

IIT KHARAGPUR

IIT GANDHINAGAR


NPTEL ONLINE CERTIFICATION COURSES

Scalable Data Science

Lecture 1: Introduction

Sourangshu Bhattacharya

Computer Science and Engineering

IIT KHARAGPUR

In this Lecture:

- Stream processing and sketching
- Dimensionality reduction and hashing
- Frameworks for big data computation
- Scalable Machine Learning

Stream processing and sketching

Data Streams

- **In many data mining situations, we do not know the entire data set in advance**

- **Stream Management** is important when the input rate is controlled **externally:**
  - Google queries
  - Twitter or Facebook status updates

- We can think of the **data** as **infinite** and **non-stationary** (the distribution changes over time)

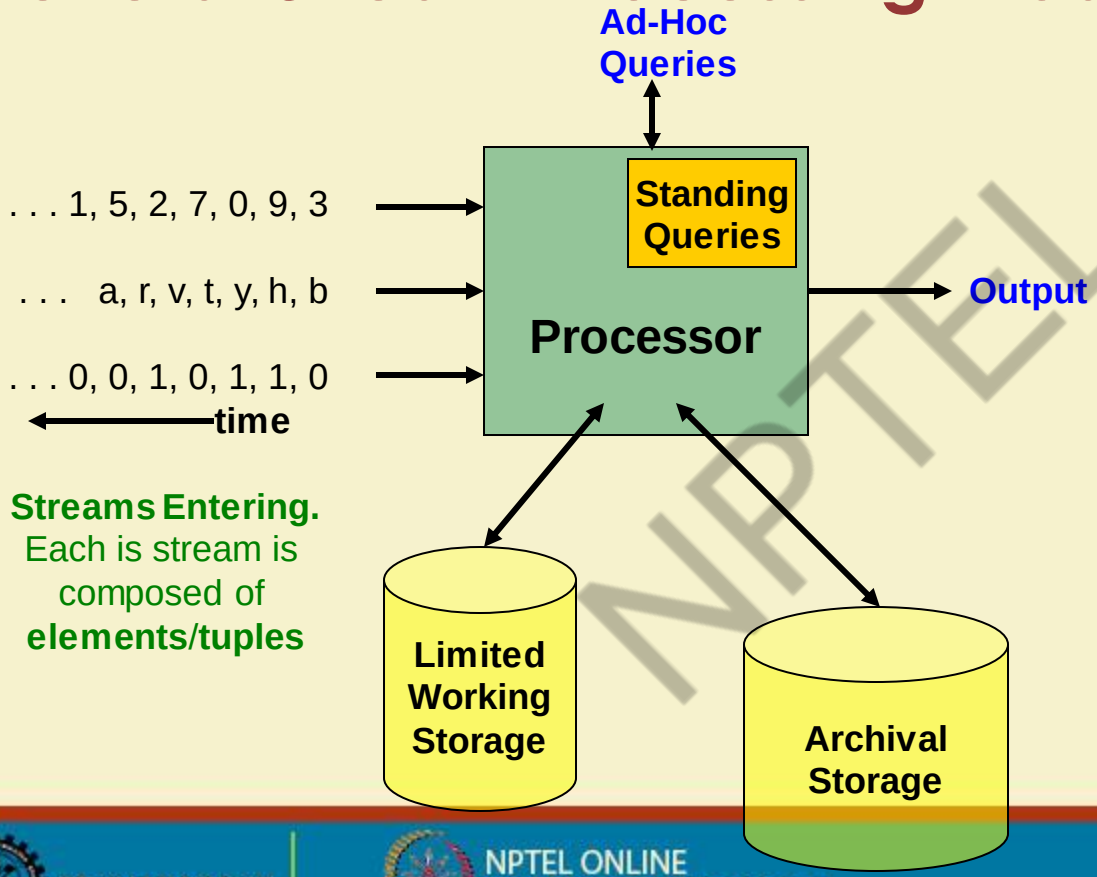
The Stream Model

- Input **elements** enter at a rapid rate, at one or more input ports (i.e., **streams**)
  - **We call elements of the stream tuples**

- **The system cannot store the entire stream accessibly**

- **Q: How do you make critical calculations about the stream using a limited amount of (secondary) memory?**

General Stream Processing Model



Streams Entering. Each is stream is composed of **elements/tuples**

Problems on Data Streams

- **Types of queries one wants on answer on a data stream:**
  - **Sampling data from a stream**
    - Construct a random sample
  - **Queries over sliding windows**
    - Number of items of type ***x*** in the last ***k*** elements of the stream

Sliding Windows

- A useful model of stream processing is that queries are about a ***window*** of length ***N*** – the ***N*** most recent elements received

- **Amazon example:**
  - For every product **X** we keep 0/1 stream of whether that product was sold in the **n**-th transaction
  - We want to answer queries, how many times have we sold **X** in the last **k** sales

Maintaining a fixed-size sample

- **Problem: Fixed-size sample**
- **Suppose we need to maintain a random sample *S* of size exactly *s* tuples**
  - E.g., main memory size constraint
- **Why?** Don't know length of stream in advance
- **Suppose at time *n* we have seen *n* items**
  - **Each item is in the sample *S* with equal prob. *s/n***

How to think about the problem: say s = 2

Stream: a x c y z k c d e g…

At **n= 5,** each of the first 5 tuples is included in the sample **S** with equal prob.
At **n= 7,** each of the first 7 tuples is included in the sample **S** with equal prob.

Impractical solution would be to store all the *n* tuples seen so far and out of them pick *s* at random

Solution: Fixed Size Sample

- **Algorithm (a.k.a. Reservoir Sampling)**
  - Store all the first ***s*** elements of the stream to ***S***
  - Suppose we have seen ***n-1*** elements, and now the ***nth*** element arrives (***n > s***)
    - With probability ***s/n***, keep the ***nth*** element, else discard it
    - If we picked the ***nth*** element, then it replaces one of the ***s*** elements in the sample ***S***, picked uniformly at random
- **Claim:** This algorithm maintains a sample ***S*** with the desired property:
  - After ***n*** elements, the sample contains each element seen so far with probability ***s/n***

Proof: By Induction

- **We prove this by induction:**
  - Assume that after ***n*** elements, the sample contains each element seen so far with probability ***s/n***
  - We need to show that after seeing element ***n+1*** the sample maintains the property
    - Sample contains each element seen so far with probability ***s/(n+1)***
- **Base case:**
  - After we see **n=s** elements the sample **S** has the desired property
    - Each out of **n=s** elements is in the sample with probability ***s/s = 1***

Proof: By Induction

- **Inductive hypothesis:** After ***n*** elements, the sample ***S*** contains each element seen so far with prob. ***s/n***
- **Now element *n+1* arrives**
- **Inductive step:** For elements already in ***S***, probability that the algorithm keeps it in ***S*** is:

$$\left(1-\frac{s}{n+1}\right)+\left(\frac{s}{n+1}\right)\left(\frac{s-1}{s}\right)=\frac{n}{n+1}$$

Element **n+1** discarded

Element **n+1** not discarded

Element in the sample not picked

- So, at time ***n,*** tuples in ***S*** were there with prob. **s/n**
- Time ***n→n+1,*** tuple stayed in ***S*** with prob. **n/(n+1)**
- So prob. tuple is in ***S*** at time ***n+1*** = $\frac{s}{n}\cdot\frac{n}{n+1}=\frac{s}{n+1}$

Problems on Data Streams

- **Types of queries one wants on answer on a data stream:**
  - **Filtering a data stream**
    - Select elements with property ***x*** from the stream
  - **Counting distinct elements**
    - Number of distinct elements in the last ***k*** elements of the stream
  - **Estimating moments**
    - Estimate avg./std. dev. of last ***k*** elements
  - **Finding frequent elements**

Applications (1)

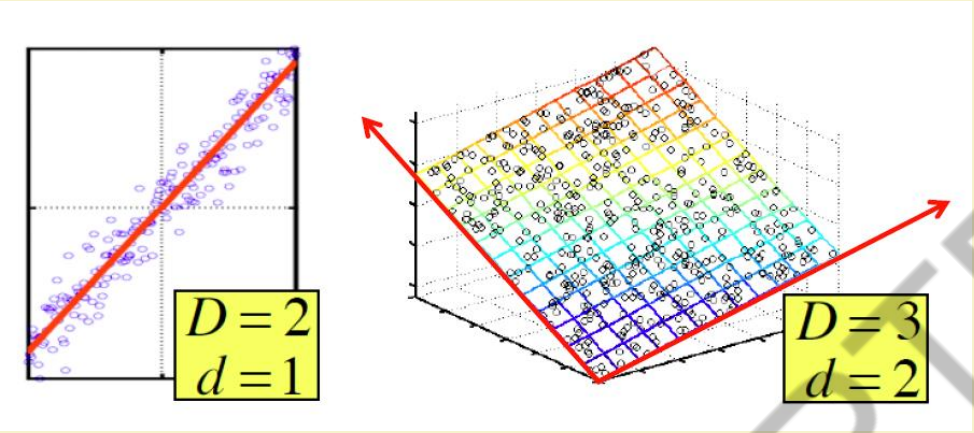
- **Mining query streams**
  - Google wants to know what queries are more frequent today than yesterday
- **Mining click streams**
  - A web company wants to know which of its pages are getting an unusual number of hits in the past hour
- **Mining social network news feeds**
  - E.g., look for trending topics on Twitter, Facebook

Applications (2)

- **Sensor Networks**
  - Many sensors feeding into a central controller
- **Telephone call records**
  - Data feeds into customer bills as well as settlements between telephone companies
- **IP packets monitored at a switch**
  - Gather information for optimal routing
  - Detect denial-of-service attacks

Dimensionality reduction

Dimensionality Reduction



- **Assumption:** Data lies on or near a low *d*-dimensional subspace
- **Axes of this subspace are effective representation of the data**

Dimensionality Reduction

- **Compress / reduce dimensionality:**
  - 10^6 rows; 10^3 columns; no updates
  - Random access to any cell(s); **small error: OK**

day / customer	We 7/10/96	Th 7/11/96	Fr 7/12/96	Sa 7/13/96	Su 7/14/96
ABC Inc.	1	1	1	0	0
DEF Ltd.	2	2	2	0	0
GHI Inc.	1	1	1	0	0
KLM Co.	5	5	5	0	0
Smith	0	0	0	2	2
Johnson	0	0	0	3	3
Thompson	0	0	0	1	1

The above matrix is really "2-dimensional." All rows can be reconstructed by scaling [1 1 1 0 0] or [0 0 0 1 1]

Why Reduce Dimensions?

Why reduce dimensions?

- **Discover hidden correlations/topics**
  - Words that occur commonly together
- **Remove redundant and noisy features**
  - Not all words are useful
- **Interpretation and visualization**
  - Genres of movies
- **Easier storage and processing of the data**

Locality sensitive hashing

[Hays and Efros, SIGGRAPH 2007]

Scene Completion Problem

Scene Completion Problem

[Hays and Efros, SIGGRAPH 2007]

Scene Completion Problem











10 nearest neighbors from a collection of 20,000 images

[Hays and Efros, SIGGRAPH 2007]

Scene Completion Problem









10 nearest neighbors from a collection of 2 million images

A Common Metaphor

- **Many problems can be expressed as finding "similar" sets:**
  - **Find near-neighbors in high-dimensional space**
- **Examples:**
  - **Pages with similar words**
    - For duplicate detection, classification by topic
  - **Customers who purchased similar products**
    - Products with similar customer sets
  - **Images with similar features**
    - Users who visited similar websites

Problem definition

- **Given: High dimensional data points $x_1, x_2, \ldots$**
  - **For example:** Image is a long vector of pixel colors

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow [1\ 2\ 1\ 0\ 2\ 1\ 0\ 1\ 0]$$

- **And some distance function $d(x_1, x_2)$**
  - Which quantifies the "distance" between x_1 and x_2
- **Goal:** Find **all pairs of data points (x_i, x_j)** that are within some distance threshold $d(x_i, x_j) \leq s$
- **Note:** Naïve solution would take $O(N^2)$ ☹

  where N is the number of data points
- **MAGIC: This can be done in $O(N)$!! How?**

Frameworks for big data computation

MapReduce

- Much of the course will be devoted to **large scale computing** for **data mining**
- **Challenges:**
  - How to distribute computation?
  - Distributed/parallel programming is hard
- **Map-reduce** addresses all of the above
  - Google's computational/data manipulation model
  - Elegant way to work with big data

Example: Language Model

- **Statistical machine translation:**
  - Need to count number of times every 5-word sequence occurs in a large corpus of documents

- **Very easy with MapReduce:**
  - **Map:**
    - Extract (5-word sequence, count) from document
  - **Reduce:**
    - Combine the counts

Example: Host size

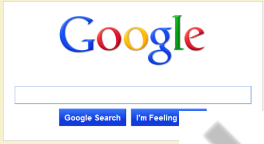
- **Suppose we have a large web corpus**
- Look at the metadata file
  - Lines of the form: (URL, size, date, …)
- **For each host, find the total number of bytes**
  - That is, the sum of the page sizes for all URLs from that particular host

- **Other examples:**
  - Link analysis and graph processing
  - Machine Learning algorithms

Scalable Machine Learning

Big Data

- 6 Billion web queries per day.
  ~ 6 TB per day, ~ 2.5 PB per year
- 10 Billion display ads per day.
  ~ 15 TB per day, ~ 5.5 PB per year
- 30 Billion text ads per day.
  ~ 30 TB per day, ~ 11 PB per year
- 150 Million Credit card transactions per day.
  ~ 150 GB per day, ~ 5.5 TB per year
- 100 Billion emails per day.
  ~ 1 PB per day, ~ 360 PB per year



Machine Learning on Big Data

- 6 Billion web queries per day.
  ~ 6 TB per day, ~ 2.5 PB per year
- 10 Billion display ads per day.
  ~ 15 TB per day, ~ 5.5 PB per year
- 30 Billion text ads per day.
  ~ 30 TB per day, ~ 11 PB per year
- 150 Million Credit card transactions per day.
  ~ 150 GB per day, ~ 5.5 TB per year
- 100 Billion emails per day.
  ~ 1 PB per day, ~ 360 PB per year

- Ranking search results
  Training ranking algorithms from past searches
- Segmentation of customers e.g. “high income male”
  View count by customer segments
- Click through rate estimation
  Training logistic regression
- Fraudulent transactions
  Anomaly detection
- Personalised spam filtering
  Multi-task binary classification

Large Scale Machine Learning

- **Main question:**
  How to efficiently train
  (build a model/find model parameters)**?**

- **Auxiliary question: fast / scalable optimization**
  - **Stochastic / online optimization**
  - **Distributed optimization.**

References:

- Jure Leskovec, Anand Rajaraman, Jeff Ullman. **Mining of Massive Datasets.** *2nd edition. - Cambridge University Press.* *http://www.mmds.org/*

Thank You!!



IIT KHARAGPUR

IIT GANDHINAGAR


NPTEL ONLINE CERTIFICATION COURSES

Scalable Data Science

Lecture 2: Background Probability Theory

Sourangshu Bhattacharya
Computer Science and Engineering
IIT KHARAGPUR

Probability: Definition

- ***Experiment***: toss a coin twice
- ***Sample space***: possible outcomes of an experiment
  - S = {HH, HT, TH, TT}
- ***Event***: a subset of possible outcomes
  - A={HH}, B={HT, TH}
- ***Probability of an event*** : an number assigned to an event Pr(A)
  - Axiom 1: $\Pr(A) \geq 0$
  - Axiom 2: $\Pr(S) = 1$
  - Axiom 3: For every sequence of disjoint events

$$\Pr\left(\bigcup_i A_i\right) = \sum_i \Pr(A_i)$$

  - Example: Pr(A) = n(A)/N: frequentist statistics

Probability

- **Joint Probability**: For events *A* and *B*, joint probability *Pr(AB)* stands for the probability that both events happen.
- **Indepedence**: Two events *A* and *B* are independent in case

$$\Pr(AB) = \Pr(A)\Pr(B)$$

- A set of events $\{A_i\}$ are independent in case

$$\Pr\left(\bigcap_i A_i\right) = \prod_i \Pr(A_i)$$

- **Conditional Probability**: If A and B are events with Pr(A) > 0, the ***conditional probability of B given A*** is

$$\Pr(B|A) = \frac{\Pr(AB)}{\Pr(A)}$$

Random Variable and Distributions

- A ***random variable X*** is a numerical outcome of a random experiment.
- **Discrete random variable**
  - Takes on one of a finite (or at least countable) number of different values.
  - Examples:
    - X = 1 if heads, 0 if tails
    - Y = 1 if male, 0 if female (phone survey)
    - Z = # of spots on face of thrown die
- **Distribution function or mass function**: For a discrete r.v. *X*, we have $\Pr(X = x)$ or $\Pr(x)$ or $P(x)$, i.e., the probability that r.v. X takes on a given value x.
- **Properties**: $\Pr(X = x) > 0$ and $\sum_x \Pr(x) = 1$.

Random Variable and Distributions

- **Continuous random variable**
  - Takes on one in an infinite range of different values
  - Examples:
    - W = % GDP grows (shrinks?) this year
    - V = hours until light bulb fails
- **Distribution function**:
  - What is the probability that a continuous r.v. takes on a specific value?
    e.g. *Prob(V = 3.14159265 hrs) =* **0**
  - However, ranges of values can have non-zero probability.
    e.g. *Prob(3 hrs <= V <= 4 hrs) = 0.1*
  - For a continuous r.v. *X*, we have $\Pr(x)$ or $P(x) = \Pr(x \leq X \leq x + dx)$.
- **Properties**: $\mathrm{P}(x) > 0$ and $\int_x P(x)dx = 1$.

Expectation

- A discrete random variable $X \sim P(X = x)$. Then, its expectation is:

$$E[X] = \sum_x x\, P(X = x)$$

- For an empirical sample, $x_1, x_2, \ldots, x_N$, expectation can be estimated as:

$$E[X] = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- Continuous random variable: $E[X] = \int_x xP(x)dx$
- Expectation of sum of random variables: $E[X_1 + X_2] = E[X_1] + E[X_2]$.
- A measure of central tendency. Other measures: median, mode, etc.

Variance

- The variance of a random variable X is the expectation of $(X - E[x])2$:

$$
\begin{aligned}
Var(X) &= E((X - E[X])^2) \\
&= E(X^2 + E[X]^2 - 2XE[X]) \\
&= E(X^2 - E[X]^2) \\
&= E[X^2] - E[X]^2
\end{aligned}
$$

Bernoulli Distribution

- The outcome of an experiment can either be success (i.e., 1) and failure (i.e., 0).
- $\Pr(X = 1) = p, \Pr(X = 0) = 1 - p$, or

$$p_\theta(x) = p^x(1-p)^{1-x}$$

- $E[X] = p, Var(X) = p(1-p)$

Binomial Distribution

- n draws of a Bernoulli distribution
  $X_i \sim Bernoulli(p), X = \sum_{i=1}^{n} X_i, X \sim Bin(p, n)$

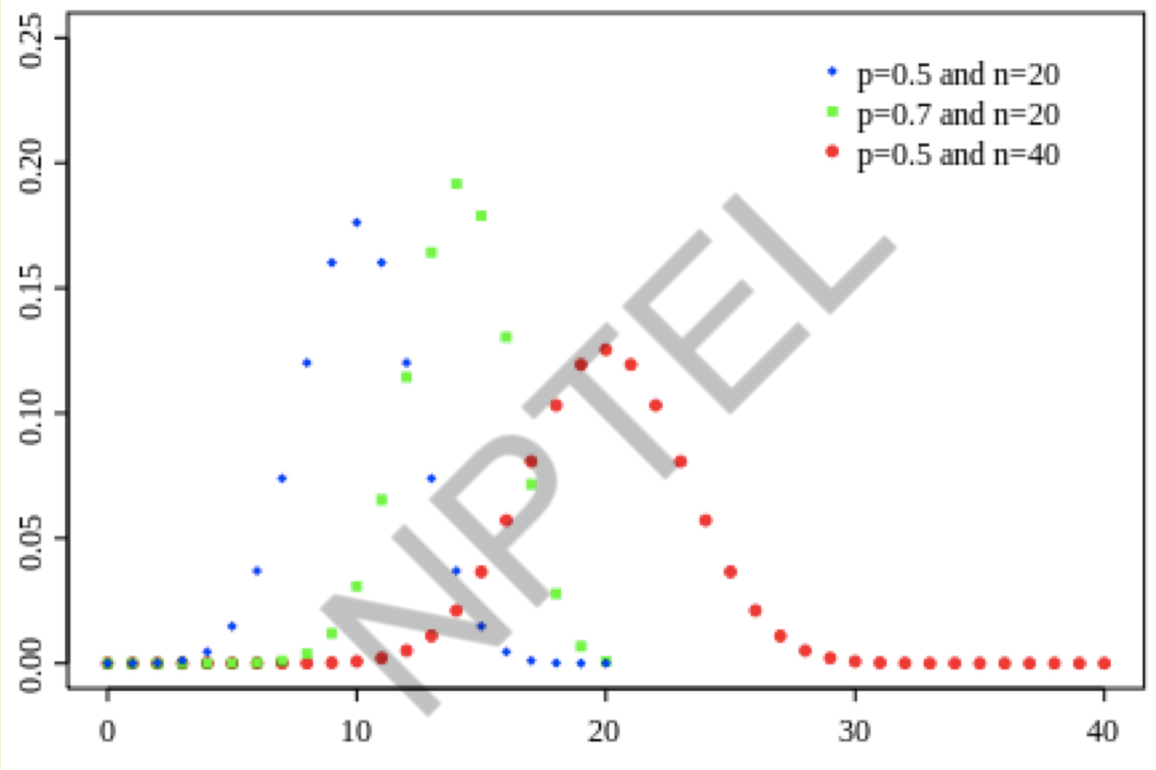
- Random variable X stands for the number of times that experiments are successful.

$$\Pr(X = x) = p_\theta(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & x = 1, 2, \ldots, n \\ 0 & \text{otherwise} \end{cases}$$

- $E[X] = np, Var(X) = np(1-p)$

Sourangshu Bhattacharya
Computer Science and Engg.

Plots


p=0.5 and n=20
p=0.7 and n=20
p=0.5 and n=40
0.00
0.05
0.10
0.15
0.20
0.25
0
10
20
30
40

Poisson Distribution

- Distribution of number of arrivals, given the average rate of arrival, λ.

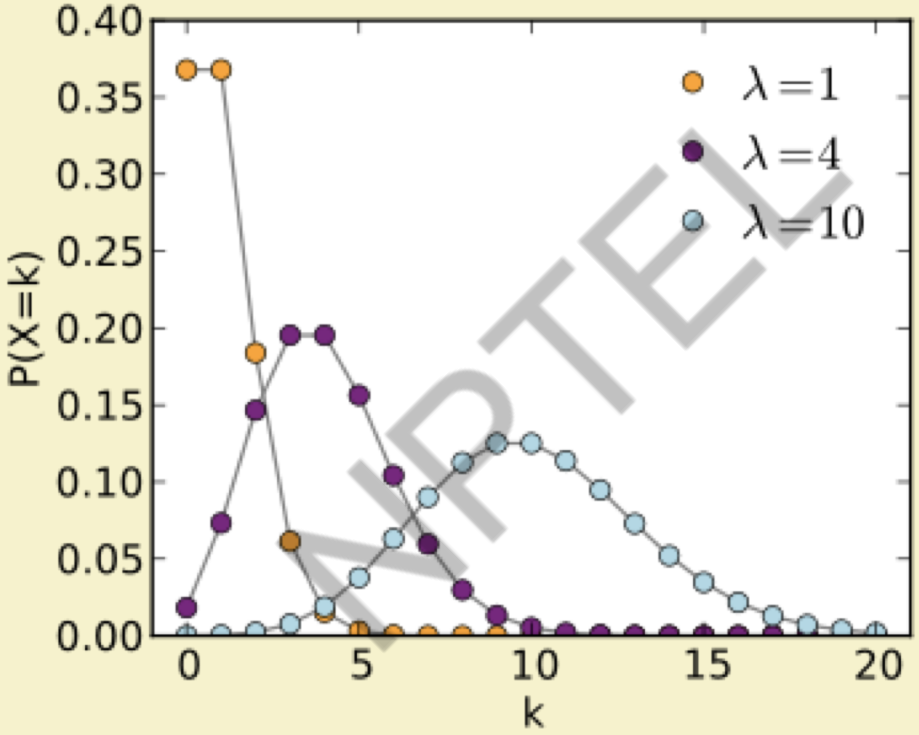
- Coming from Binomial distribution
  - Fix the expectation $\lambda = np$
  - Let the number of trials $n \to \infty$

  A Binomial distribution will become a Poisson distribution

$$\Pr(X = x) = p_\theta(x) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- $E[X] = \lambda$, $Var(X) = \lambda$

Plots



Normal (Gaussian) Distribution

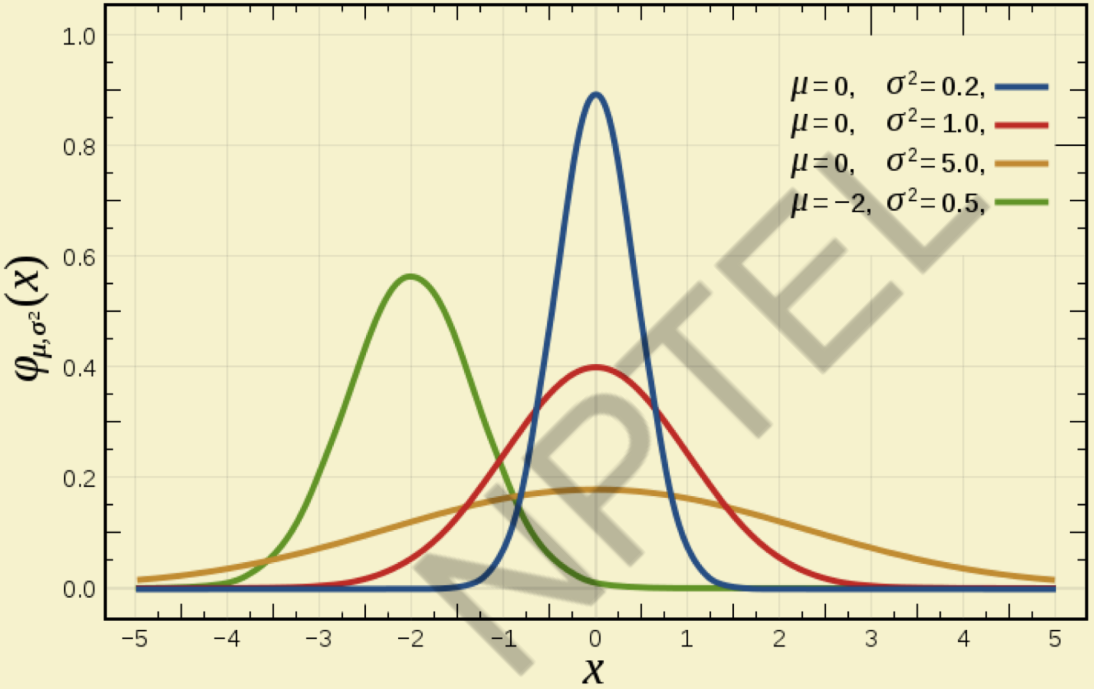
- Continuous valued distribution
- $X \sim N(\mu, \sigma)$

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

$$\Pr(a \le X \le b) = \int_a^b p_\theta(x)dx = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} dx$$

- $E[X] = \mu, Var(X) = \sigma^2$
- If $X_1 \sim N(\mu_1, \sigma_1^2)$ and $X_2 \sim N(\mu_2, \sigma_2^2)$, $X = X_1 + X_2$, then $X \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.

Plots


1.0
0.8
0.6
0.4
0.2
0.0
$\varphi_{\mu,\sigma^2}(x)$
−5
−4
−3
−2
−1
0
1
2
3
4
5
x
$\mu = 0, \quad \sigma^2 = 0.2,$
$\mu = 0, \quad \sigma^2 = 1.0,$
$\mu = 0, \quad \sigma^2 = 5.0,$
$\mu = -2, \quad \sigma^2 = 0.5,$

Concentration Inequalities

Motivation

Many times we do not need to calculate probabilities exactly.

Sometimes it is enough to know that a probability is very small (or very large)

E.g. $P(\text{earthquake tomorrow}) = ?$

This is often a lot easier

I toss a coin 1000 times. The probability that I get 14 consecutive heads is

A	B	C
$< 10\%$	$\approx 50\%$	$> 90\%$

Consecutive heads

Let N be the number of occurrences of 14 consecutive heads in 1000 coin flips.

$$N = I_1 + \ldots + I_{987}$$

where I_i is an indicator r.v. for the event

"14 consecutive heads starting at position i"

$$E[I_i] = P(I_i = 1) = 1/2^{14}$$

$$E[N] = 987 \cdot 1/2^{14} = 987/16384 \approx 0.0602$$

Markov's inequality

For every non-negative random variable X and every value a:

$$P(X \geq a) \leq E[X] / a.$$

$$E[N] \approx 0.0602$$

$$P[N \geq 1] \leq E[N] / 1 \leq 6\%.$$

Sourangshu Bhattacharya
Computer Science and Engg.

Markov's inequality

For every non-negative random variable X: and every value a:

$$P(X \geq a) \leq E[X] / a.$$

$$E[X] = E[X \mid X \geq a]\, P(X \geq a) + E[X \mid X < a]\, P(X < a)$$

$\geq a$ ≥ 0 ≥ 0

$$E[X] \geq a\, P(X \geq a) + 0.$$

Patterns

A coin is tossed 1000 times. Give an upper bound on the probability that the pattern HH occurs:

(a) at least 500 times

(b) at most 100 times

Patterns

(a) **Let N be the number of occurrences of HH.**
Last time we calculated $E[N] = 999/4 = 249.75.$

$$P[N \geq 500] \leq E[N] \,/\, 500 = 249.75/500 \approx 49.88\%$$

so 500+ HHs occur with probability $\leq 49.88\%$**.**

(b) $P[N \leq 100] \leq ?$

$$P[N \leq 100] = P[999 - N \geq 899] \leq E[999 - N] \,/\, 899$$
$$= (999 - 249.75)/\, 899$$
$$\leq 83.34\%$$

Chebyshev's inequality

For every random variable X and every t:

$$P(|X - \mu| \geq t\sigma) \leq 1 / t^2.$$

where $\mu = E[X]$, $\sigma = \sqrt{Var[X]}$.

Patterns

$E[N] = 999/4 = 249.75$ $\mu = 249.75$

$Var[N] = (5 \cdot 999 - 7)/16 = 311.75$ $\sigma \approx 17.66$

(a) $P(X \geq 500) \leq P(|X - \mu| \geq 14.17\sigma)$

$\leq 1/14.17^2 \approx 0.50\%$

(b) $P(X \leq 100) \leq P(|X - \mu| \geq 8.47\sigma)$

$\leq 1/8.47^2 \approx 1.39\%$

Sourangshu Bhattacharya
Computer Science and Engg.

Chebyshev's inequality

For every random variable X and every a:

$$P(|X-\mu| \geq t\sigma) \leq 1 / t^2.$$

where $\mu = E[X]$, $\sigma = \sqrt{Var}[X]$.

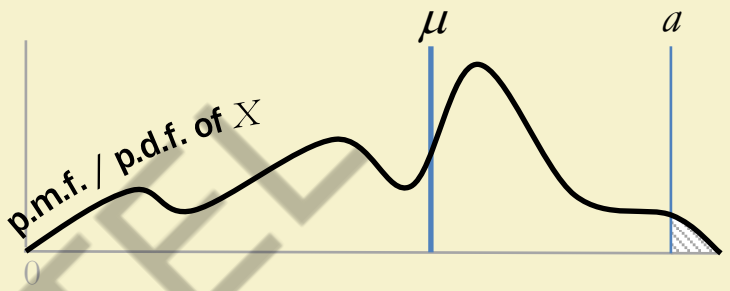
$$P(|X-\mu| \geq t\sigma) = P((X-\mu)^2 \geq t^2\sigma^2) \leq E[(X-\mu)^2] / t^2\sigma^2 = 1 / t^2.$$

Sourangshu Bhattacharya
Computer Science and Engg.

An illustration

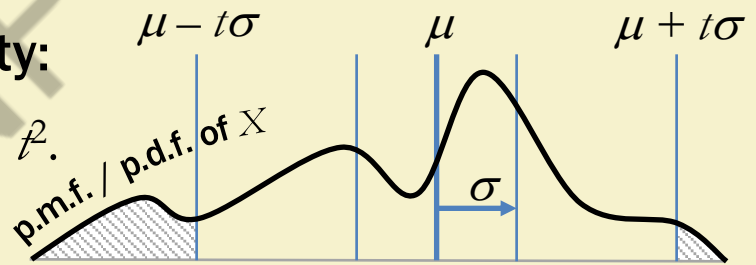
Markov's inequality:

$P(X \geq a) \leq \mu / a.$



Chebyshev's inequality:

$P(|X - \mu| \geq t\sigma) \leq 1 / t^2.$



Repertoire of tools

- **Linearity of expectation**: For any random variables $X_1, X_2, \ldots, X_n$, we have
  - $E[\Sigma_i\, Xi] \;=\; \Sigma_i\, E[Xi]$
- **Markov's inequality**: For any random variable X
  - $\Pr[X \;\geq\; c] \;\leq\; E[X]/c$
- **Union bound:** For any sequence of events $E_1, E_2, \ldots, En$, we have
  - $\Pr[U_i\, E_i] \;\leq\; \Sigma_i \Pr[E_i]$

Chernoff bounding

The Chernoff bound for a random variable X is obtained as follows: for any t >0,

$$\Pr[X \geq a] = \Pr[e^{tX} \geq e^{ta}] \leq E[e^{tX}] / e^{ta}$$

Similarly, for any $t < 0$,

$$\Pr[X \leq a] = \Pr[e^{tX} \geq e^{ta}] \leq E[e^{tX}] / e^{ta}$$

The value of t that **minimizes** $E[e^{tX}] / e^{ta}$ gives the best possible bounds.

When $X = X_1 + \cdots + X_n$:

$$\Pr[X \leq a] \leq \min_{t>0} e^{-ta} \prod_i E[e^{tX_i}]$$

Chernoff bounding

- **Def**: The **moment generating function** of a random variable X is $M_X(t) = E[e^{tX}]$.
- $E[X^n] = M_X^n(0)$, which is the nth derivative of $M_X(t)$ evaluated at $t = 0$.
- Fact: If $M_X(t) = MY(t)$ for all t in (-c, c) for some $c > 0$, then X and Y have the same distribution.
- If X and Y are independent r.v., then $M_{X+Y}(t) = MX(t)\,MY(t)$.

Chernoff bounding

- Let $X_1, X_2, \ldots, X_n$ be n independent random variables in {0,1}, with $X = X_1 + X_2 + \cdots + X_n$.

- For any nonnegative δ

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu}$$

- For any δ in [0,1]

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}$$

Chernoff bounding

- Extensions:
  - Chernoff-Hoeffding bounds (bounded r.vs)
  - Azuma's inequality for martingales

- Applications in this course:
  - Sketching: Hashing.
  - Random Projection: Proof of Johnson-Lindenstrauss lemma.
  - Dimensionality reduction: CUR decomposition.

References:

- Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

- Lecture notes of Andrej Bogdanov. http://www.cse.cuhk.edu.hk/~andrejb/engg2040c/s13/

- Wikipedia.

Thank You!!



IIT KHARAGPUR

IIT GANDHINAGAR


NPTEL ONLINE CERTIFICATION COURSES

Scalable Data Science

Lecture 3: Background on Linear Algebra

Sourangshu Bhattacharya

Computer Science and Engineering

IIT KHARAGPUR

In this review

- Recall concepts we'll need in this class
- Geometric intuition for linear algebra
- Outline:
  - Matrices as linear transformations.
  - Linear systems & vector spaces.
  - Solving linear systems.
  - Eigenvalues & eigenvectors.

Basic concepts

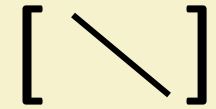
- *Vector* in R^n is an ordered set of n real numbers.
  - e.g. $v = (1,6,3,4)$ is in R^4
  - $(1,6,3,4)$ is a column vector:
  - as opposed to a row vector:
- $m - by - n$ *matrix* is an object with m rows and n columns, each entry fill with a real number:

$$\begin{pmatrix} 1 \\ 6 \\ 3 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 6 & 3 & 4 \end{pmatrix}$$

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

Basic concepts

- Transpose: reflect vector/matrix on line: 

$$\begin{pmatrix} a \\ b \end{pmatrix}^T = \begin{pmatrix} a & b \end{pmatrix} \qquad \begin{pmatrix} a & b \\ c & d \end{pmatrix}^T = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

  - Note: $(Ax)^T = x^T A^T$
- Vector norms:
  - L_p norm of $v = (v_1, \ldots, v_k)$ is $(\Sigma_i |v_i|^p)^{1/p}$
  - Common norms: L_1, L_2
  - $L_{infinity} = maxi\ |vi|$
- Length of a vector v is $L_2(v)$

Basic concepts

- Vector dot product:

$$u \bullet v = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \bullet \begin{pmatrix} v_1 & v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2$$

  - Note dot product of u with itself is the square of the length of u.

- Matrix product (multiplication):

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Basic concepts

- Vector products in matrix multiplication notation:
  - Dot product:

$$u \bullet v = u^T v = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2$$

  - Outer product:

$$uv^T = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} v_1 & v_2 \end{pmatrix} = \begin{pmatrix} u_1 v_1 & u_1 v_2 \\ u_2 v_1 & u_2 v_2 \end{pmatrix}$$

Special matrices

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$$ diagonal

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

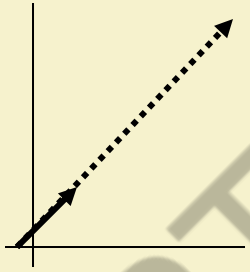
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ I (identity matrix)

$$\begin{pmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{pmatrix}$$ lower-triangular

Matrices as linear transformations

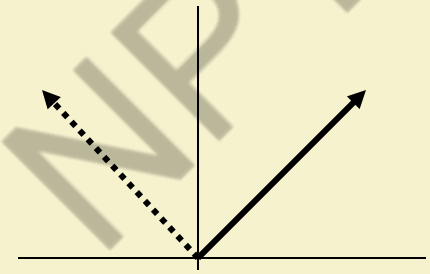
Multiplication with $m \times n$ matrices transform vectors in R^n into vectors in R^m

$$\begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$



Scaling: scalar product of identity matrix

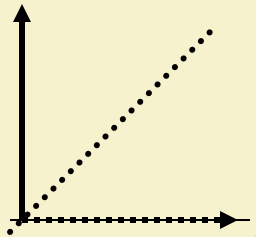
$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$



Rotation: Orthogonal matrices

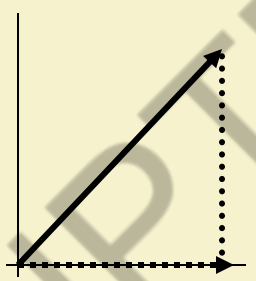
Matrices as linear transformations

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



Reflection

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



Projection onto axis

Vector spaces

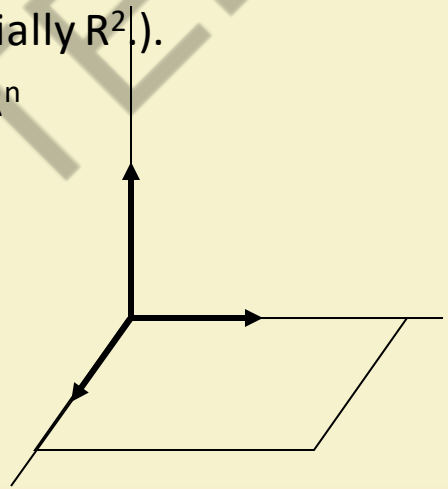
- Formally, a *vector space* is a set of vectors which is closed under addition and multiplication by real numbers (also called linear combination).

$$x = \alpha_1 v_1 + \cdots + \alpha_n v_n$$

- A *subspace* is a subset of a vector space which is a vector space itself, e.g. the plane z=0 is a subspace of R^3 (It is essentially R^2.).
- We'll be looking at R^n and subspaces of R^n

Our notion of planes in R^3 may be extended to *hyperplanes* in R^n (of dimension n-1)

Note: subspaces must include the origin (zero vector).



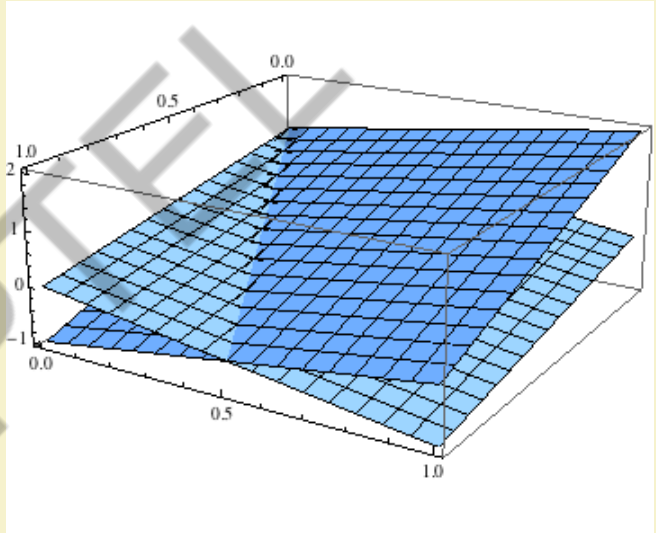
Matrices as sets of constraints

Matrix equations (linear system of equations) can encode a set of linear constraints

$$x + y + z = 1$$

$$2x - y + z = 2$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

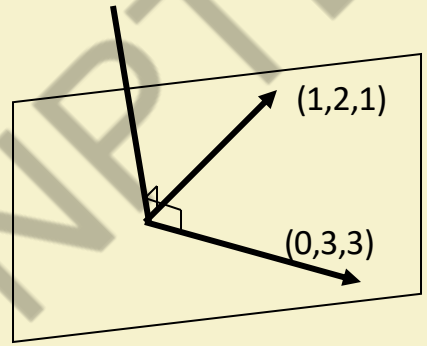


Linear system & subspaces

$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$u\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} + v\begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

- $Ax = b$ is solvable iff b may be written as a linear combination of the columns of A
- The set of all possible vectors b forms a subspace called the *column space* of A



Linear system & subspaces

The set of solutions to $Ax = 0$ forms a subspace called the *null space* of A.

$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

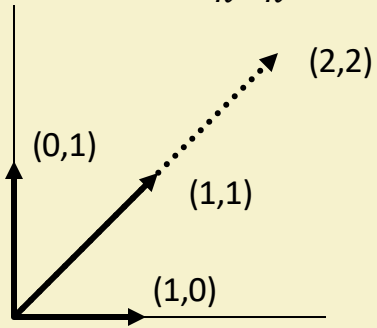
→ Null space: {(0,0)}

$$\begin{pmatrix} 1 & 0 & 1 \\ 2 & 3 & 5 \\ 1 & 3 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

→ Null space: {(c,c,-c)}

Linear independence and basis

- Vectors $v_1, \ldots, v_k$ are linearly independent if $c_1 v_1 + \cdots + c_k v_k = 0$ implies $c_1 = \cdots = c_k = 0$



i.e. the nullspace is the origin

$$\begin{pmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

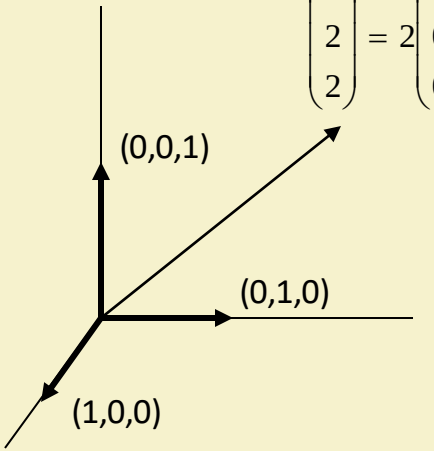
$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Recall nullspace contained only $(u, v) = (0,0)$.
i.e. the columns are linearly independent.

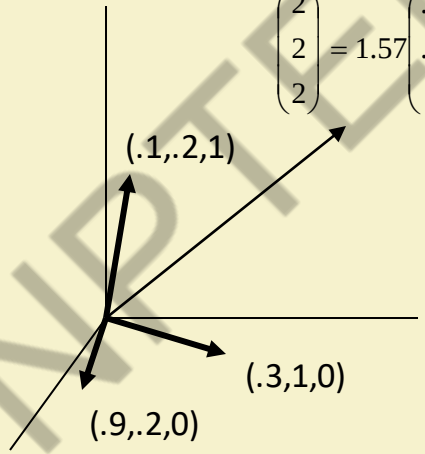
Linear independence and basis

- If all vectors in a vector space may be expressed as linear combinations of $v_1, \ldots, v_k$, then $v_1, \ldots, v_k$ *span* the space.

$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 2\begin{pmatrix}1\\0\\0\end{pmatrix} + 2\begin{pmatrix}0\\1\\0\end{pmatrix} + 2\begin{pmatrix}0\\0\\1\end{pmatrix}$$



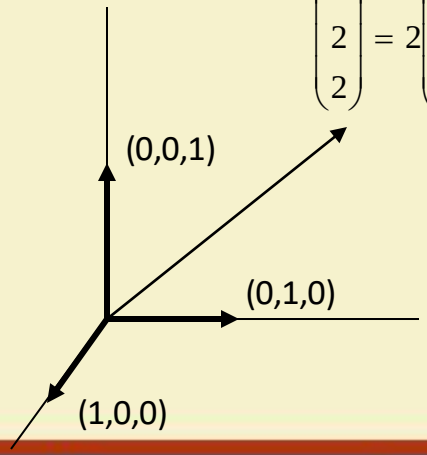
$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 1.57\begin{pmatrix}.9\\.2\\0\end{pmatrix} + 1.29\begin{pmatrix}.3\\1\\0\end{pmatrix} + 2\begin{pmatrix}.1\\.2\\1\end{pmatrix}$$



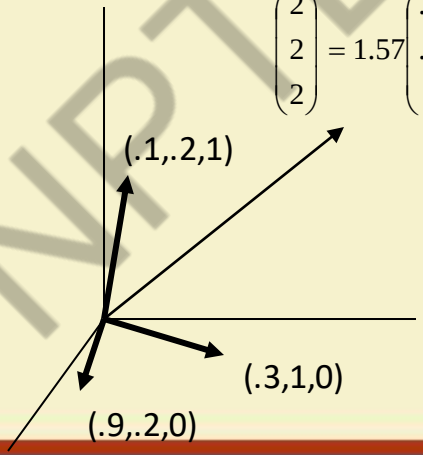
Linear independence and basis

- A *basis* is a set of linearly independent vectors which span the space.
- The *dimension* of a space is the # of "degrees of freedom" of the space; it is the number of vectors in any basis for the space.
- A basis is a maximal set of linearly independent vectors and a minimal set of spanning vectors.

$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 2\begin{pmatrix}1\\0\\0\end{pmatrix} + 2\begin{pmatrix}0\\1\\0\end{pmatrix} + 2\begin{pmatrix}0\\0\\1\end{pmatrix}$$



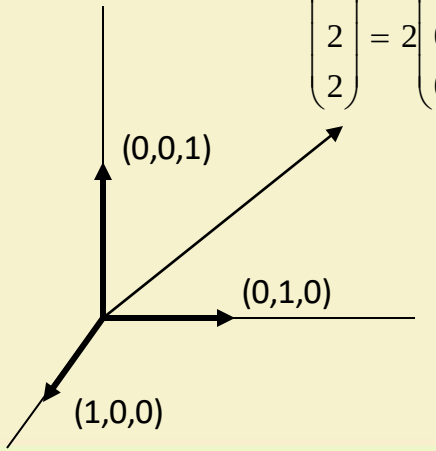
$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 1.57\begin{pmatrix}.9\\.2\\0\end{pmatrix} + 1.29\begin{pmatrix}.3\\1\\0\end{pmatrix} + 2\begin{pmatrix}.1\\.2\\1\end{pmatrix}$$



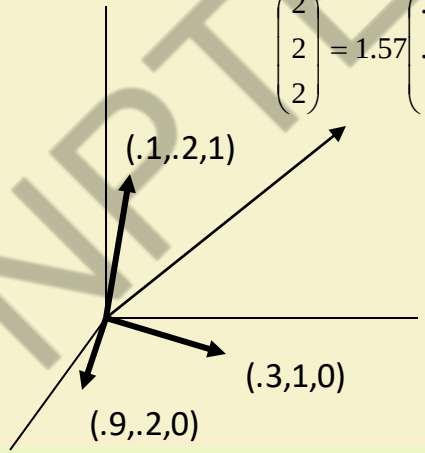
Linear independence and basis

- Two vectors are *orthogonal* if their dot product is 0.
- An *orthogonal basis* consists of orthogonal vectors.
- An *orthonormal basis* consists of orthogonal vectors of unit length.

$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 2\begin{pmatrix}1\\0\\0\end{pmatrix} + 2\begin{pmatrix}0\\1\\0\end{pmatrix} + 2\begin{pmatrix}0\\0\\1\end{pmatrix}$$



$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 1.57\begin{pmatrix}.9\\.2\\0\end{pmatrix} + 1.29\begin{pmatrix}.3\\1\\0\end{pmatrix} + 2\begin{pmatrix}.1\\.2\\1\end{pmatrix}$$



About subspaces

- The *rank* of A is the dimension of the column space of A.
- It also equals the dimension of the *row space* of A (the subspace of vectors which may be written as linear combinations of the rows of A).

$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix}$$

(1,3) = (2,3) – (1,0)

Only 2 linearly independent rows, so rank = 2.

About subspaces

Fundamental Theorem of Linear Algebra:

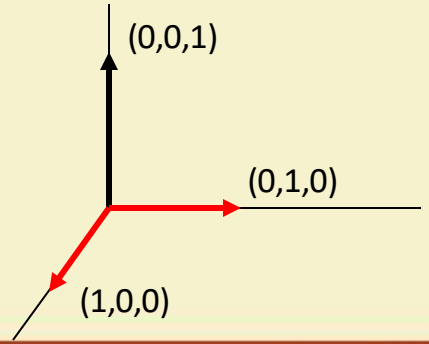
If A is $m \; x \; n$ with rank r,

Column space(A) has dimension r

Nullspace(A) and Nullspace(A^T) has dimension $n - r$ (= *nullity* of A)

Row space(A) = Column space(A^T) has dimension r

Rank-Nullity Theorem: rank + nullity = n



$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

m = 3
n = 2
r = 2

Matrix inversion

- To solve $Ax = b$, we can write a closed-form solution if we can find a matrix A^{-1} s.t. $AA^{-1} = A^{-1}A = I$ (identity matrix)

- Then $Ax = b$ iff $x = A^{-1}b$:

$$x = Ix = A^{-1}Ax = A^{-1}b$$

- A is *non-singular* iff A^{-1} exists iff $Ax = b$ has a unique solution.

- Note: If A^{-1},B^{-1} exist, then $(AB)^{-1} = B^{-1}A^{-1}$,
  and $(A^T)^{-1} = (A^{-1})^T$

Special matrices

- Matrix A is *symmetric* if $A = A^T$
- A is *positive definite* if $x^T Ax > 0$ for all non-zero x (*positive semi-definite* if inequality is not strict).

$$\begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a^2 + b^2 + c^2 \qquad \begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a^2 - b^2 + c^2$$

- Useful fact: Any matrix of form A^TA is positive semi-definite.

  To see this, $x^T(ATA)x = (x^TA^T)(Ax) = (Ax)^T(Ax) \geq 0$

Sourangshu Bhattacharya
Computer Science and Engg.

Determinants

- If det(A) = 0, then A is singular, also called rank deficient
- If det(A) ≠ 0, then A is invertible.
- To compute:

$$\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

$$\det\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Eigenvalues & eigenvectors

- How can we characterize matrices?
- The solutions to $Ax = \lambda x$ in the form of eigenpairs (λ,x) = (eigenvalue,eigenvector) where x is non-zero.
- To solve this, $(A - \lambda I)x = 0$
- λ is an eigenvalue iff $\det(A - \lambda I) = 0$

Eigenvalues & eigenvectors

$(A - \lambda I)x = 0$

λ is an eigenvalue iff $\det(A - \lambda I) = 0$

Example:

$$A = \begin{pmatrix} 1 & 4 & 5 \\ 0 & 3/4 & 6 \\ 0 & 0 & 1/2 \end{pmatrix}$$

$$\det(A - \lambda I) = \begin{pmatrix} 1-\lambda & 4 & 5 \\ 0 & 3/4-\lambda & 6 \\ 0 & 0 & 1/2-\lambda \end{pmatrix} = (1-\lambda)(3/4-\lambda)(1/2-\lambda)$$

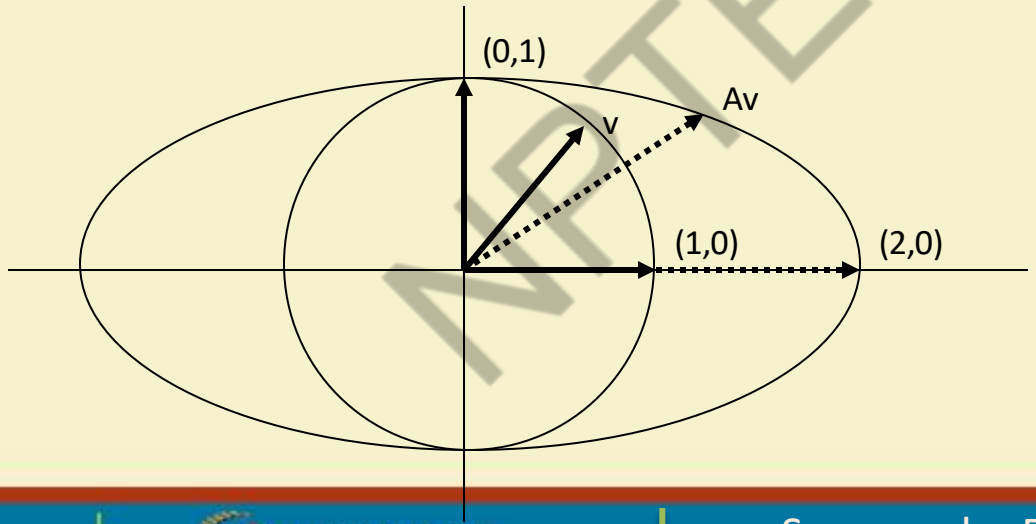
$$\lambda = 1, \lambda = 3/4, \lambda = 1/2$$

Eigenvalues & eigenvectors

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

Eigenvalues λ = 2, 1 with eigenvectors (1,0), (0,1)

Eigenvectors of a linear transformation A are not rotated (but will be scaled by the corresponding eigenvalue) when A is applied.



Properties of Eigenvalues and Eigenvectors

- If $\lambda_1, \ldots, \lambda_n$ are *distinct* eigenvalues of a matrix, then the corresponding eigenvectors $e_1, \ldots, e_n$ are linearly independent.

- If e_1 is an eigenvector of a matrix with corresponding eigenvalue λ_1, then any nonzero scalar multiple of e_1 is also an eigenvector with eigenvalue λ_1.

- A real, symmetric square matrix has real eigenvalues, with orthogonal eigenvectors (can be chosen to be orthonormal).

SVD: Singular Value Decomposition

- Any matrix A $(m \quad n)$ can be written as the product of three matrices: $A = UDV^T$
  - U is an $m \times m$ orthonormal matrix
  - D is an $m \times n$ diagonal matrix. Its diagonal elements, $\sigma_1, \sigma_2, \ldots$, are called the **singular values** of A, and satisfy $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$.
  - V is an $n \times n$ orthonormal matrix
- Example: if $m > n$

$$\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ | & | & | & & | \\ u_1 & u_2 & u_3 & \text{L} & u_m \\ | & | & | & & | \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_n \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \leftarrow & v_1^T & \rightarrow \\ & \text{M} & \\ \leftarrow & v_n^T & \rightarrow \end{bmatrix}$$

$A \qquad U \qquad D \qquad V^T$

Some Properties of SVD

- The rank of matrix A is equal to the number of nonzero singular values σ_i .
- A square $(n \times n)$ matrix A is singular
  if and only if
  at least one of its singular values $\sigma_1, \ldots, \sigma_n$ is zero.

References:

- Strang, Gilbert. **Introduction to Linear Algebra.** 4th ed. *Wellesley-Cambridge Press*, 2009.

- Wikipedia.

Thank You!!



IIT KHARAGPUR

IIT GANDHINAGAR


NPTEL ONLINE CERTIFICATION COURSES

Scalable Data Science

Lecture 4: Background on Optimization

Sourangshu Bhattacharya

Computer Science and Engineering

IIT KHARAGPUR

In this review

- Outline:
  - Definition of an optimization problem
  - Properties of optima
  - Algorithm for differentiable objectives
  - Convex functions – subgradients
  - Subgradient descent.
  - Stochastic gradient descent.

What is Optimization?

Find the minimum or maximum value of an objective function (f_0) w.r.t. arguments x.

The arguments must satisfy given a set of inequality and equality constraints.

General form:

$$\arg\min_x f_0(x)$$
$$\text{s.t.} f_i(x) \leq 0, i = \{1, \ldots, k\}$$
$$h_j(x) = 0, j = \{1, \ldots l\}$$

Example:

$$\min f(x,y) = x^2 + 2y^2$$
$$x > 0$$

or

$$\min f(x,y) = x^2 + 2y^2$$
$$-2 < x < 5, y \geq 1$$

or

$$\min f(x,y) = x^2 + 2y^2$$
$$x + y = 2$$

Why Do We Care?

Linear Classification

$$\arg\min_{w} \sum_{i=1}^{n} ||w||^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t. } 1 - y_i x_i^T w \leq \xi_i$$

$$\xi_i \geq 0$$

Maximum Likelihood

$$\arg\max_{\theta} \sum_{i=1}^{n} \log p_\theta(x_i)$$

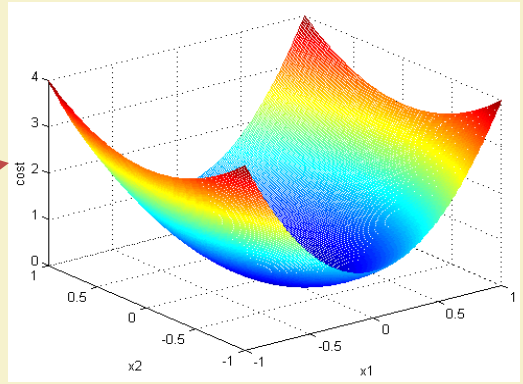
Machine Learning is Optimization !!

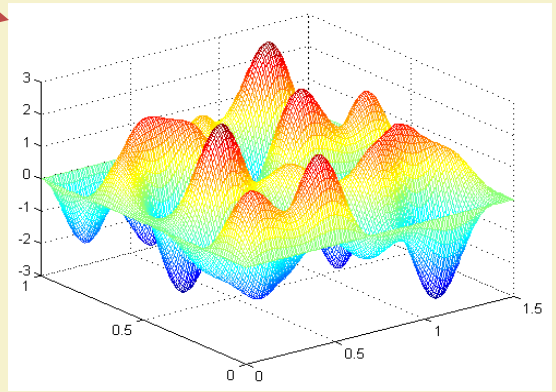
K-Means

$$\arg \min_{\mu_1, \mu_2, \ldots, \mu_k} J(\mu) = \sum_{j=1}^{k} \sum_{i \in C_j} ||x_i - \mu_j||^2$$

Types of objective functions

1. Objective functions may be unimodal or multimodal.
   a. Unimodal – only one optimum
   b. Multimodal – more than one optimum
2. Most algorithms work on unimodal functions. The optimum determined in such cases is called a local optimum.
3. The global optimum is the best of all local optimum designs.





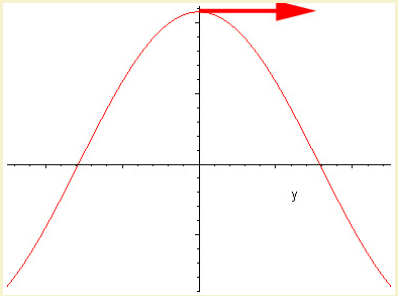
Types of optimization algorithms

- Derivative-based optimization (gradient based)
  - Objective function should be differentiable.
  - Capable of determining "search directions" according to an objective function's derivative.
    - Steepest descent method (Gradient descent);
    - Newton's method;
    - Conjugate gradient, etc.
- Derivative-free optimization
  - Searches over the feasible set in a systematic manner.
    - random search method;
    - genetic algorithm;
    - simulated annealing; etc.

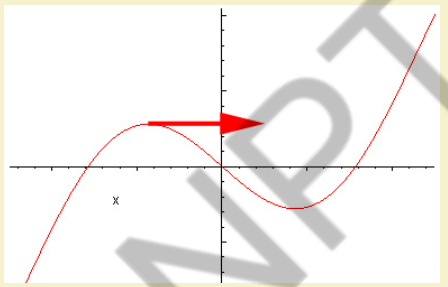
Gradient

- Definition: The gradient of f: $R^n \rightarrow R$ is a function ∇f: $R^n \rightarrow R^n$ given by

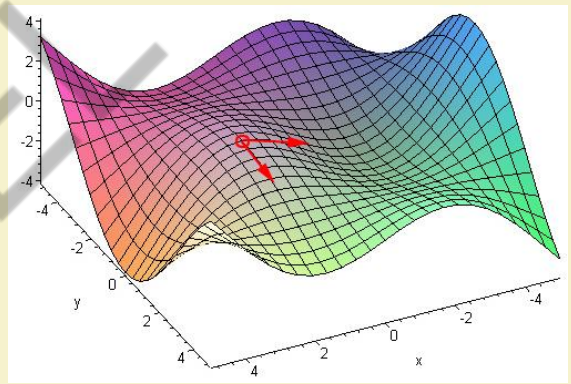
$$\nabla f(x_1, \ldots, x_n) := \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^T$$



$$\frac{\partial f(x,y)}{\partial y}$$



$$\frac{\partial f(x,y)}{\partial x}$$

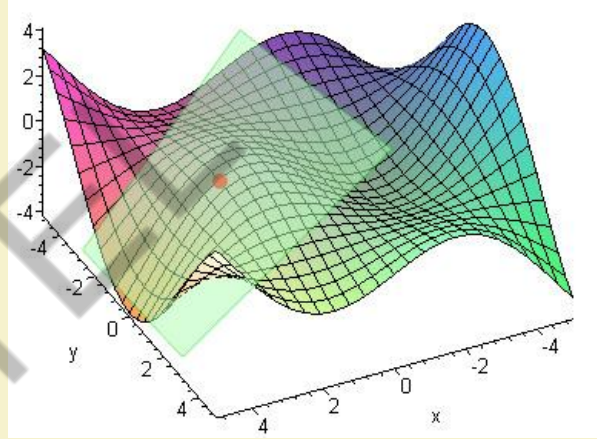


Gradient: $[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}]$

Gradient

- The gradient defines (hyper) plane approximating the function infinitesimally

$$\Delta z = \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y$$



- For all directions v, $|v| = 1$:

$$\frac{\partial f}{\partial v}(p) = \langle \nabla f_p, v \rangle$$

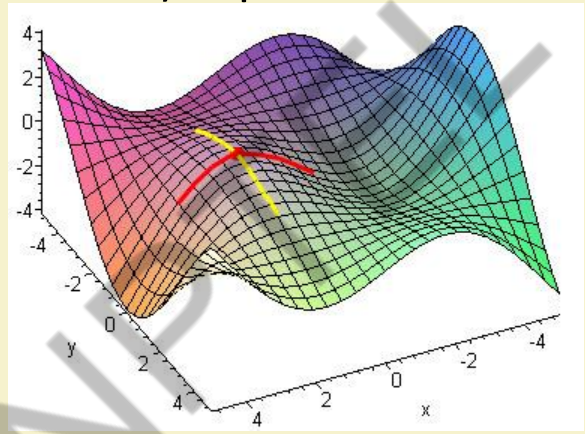
- Magnitude is highest when v and ∇f_p point to the same direction.
- Gradient points to direction of steepest descent

Gradient

- Let f: $R^n \rightarrow R$ be a smooth function around p, if f has local **minimum** (maximum) at p then:

$$\nabla f_p = \bar{0}$$

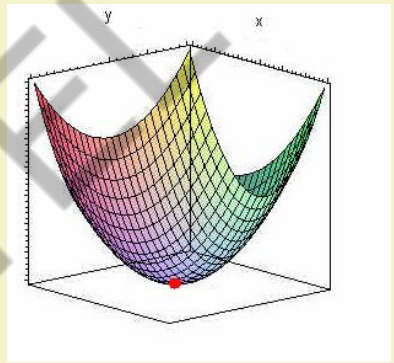
Intuitive: necessary for local min(max)

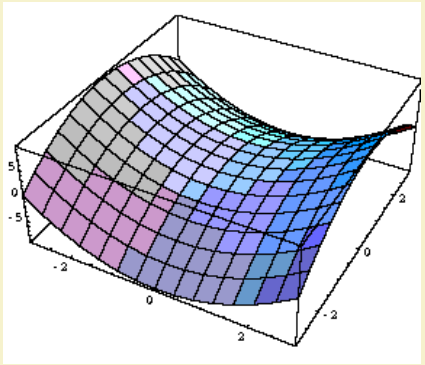


Hessian matrix

- If the derivative of ∇f exists, we say that f is twice differentiable.
  - Write the second derivative as $D^2 f$ (or $\boldsymbol{F}$), and call it the *Hessian* of f.

$$\boldsymbol{F} = D^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\boldsymbol{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(\boldsymbol{x}) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(\boldsymbol{x}) & \frac{\partial^2 f}{\partial x_2^2}(\boldsymbol{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2}(\boldsymbol{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(\boldsymbol{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\boldsymbol{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\boldsymbol{x}) \end{bmatrix}$$





- The local optimum is mínimum (máximum) if the Hessian matrix is positive (negative) definite.
- Else it is a saddle point.

Constrained Optimization

$$
\begin{aligned}
&\text{Minimize} && f(x) \\
&\text{Subject to} && g_j(x) \geqslant 0 && \text{for } j = 1, 2, \ldots, J \\
& && h_k(x) = 0 && \text{for } k = 1, 2, \ldots, K \\
& && x = (x_1, x_2, \ldots, x_N)
\end{aligned}
$$

Lagrangian: $L(x, u, v) = f(x) - \sum_{j=1}^{J} u_j g_j(x) - \sum_{k=1}^{K} v_k h_k(x)$

Kuhn-Tucker conditions

Find vectors $x_{(N\times 1)}$, $u_{(1\times J)}$, and $v_{(1\times K)}$ that satisfy

$$\nabla f(x) - \sum_{j=1}^{J} u_j \nabla g_j(x) - \sum_{k=1}^{K} v_k \nabla h_k(x) = 0$$

$$g_j(x) \geqslant 0 \qquad \text{for } j = 1, 2, \ldots, J$$

$$h_k(x) = 0 \qquad \text{for } k = 1, 2, \ldots, K$$

$$u_j g_j(x) = 0 \qquad \text{for } j = 1, 2, \ldots, J$$

$$u_j \geqslant 0 \qquad \text{for } j = 1, 2, \ldots, J$$

Algorithms

Gradient Descent

- An algorithm for: $\min_x f(x)$

Input $x_0 \in R^n$

Step 0: set $i = 0$

Step 1: if $\nabla f(x_i) = 0$ **stop**,

else, compute ***search direction*** $h_i \in R^n$

Negative Gradient direction

Step 2: compute the ***step-size*** $\lambda_i \in \arg\min_{\lambda \geq 0} f(x_i + \lambda \cdot h_i)$

Step 3: set $x_{i+1} = x_i + \lambda_i \cdot h_i$ go to step 1

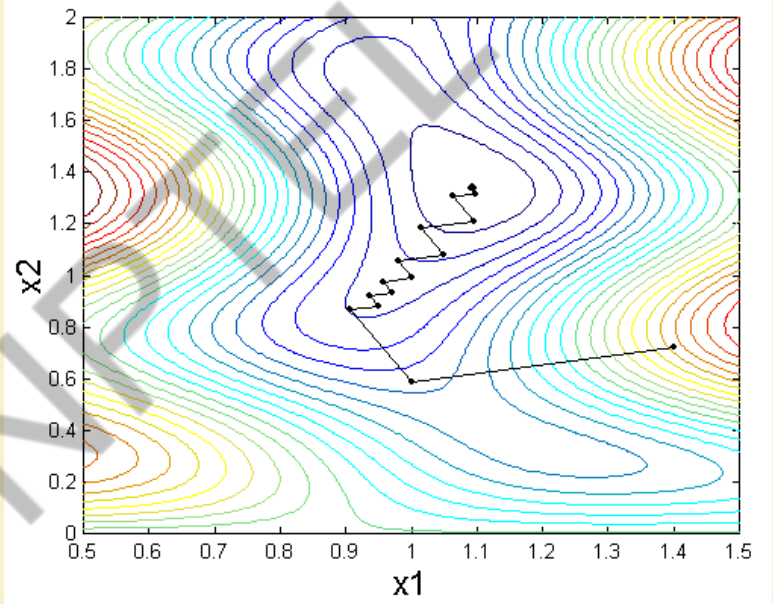
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Sourangshu Bhattacharya
Computer Science and Engg.

Gradient Descent

Given:

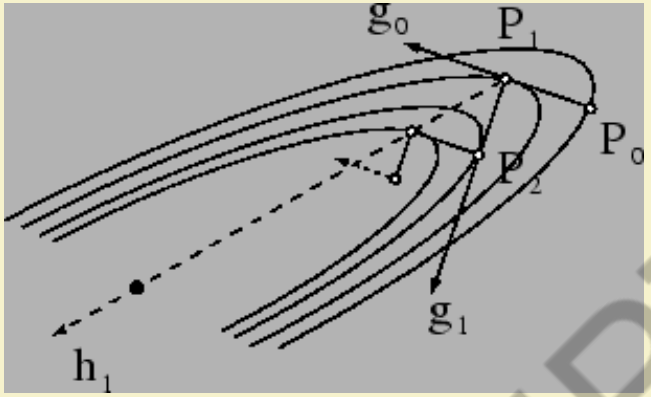
$$f(x_1, x_2) = 2\sin(1.47x_1)\sin(0.34x_2) + \sin(x_1)\sin(1.9x_2)$$

Find the minimum when x_1 is allowed to vary from 0.5 to 1.5 and x_2 is allowed to vary from 0 to 2.



Gradient Descent

- What is the problem with steepest descent?



  - We can repeat the same directions over and over…
- Wouldn't it be better if, every time we took a step, we got it right the first time?

Newton's Method

Idea: use a second-order approximation to function.

$$f(x + \Delta x) \approx f(x) + \nabla f(x)^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 f(x) \Delta x$$

Choose Δx to minimize above:

$$\Delta x = - \left[\nabla^2 f(x) \right]^{-1} \nabla f(x)$$

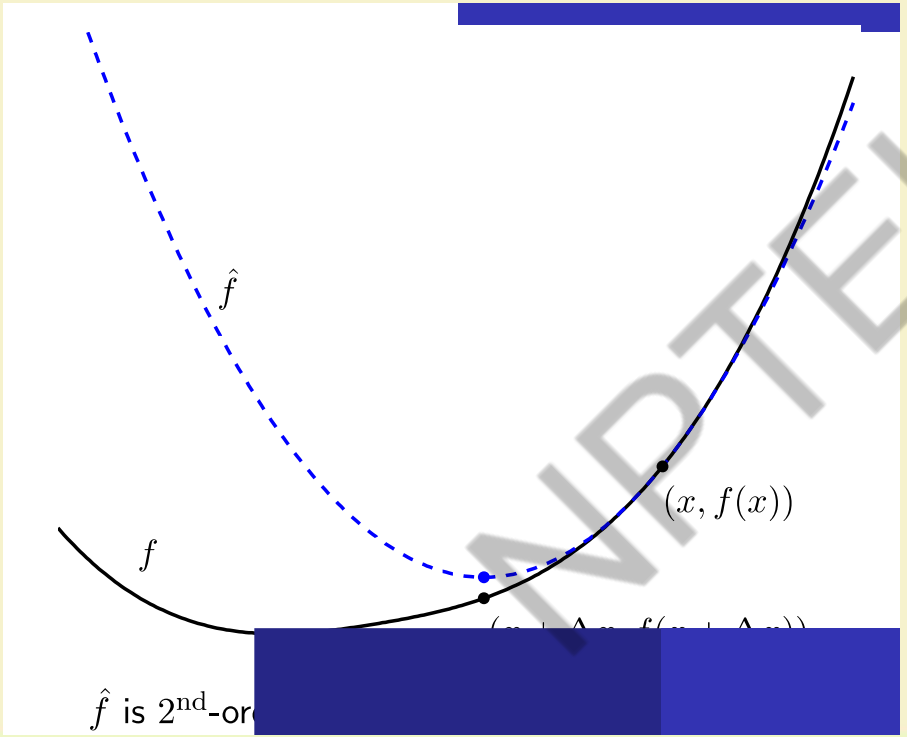
Inverse Hessian

Gradient

This is descent direction:

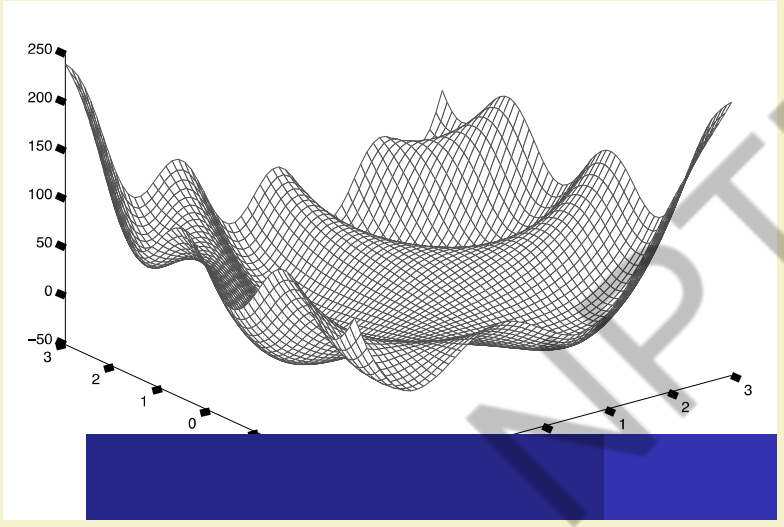
$$\nabla f(x)^T \Delta x = -\nabla f(x)^T \left[\nabla^2 f(x) \right]^{-1} \nabla f(x) < 0.$$

Newton's Method Picture



Prefer Convex Problems

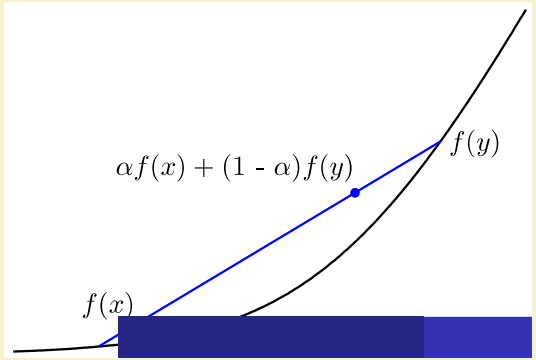
Local (non global) minima and maxima:

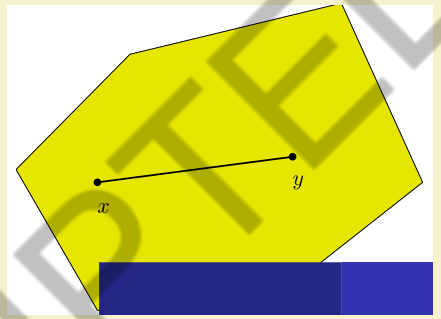


Convex Functions and Sets

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for $x, y \in \text{dom} f$ and any $a \in [0, 1]$,

$$f(ax + (1 - a)y) \leq af(x) + (1 - a)f(y)$$





A set $C \subseteq \mathbb{R}^n$ is convex if for $x, y \in C$ and any $a \in [0, 1]$,

$$ax + (1 - a)y \in C$$

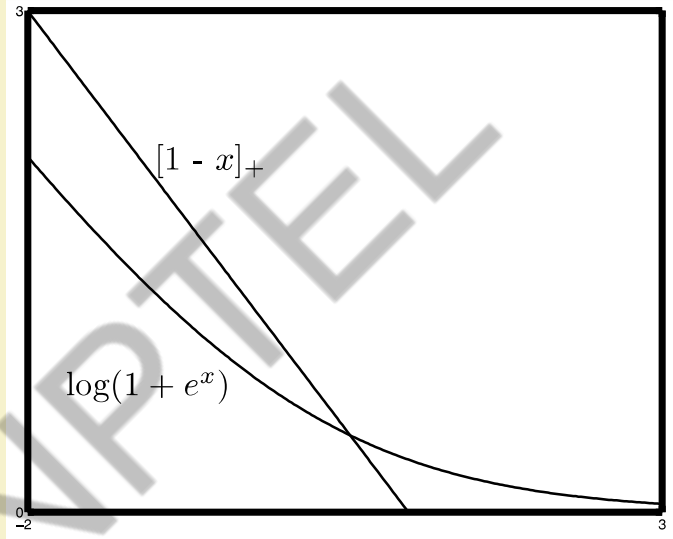
Important Convex Functions

SVM loss:

$$f(w) = \left[1 - y_i x_i^T w\right]_+$$

Binary logistic loss:

$$f(w) = \log\left(1 + \exp(-y_i x_i^T w)\right)$$

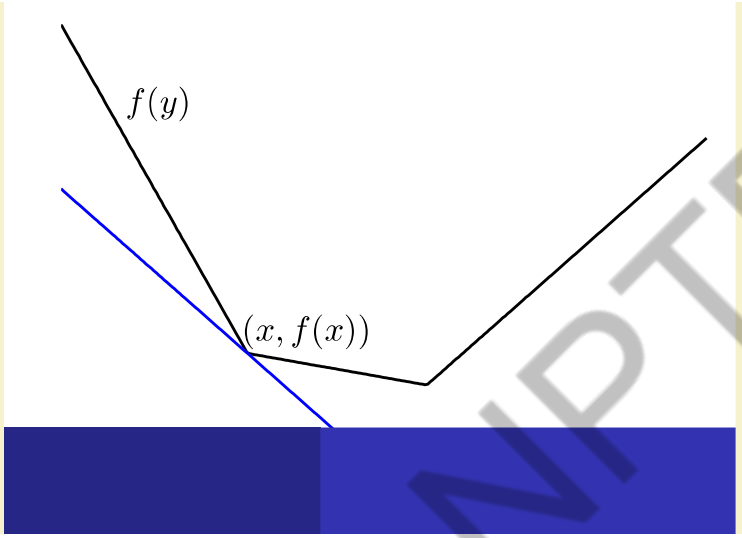


Convex Optimization Problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) && \text{(Convex function)} \\ \text{s.t.} \quad & f_i(x) \leq 0 && \text{(Convex sets)} \\ & h_j(x) = 0 && \text{(Affine)} \end{aligned}$$

Subgradient Descent Motivation

Lots of non-differentiable convex functions used in machine learning:



The *subgradient set*, or subdifferential set, $\partial f(x)$ of f at x is

$$\partial f(x) = \left\{ g : f(y) \geq f(x) + g^T (y - x) \quad \text{for all } y \right\}.$$

Subgradient Descent – Algorithm

Really, the simplest algorithm in the world. Goal:

$$\underset{x}{\text{minimize}} \; f(x)$$

Just iterate

$$x_{t+1} = x_t - \eta_t g_t$$

where η_t is a stepsize, $g_t \in \partial f(x_t)$.

Online learning and optimization

- Goal of machine learning :
  - Minimize expected loss

$$\min_h L(h) = \mathbf{E}\left[\text{loss}(h(x), y)\right]$$

given samples

$$(x_i, y_i)\ i = 1, 2...m$$

- This is Stochastic Optimization
  - Assume loss function is convex

Batch (sub)gradient descent for ML

- Process all examples together in each step

$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial L(w, x_i, y_i)}{\partial w} \right)$$

where L is the regularized loss function

- Entire training set examined at each step
- Very slow when *n* is very large

Stochastic (sub)gradient descent

- "Optimize" one example at a time
- Choose examples randomly (or reorder and choose in order)
  - Learning representative of example distribution

for $i = 1$ to n:

$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \frac{\partial L(w, x_i, y_i)}{\partial w}$$

where L is the regularized loss function

Stochastic (sub)gradient descent

for $i = 1$ to n:

$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \frac{\partial L(w, x_i, y_i)}{\partial w}$$

where L is the regularized loss function

- Equivalent to online learning (the weight vector *w* changes with every example)
- Convergence guaranteed for convex functions (to local minimum)

References:

- R. Fletcher **Practical Methods of Optimization**, 2nd Edition. *John Wiley & Sons, Inc.* July 2000.
- Stephen Boyd and Lieven Vandenberghe. **Convex Optimization** *Cambridge University Press 2009*.
- Wikipedia.

Thank You!!



IIT KHARAGPUR

IIT GANDHINAGAR


NPTEL ONLINE CERTIFICATION COURSES

Scalable Data Science

Lecture 5: Background on Machine Learning

Sourangshu Bhattacharya
Computer Science and Engineering
IIT KHARAGPUR

In this review

- Outline:
  - What is Machine Learning ?
  - Supervised learning
  - Linear Regression
  - Generalization
  - Classification
  - Clustering

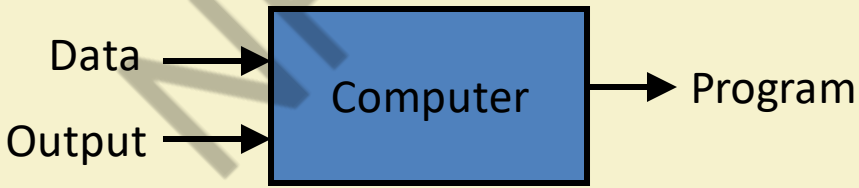
What Is Machine Learning?

- Automating automation
- Getting computers to program themselves
- Writing software is the bottleneck
- Let the data do the work instead!

What Is Machine Learning?

Traditional Programming

Machine Learning



Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- [Your favorite area]

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
    - Representation / Model
    - Evaluation / Metric / Loss
    - Optimization / Estimation

Representation / Model

The function or set of equations, describing how input and outputs of the problem are related. Generally, equations have parameters.

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Evaluation / Metric

Describes the way of measuring the quality of output given all the inputs, (including the true output labels).

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Margin
- Entropy
- K-L divergence
- Etc.

Optimization / Estimation

Provides a method for finding the values of parameters which achieve the best performance on the supplied dataset.

- Closed form equations – e.g.: linear regression
- Sampling based techniques – e.g. collapsed Gibbs sampling for LDA.
- Combinatorial optimization - E.g.: Grid search for hyperparameters
- Convex optimization - E.g.: Stochastic Gradient descent
- Constrained optimization - E.g.: Linear programming

Types of Learning

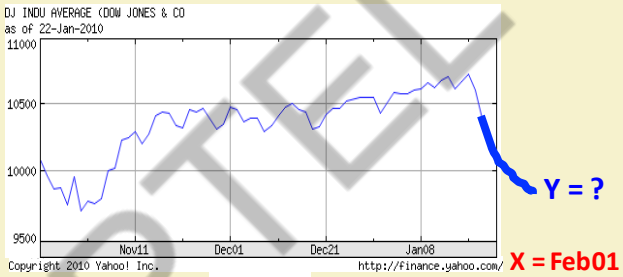
- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

Supervised Learning

Supervised Learning

Goal: Construct a **predictor** $f : X \rightarrow Y$ to minimize loss function (performance measure)

Sports
Science
News



Classification:

$$P(f(X) \neq Y)$$

Probability of Error

Regression:

$$\mathbb{E}[(f(X) - Y)^2]$$

Mean Squared Error

Regression algorithms

Linear Regression

Replace Expectation with Empirical Mean

Optimal predictor: $f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$

Empirical Minimizer: $\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$

Empirical mean

Law of Large Numbers:

$$\frac{1}{n} \sum_{i=1}^{n} [\text{loss}(Y_i, f(X_i))] \xrightarrow{n \to \infty} \mathbb{E}_{XY} [\text{loss}(Y, f(X))]$$

Restrict class of predictors

Optimal predictor: $f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$

Empirical Minimizer: $\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$

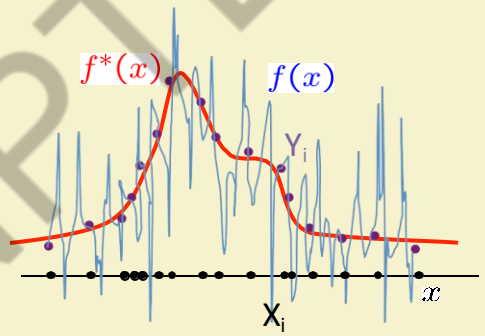
Class of predictors

Why?

Overfitting!

Empiricial loss minimized by any function of the form

$$f(x) = \begin{cases} Y_i, & x = X_i \text{ for } i = 1, \ldots, n \\ \text{any value}, & \text{otherwise} \end{cases}$$



Restrict class of predictors

Optimal predictor: $f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$

Empirical Minimizer: $\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$

Class of predictors

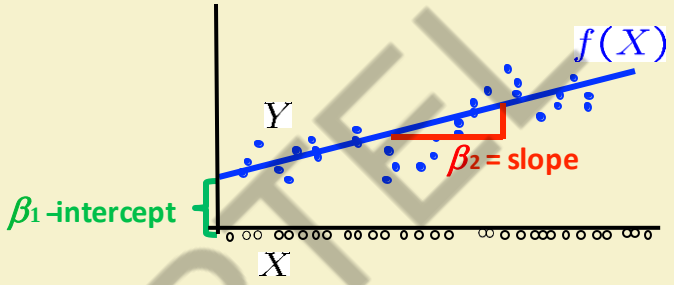
- Class of Linear functions
- Class of Polynomial functions
- Class of nonlinear functions

Linear Regression

$$\hat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

Least Squares Estimator

$\mathcal{F}_L$ –Class of Linear functions



Uni–variate case:

$$f(X) = \beta_1 + \beta_2 X$$

Multi–variate case:

$$f(X) = f(X^{(1)}, \ldots, X^{(p)}) = \beta_1 X^{(1)} + \beta_2 X^{(2)} + \cdots + \beta_p X^{(p)}$$

$$= X\beta \qquad \text{where} \quad X = [X^{(1)} \ldots X^{(p)}], \quad \beta = [\beta_1 \ldots \beta_p]^T$$

Least Squares Estimator

$$\hat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

$$f(X_i) = X_i\beta$$

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (X_i\beta - Y_i)^2$$

$$\hat{f}_n^L(X) = X\hat{\beta}$$

$$= \arg\min_{\beta} \frac{1}{n} (\mathbf{A}\beta - \mathbf{Y})^T (\mathbf{A}\beta - \mathbf{Y})$$

$$\mathbf{A} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1^{(1)} & \dots & X_1^{(p)} \\ \vdots & \ddots & \vdots \\ X_n^{(1)} & \dots & X_n^{(p)} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_n \end{bmatrix}$$

Least Squares Estimator

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

$$J(\beta) = (\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y})$$

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0$$

Normal Equations

$$(\mathbf{A}^T\mathbf{A})\hat{\beta} = \mathbf{A}^T\mathbf{Y}$$

p x p p x 1 p x 1

If $(\mathbf{A}^T\mathbf{A})$ is invertible,

$$\hat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \qquad \hat{f}_n^L(X) = X\hat{\beta}$$

When is $(\mathbf{A}^T\mathbf{A})$ invertible ?
Recall: Full rank matrices are invertible.

What if $(\mathbf{A}^T\mathbf{A})$ is not invertible ?

Non-linear basis functions

- What type of functions can we use?
- A few common examples:

- Polynomial: $\phi_j(x) = x^j$ for j=0 … n

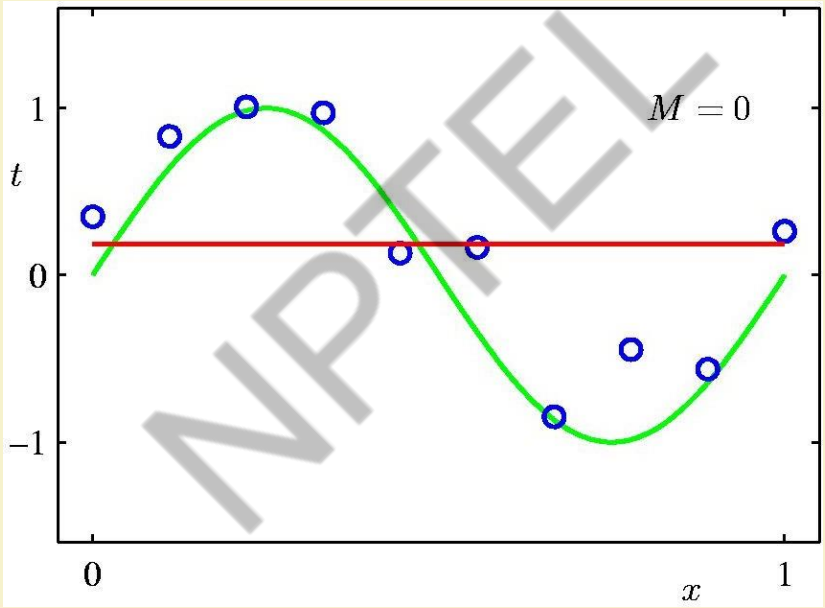
- Gaussian: $\phi_j(x) = \frac{(x - \mu_j)}{2\sigma_j^2}$

- Sigmoid: $\phi_j(x) = \frac{1}{1 + \exp(-s_j x)}$

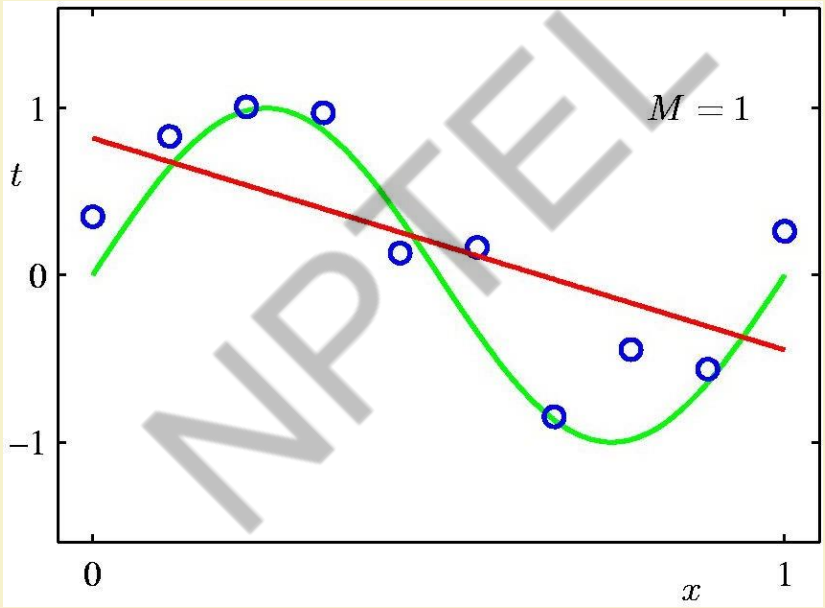
- Logs: $\phi_j(x) = \log(x + 1)$

Any function of the input values can be used. The solution for the parameters of the regression remains the same.

0th Order Polynomial

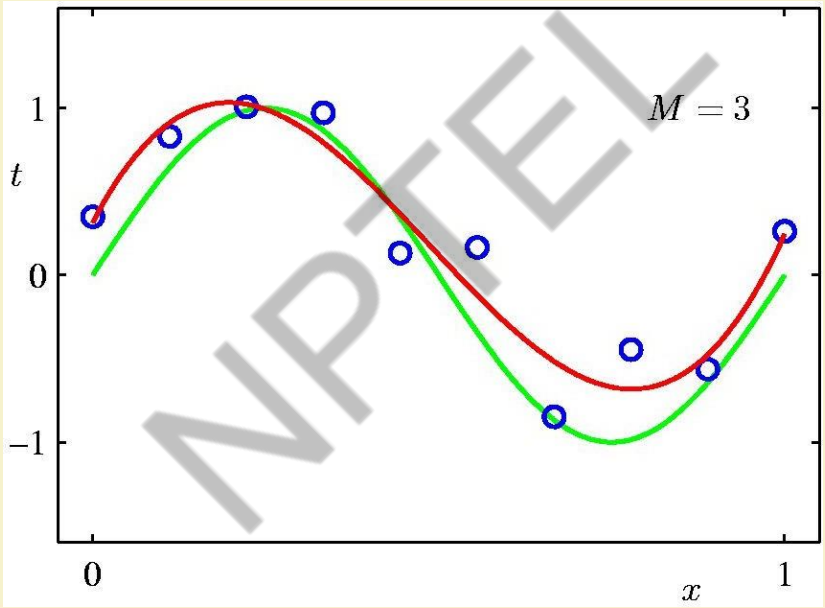


1st Order Polynomial

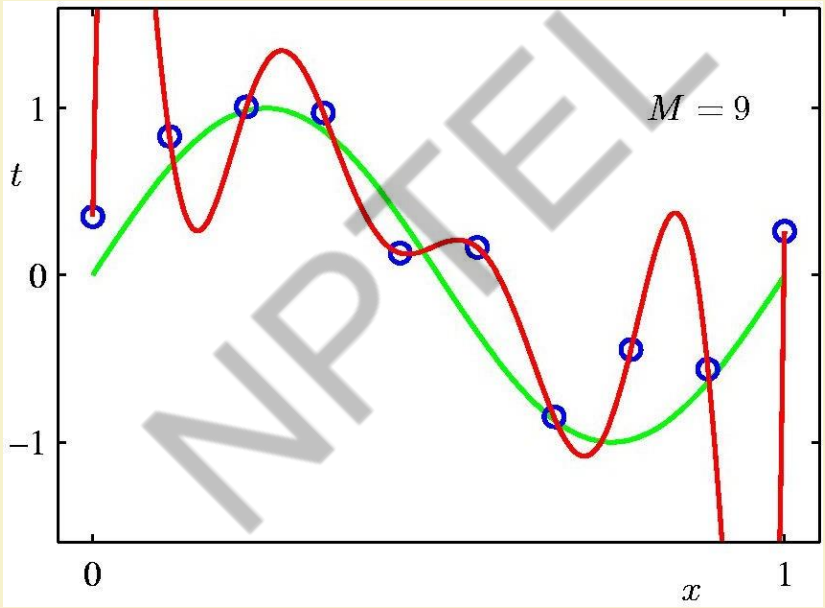


Slide courtesy of William Cohen

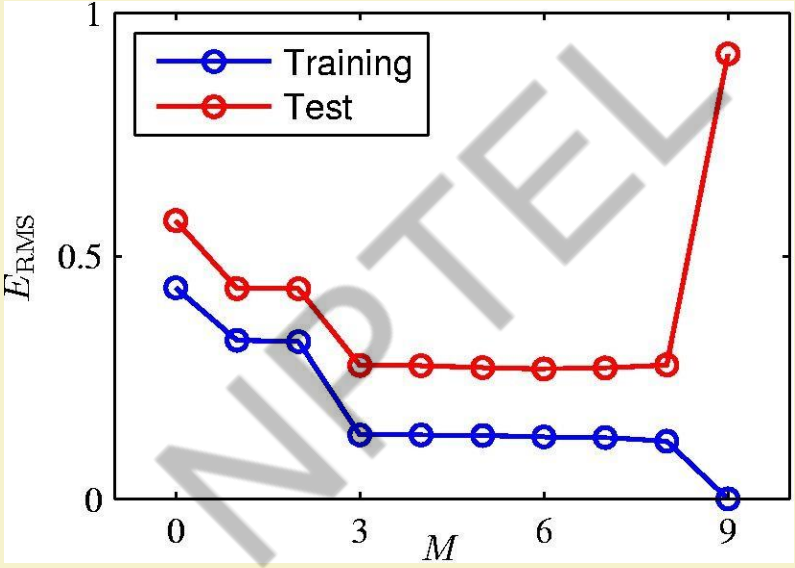
3rd Order Polynomial



9th Order Polynomial



Over-fitting



Root-Mean-Square (RMS) Error

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Regularization

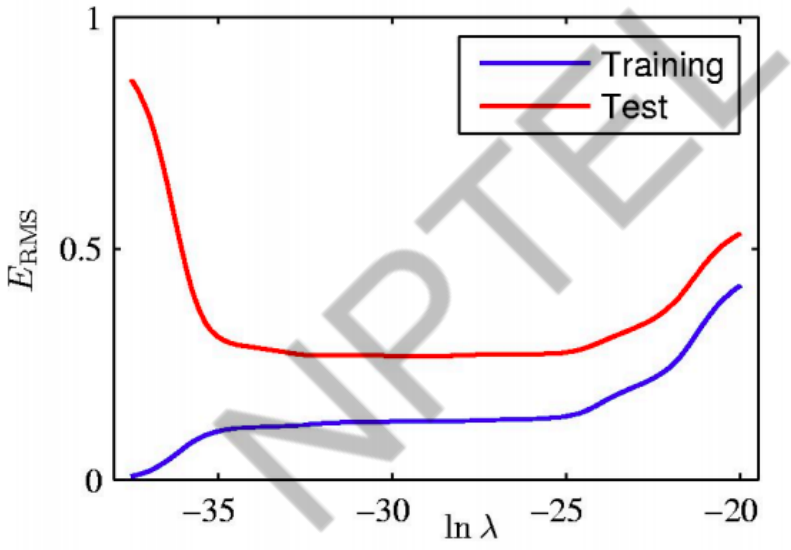
Penalize large coefficient values

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \frac{1}{2}\sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Slide courtesy of William Cohen

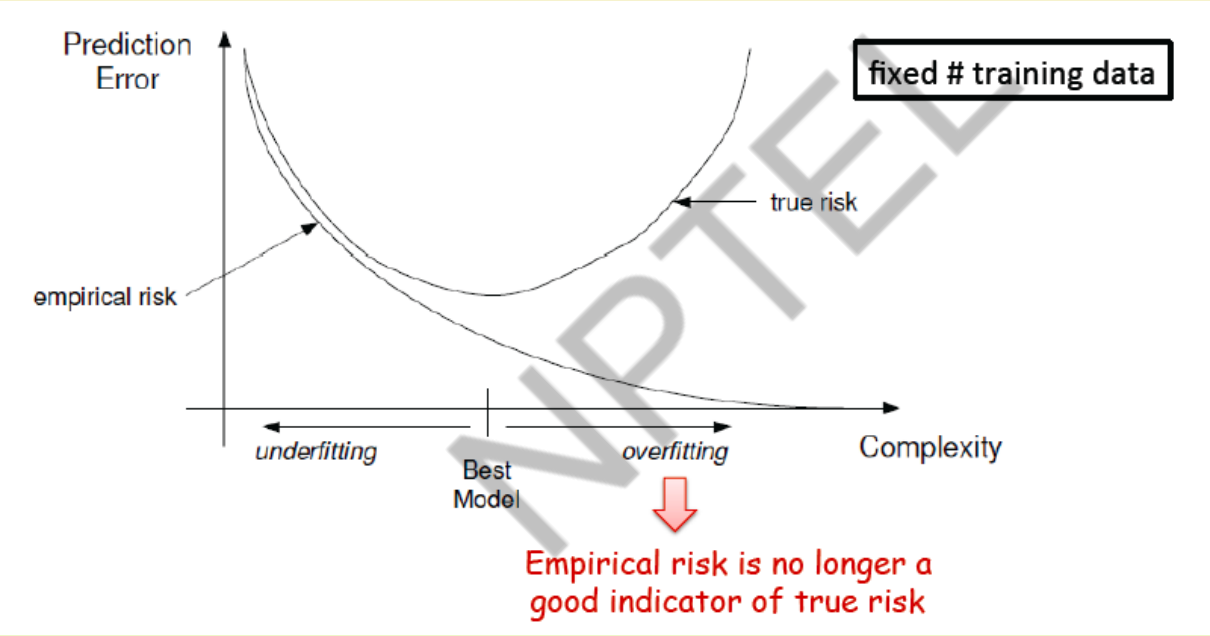
Regularization

9th Order Polynomial



Effect of Model Complexity

- If we allow very complicated predictors, we could overfit the training data.



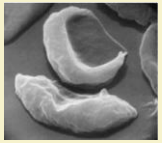
Discrete and Continuous Labels

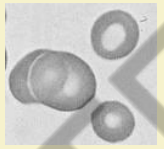
Classification

X = Document

Sports
Science
News

Y = Topic





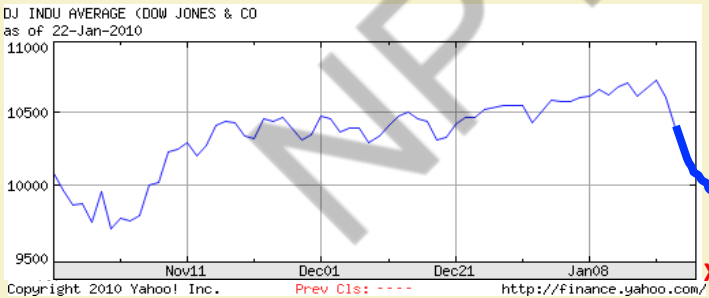
X = Cell Image

Anemic cell
Healthy cell

Y = Diagnosis

Regression

Stock Market Prediction



Y = ?

X = Feb01

An example application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

From Linear to Logistic Regression

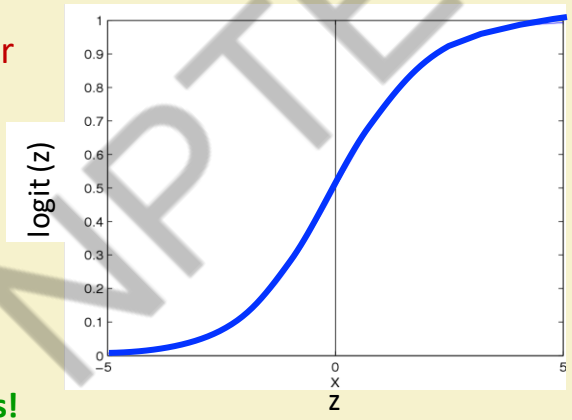
Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Logistic function applied to a linear function of the data

Logistic function (or Sigmoid):

$$\frac{1}{1+exp(-z)}$$



Features can be discrete or continuous!

Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

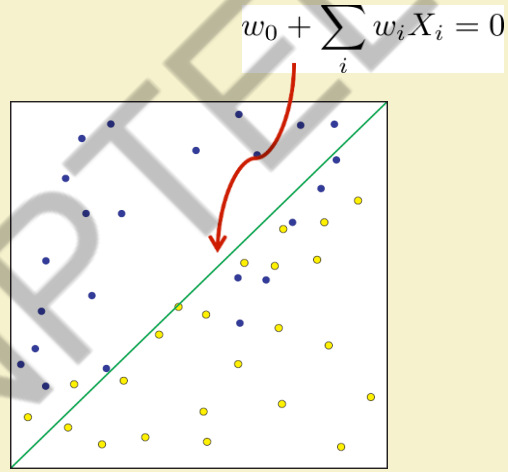
$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Decision boundary:

$$P(Y=0|X) \underset{1}{\overset{0}{\gtrless}} P(Y=1|X)$$

$$w_0 + \sum_i w_i X_i \underset{1}{\overset{0}{\gtrless}} 0$$

(Linear Decision Boundary)



Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0+\sum_i w_i X_i)}$$

$$\Rightarrow P(Y=0|X) = \frac{\exp(w_0+\sum_i w_i X_i)}{1+\exp(w_0+\sum_i w_i X_i)}$$

$$\Rightarrow \frac{P(Y=0|X)}{P(Y=1|X)} = \exp(w_0+\sum_i w_i X_i) \underset{1}{\overset{0}{\gtrless}} 1$$

$$\Rightarrow w_0+\sum_i w_i X_i \underset{1}{\overset{0}{\gtrless}} 0$$

Other classifiers

- Naïve Bayes
- Support vector Machines
- Neural Networks.
- K- nearest neighbors.
- Random Forests
- etc.

Unsupervised Learning

The Goals of Unsupervised Learning

- The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We discuss two methods:
  - *Latent Semantic Indexing*, a dimensionality reduction technique used for data visualization or data pre-processing before supervised techniques are applied, and
  - *Clustering*, a broad class of methods for discovering unknown subgroups in data.

Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other,
- It make this concrete, we must define what it means for two or more observations to be *similar* or *different*.
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

K-Means

- Assumes datapoints are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\vec{\mu}(c) = \frac{1}{|c|} \sum_{\vec{x} \in c} \vec{x}$$

- Reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

K-Means Algorithm

Select K random datapoints $\{s_1, s_2, \ldots s_K\}$ as seeds.

Until clustering *converges* (or other stopping criterion):

For each datapoint d_i:

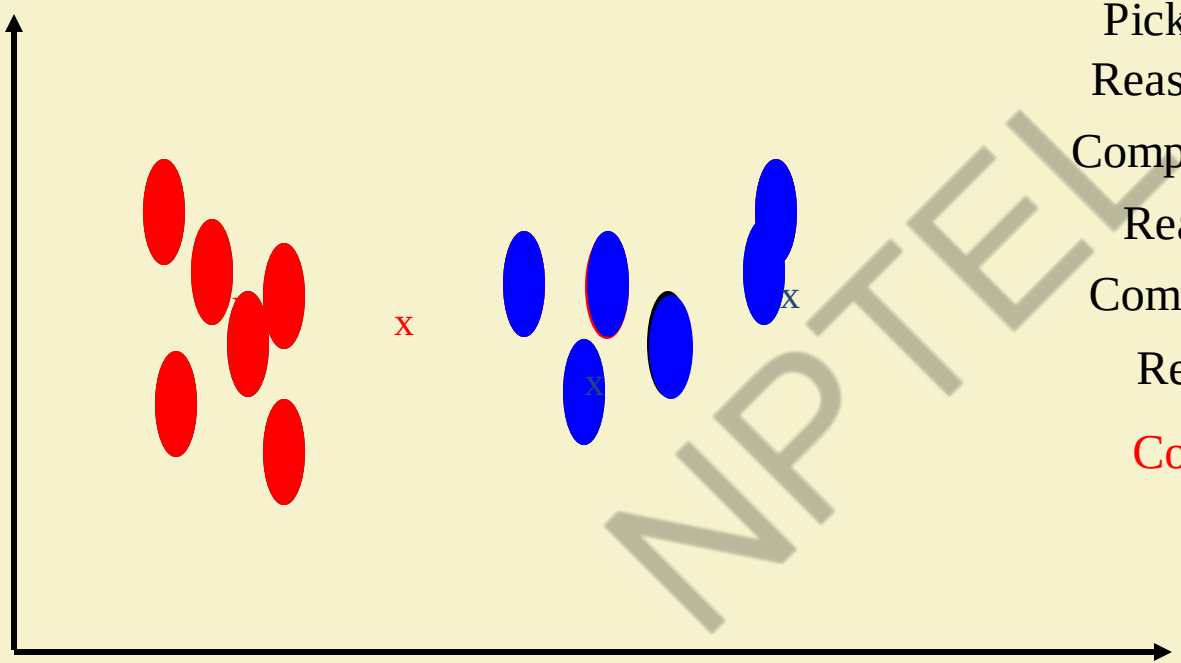
Assign d_i to the cluster c_j such that $dist(x_i, s_j)$ is minimal.

(*Next, update the seeds to the centroid of each cluster*)

For each cluster c_j

$s_j = \mu(c_j)$

K Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

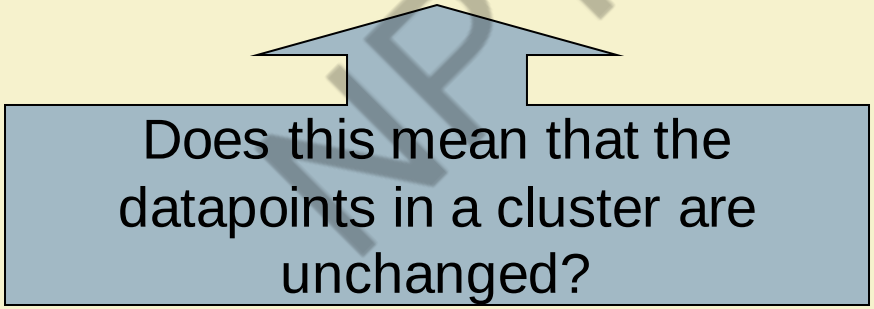
Compute centroids

Reassign clusters

Converged!

Termination conditions

- Several possibilities, e.g.,
  - A fixed number of iterations.
  - Partition unchanged.
  - Centroid positions don't change.



Convergence of K-Means

- Define goodness measure of cluster *k* as sum of squared distances from cluster centroid:
  - $G_k = \Sigma_i (d_i - c_k)^2$ (sum over all d_i in cluster *k*)
- $G = \Sigma_k G_k$
- Reassignment monotonically decreases G since each vector is assigned to the closest centroid.

Convergence of K-Means

- Recomputation monotonically decreases each G_k since (m_k is number of members in cluster k):
  - $\Sigma\ (d_i - a)^2$ reaches minimum for:
  - $\Sigma\ -2(d_i - a) = 0$
  - $\Sigma\ d_i = \Sigma\ a$
  - $m_K\ a = \Sigma\ d_i$
  - $a = (1/\ m_k)\ \Sigma\ d_i = c_k$
- K-means typically converges quickly

Time Complexity

- Computing distance between two datapoints is O*(M)* where *M* is the dimensionality of the vectors.
- Reassigning clusters: O*(KN)* distance computations, or O*(KNM).*
- Computing centroids: Each datapoint gets added once to some centroid: O*(NM).*
- Assume these two steps are each done once for *I* iterations: O*(IKNM).*

References:

- Christopher M. Bishop. **Pattern Recognition and Machine Learning.** *Springer-Verlag New York Inc.; 1st ed. 2006.*

- Many other books.

- Wikipedia.

Thank You!!