has multiplicity k .

The Eigenspace of $\lambda_1 = 0$

- The eigenspace corresponding to $\lambda_1 = \dots = \lambda_k = 0$ is spanned by the k mutually orthogonal vectors:

$$\mathbf{u}_1 = \mathbf{1}_{L_1}$$

...

$$\mathbf{u}_k = \mathbf{1}_{L_k}$$

- with $\mathbf{1}_{L_i} = (0000111110000)^\top \in \mathbb{R}^n$
- These vectors are the *indicator vectors* of the graph's connected components.
- Notice that $\mathbf{1}_{L_1} + \dots + \mathbf{1}_{L_k} = \mathbf{1}_n$

The Fiedler Vector

- The first non-null eigenvalue λ_{k+1} is called the Fiedler value.
- The corresponding eigenvector \mathbf{u}_{k+1} is called the Fiedler vector.
- The multiplicity of the Fiedler eigenvalue is always equal to 1.
- The Fiedler value is the *algebraic connectivity of a graph*, the further from 0, the more connected.
- The Fiedler vector has been extensively used for *spectral bi-partitioning*
- Theoretical results are summarized in Spielman & Teng 2007:
<http://cs-www.cs.yale.edu/homes/spielman/>