MATH (160)

Mathematics for Data Analytics and Decision Making

Jesus De Loera

UC Davis, Mathematics

Monday, March 28, 2011

Syllabus, Class Schedule, Office Hours, Textbook, etc.

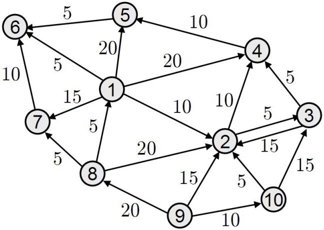
See course website.

`https://www.math.ucdavis.edu/~deloera/TEACHING/MATH160/`

Also I will use your UC Davis email for announcements and quizzes!

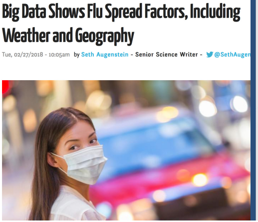
MATHEMATICAL MODELS!!!
That use data to make intelligent decisions!!

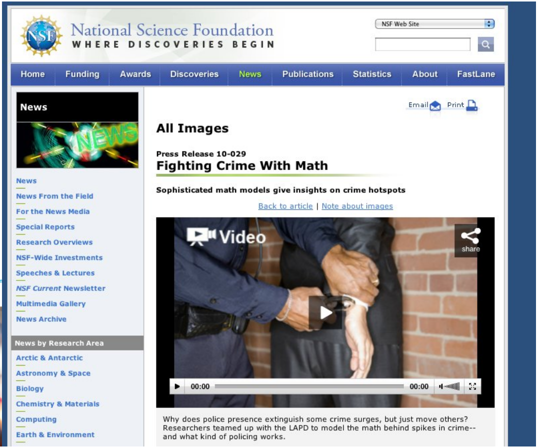
- How does Google's search engine and GPS routing work?



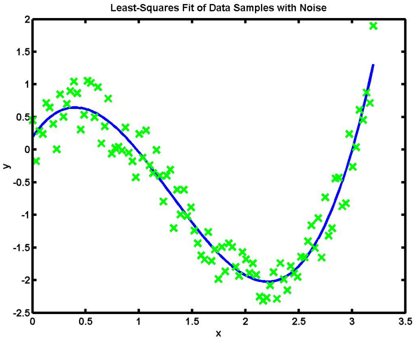
- The engine behind the engines is all MATHEMATICS!!!

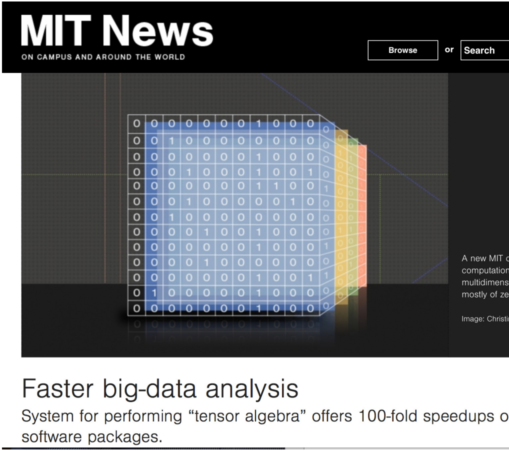
Today the world has abundant data of all sorts, we need to go FROM DATA TO DECISIONS!





and MATH is the only way we can analyze it to extract relevant information.





By the end of this course, you should make BETTER (quantitative) DECISIONS!!

By the end of this course, you should make BETTER (quantitative) DECISIONS!!

MULTITRANS is a fictional bus company that is run by a student association in an unnamed college town with partial funding from the city.

(It started out in 1972 with a fleet of historic triple-deck buses from Moscow, but has added more modern buses since.)

During most of the day, though, all scheduled buses are completely empty.

A scientific study was conducted in 2010 to find out why this is the case. The result of the study was:

> *[...] 85% of the representative sample of 1256 potential users of MULTITRANS replied that "I would use it regularly instead of using my bike or car, but the MULTITRANS schedule [is suboptimal]." [...]*

MULTITRANS hires YOU as a consultant to help them improve the schedule.

What are the next steps you need to take to solve the challenge?

Part I of the course:

LINEAR ALGEBRA MODELS

Least-squares applications to modeling.

What is the best team? What is the best candidate? etc: The science of ranking

-Say we are given data about teams with game scores, how could we predict the best team of the season?
-Say we are given voters information about candidates, how can we rank them? (arrange them in order of importance).
-Say we are given a list of movies with user preferences (prefer to star wars v.s. star trek?)

Goal

We will see five ways to make predictions of WHO IS NUMBER ONE using data of comparisons and solving linear equations.

The models depend on how much information we have on the situation. We hope that using these five methods allow users to determine who is the most viable candidate and show the user that different methods may provide different results.

Two old “COUNTING” methods

Plurality

Candidate rated number one is the one with the most wins or the most votes.

The plurality method was originally used in determining a winner in voting systems. It soon became one of the most commonly used methods for selecting a winner.
But what if we knew an ordered list of candidates, one for each voter? We could use more sophisticated methods!! Borda Count is a method that dates back to 1770

Borda Count

Count the number of times a candidate appears in *i*th position and apply the following set of weights as follows (here n is the total number of candidates)
$(n-1)$ points are given to first place, $(n-k)$ to k-th place, and 0 is given to last place. The tallies are summed and the winner is the candidate with the highest sum.

Voter	HC	BS	JK	TC	DT
Voter 1	5	4	3	2	1
Voter 2	5	4	3	2	1
Voter 3	4	5	3	1	2
Voter 4	2	5	1	3	4
Voter 5	3	5	2	1	4
Voter 6	4	5	2	1	3
Voter 7	1	5	3	4	2
Voter 8	5	4	3	1	2
Voter 9	4	5	2	3	1
Voter 10	4	5	2	1	3

Table: Data set (only 10 voters).

1st Place votes	HC	BS	JK	TC	DT
Votes	3	7	0	0	0

Table: Plurality Scores.

	HC	BS	JK	TC	DT
Average Rating	3.7	4.7	2.4	1.9	2.3

Table: Average Scores.

Ranking	HC	BS	JK	TC	DT
1st Place	3	7	0	0	0
2nd Place	4	3	0	1	1
3rd Place	1	0	5	2	2
4th Place	1	0	4	2	4
5th Place	1	0	1	5	3

Table: Borda Rankings.

Candidate	Score
Hillary Clinton	27
Bernie Sanders	37
John Kasich	14
Ted Cruz	9
Donald Trump	11

Table: Borda Scores.

Massey Ranking

- The fundamental idea: Find the rankings or score values of team r_i by noting $r_i - r_j = y_{ij}$ where y_{ij} is the margin of victory of team i over j .
- For each game played there is one such equation, and then there is a system of linear equations. The matrix X is $m \times n$ with $m =$number of games played and $n =$ number of teams.

$$Xr = y.$$

Massey Ranking

- The fundamental idea: Find the rankings or score values of team r_i by noting $r_i - r_j = y_{ij}$ where y_{ij} is the margin of victory of team i over j .
- For each game played there is one such equation, and then there is a system of linear equations. The matrix X is $m \times n$ with $m =$number of games played and $n =$ number of teams.

$$Xr = y.$$

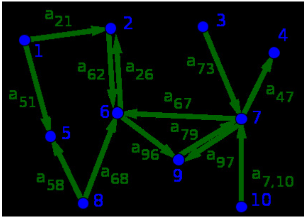
- Each row of the matrix X is nearly all zeros, except a 1 and a -1.

Massey Ranking

- The fundamental idea: Find the rankings or score values of team r_i by noting $r_i - r_j = y_{ij}$ where y_{ij} is the margin of victory of team i over j .
- For each game played there is one such equation, and then there is a system of linear equations. The matrix X is $m \times n$ with $m =$number of games played and $n =$ number of teams.

$$Xr = y.$$

- Each row of the matrix X is nearly all zeros, except a 1 and a -1.
- It is the transpose of a **node-arc incidence matrix of a network** Typically the number of games is very large, larger than the number of teams.



- Note: y_{ij} can be interpreted in many ways: can mean traffic levels, to rank websites, y_{ij} can be number of in-links or outlinks.

- We may not have an exact solution of $Xr = y$, but we can solve the associated **least-squares problem**.

- We may not have an exact solution of $Xr = y$, but we can solve the associated **least-squares problem**.
- Recall, solving the least squares problem $Xr = y$ same as solving the system of equations: $X^T Xr = Mr = X^T y = d$.

$$\begin{pmatrix} t_1 & -m_{12} & \cdots & -m_{1n} \\ -m_{21} & t_2 & \cdots & -m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & -m_{n2} & \cdots & t_n \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

- We may not have an exact solution of $Xr = y$, but we can solve the associated **least-squares problem**.
- Recall, solving the least squares problem $Xr = y$ same as solving the system of equations: $X^T Xr = Mr = X^T y = d$.

$$\begin{pmatrix} t_1 & -m_{12} & \cdots & -m_{1n} \\ -m_{21} & t_2 & \cdots & -m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & -m_{n2} & \cdots & t_n \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

- Here t_i is the total number of games, m_{ij} is the number of games team i played against j and d_i is the score differential of the team.

- We may not have an exact solution of $Xr = y$, but we can solve the associated **least-squares problem**.
- Recall, solving the least squares problem $Xr = y$ same as solving the system of equations: $X^T Xr = Mr = X^T y = d$.

$$\begin{pmatrix} t_1 & -m_{12} & \cdots & -m_{1n} \\ -m_{21} & t_2 & \cdots & -m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & -m_{n2} & \cdots & t_n \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

- Here t_i is the total number of games, m_{ij} is the number of games team i played against j and d_i is the score differential of the team.
- The elements of $\vec{d}$ are calculated as follows:

$$d_i = \sum_{1}^{x} (n_i - n_j) \ = (n_i - n_1) + (n_i - n_2) + \cdots + (n_i - n_x)$$

- We may not have an exact solution of $Xr = y$, but we can solve the associated **least-squares problem**.
- Recall, solving the least squares problem $Xr = y$ same as solving the system of equations: $X^T Xr = Mr = X^T y = d$.

$$\begin{pmatrix} t_1 & -m_{12} & \cdots & -m_{1n} \\ -m_{21} & t_2 & \cdots & -m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & -m_{n2} & \cdots & t_n \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

- Here t_i is the total number of games, m_{ij} is the number of games team i played against j and d_i is the score differential of the team.
- The elements of $\vec{d}$ are calculated as follows:

$$d_i = \sum_{1}^{x} (n_i - n_j) \; = (n_i - n_1) + (n_i - n_2) + \cdots + (n_i - n_x)$$

- Outputs the ratings of candidates by approximately solving the system of linear equalities (or least-squares!). We are trying to find the ratings r_i.

	HC	BS	JK	TC	DT
HC	40	-10	-10	-10	-10
BS	-10	40	-10	-10	-10
Jk	-10	-10	40	-10	-10
TC	-10	-10	-10	40	-10
DT	-10	-10	-10	-10	40

Table: Massey Matrix.

	HC	BS	JK	TC	DT
HC	40	-10	-10	-10	-10
BS	-10	40	-10	-10	-10
Jk	-10	-10	40	-10	-10
TC	-10	-10	-10	40	-10
DT	-10	-10	-10	-10	40

Table: Massey Matrix.

Games	HC	BS	JK	TC	DT
1 - 4 (V_1)	10	5	0	-5	-10
5 - 8 (V_2)	10	5	0	-5	-10
9 - 12 (V_3)	5	10	0	-10	-5
13 - 16 (V_4)	-5	10	-10	0	5
17 - 20 (V_5)	0	10	-5	-10	5
21 - 24 (V_6)	5	10	-5	-10	0
25 - 28 (V_7)	-10	10	0	5	-5
29 - 32 (V_8)	10	5	0	-10	-5
33 - 36 (V_9)	5	10	-5	0	-10
37 - 40 (V_{10})	5	10	-5	-10	0
Total	35	85	-30	-55	-35

Table: Candidates' Point Differentials: how many times candidate won or lost to another

	HC	BS	JK	TC	DT
HC	40	-10	-10	-10	-10
BS	-10	40	-10	-10	-10
Jk	-10	-10	40	-10	-10
TC	-10	-10	-10	40	-10
DT	-10	-10	-10	-10	40

Table: Massey Matrix.

Games	HC	BS	JK	TC	DT
1 - 4 (V_1)	10	5	0	-5	-10
5 - 8 (V_2)	10	5	0	-5	-10
9 - 12 (V_3)	5	10	0	-10	-5
13 - 16 (V_4)	-5	10	-10	0	5
17 - 20 (V_5)	0	10	-5	-10	5
21 - 24 (V_6)	5	10	-5	-10	0
25 - 28 (V_7)	-10	10	0	5	-5
29 - 32 (V_8)	10	5	0	-10	-5
33 - 36 (V_9)	5	10	-5	0	-10
37 - 40 (V_{10})	5	10	-5	-10	0
Total	35	85	-30	-55	-35

Table: Candidates' Point Differentials: how many times candidate won or lost to another

There is a problem with M: it is not full rank. Without a full rank we will not be able to find a unique solution.
Therefore we replace the entire last row of Matrix M with 1's and replace the last element of $\vec{p}$ with a 0. This yields the system

There is a problem with M: it is not full rank. Without a full rank we will not be able to find a unique solution.
Therefore we replace the entire last row of Matrix M with 1's and replace the last element of $\vec{p}$ with a 0. This yields the system

$$\begin{bmatrix} 40 & -10 & -10 & -10 & -10 \\ -10 & 40 & -10 & -10 & -10 \\ -10 & -10 & 40 & -10 & -10 \\ -10 & -10 & -10 & 40 & -10 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix} = \begin{bmatrix} 35 \\ 85 \\ -30 \\ -55 \\ 0 \end{bmatrix}$$

The unique solution gives now a ranking

The unique solution gives now a ranking
$\vec{r}$ =
$$\begin{bmatrix} 0.700 \\ 1.700 \\ -0.600 \\ -1.100 \\ -0.700 \end{bmatrix}$$
The largest rating of 1.700 corresponds to Bernie Sanders. Therefore, Bernie Sanders is the winner for Massey's Method.

The unique solution gives now a ranking

$\vec{r}$ =

$$\begin{bmatrix} 0.700 \\ 1.700 \\ -0.600 \\ -1.100 \\ -0.700 \end{bmatrix}$$

The largest rating of 1.700 corresponds to Bernie Sanders. Therefore, Bernie Sanders is the winner for Massey's Method.

Rank	Candidate
1	Bernie Sanders
2	Hillary Clinton
3	John Kasich
4	Donald Trump
5	Ted Cruz

Table: Ranked Candidates.

Colley's method

- Colley's method (invented in 2001 by Astrophysicists Wesley Colley), looks very similar but comes from a different intuition!

Colley's method

- Colley's method (invented in 2001 by Astrophysicists Wesley Colley), looks very similar but comes from a different intuition!
- INITIAL IDEA: Winning percentage score of team $i = r_i = \frac{w_i}{t_i}$. Where w_i is number of wins, t_i number of games played.

Colley's method

- Colley's method (invented in 2001 by Astrophysicists Wesley Colley), looks very similar but comes from a different intuition!
- INITIAL IDEA: Winning percentage score of team $i = r_i = \frac{w_i}{t_i}$. Where w_i is number of wins, t_i number of games played.
- Problems: There are ties, the strength of an opponent is not represented (defeating a weak team not the same as defeating a champion).

Colley's method

- Colley's method (invented in 2001 by Astrophysicists Wesley Colley), looks very similar but comes from a different intuition!
- INITIAL IDEA: Winning percentage score of team $i = r_i = \frac{w_i}{t_i}$. Where w_i is number of wins, t_i number of games played.
- Problems: There are ties, the strength of an opponent is not represented (defeating a weak team not the same as defeating a champion).
- BETTER IDEA: Fix the problems with a small modification:

$$r_i = \frac{1 + w_i}{2 + t_i}$$

Colley's method

- Colley's method (invented in 2001 by Astrophysicists Wesley Colley), looks very similar but comes from a different intuition!
- INITIAL IDEA: Winning percentage score of team $i = r_i = \frac{w_i}{t_i}$. Where w_i is number of wins, t_i number of games played.
- Problems: There are ties, the strength of an opponent is not represented (defeating a weak team not the same as defeating a champion).
- BETTER IDEA: Fix the problems with a small modification:

$$r_i = \frac{1 + w_i}{2 + t_i}$$

- Intuitively all teams start with score $1/2$ at the beginning and things change. Denote l_i number of games lost by i.

$$w_i = \frac{w_i - l_i}{2} + \frac{w_i + l_i}{2} = \frac{w_i - l_i}{2} + \frac{t_i}{2} = \frac{w_i - l_i}{2} + \sum_{j=1}^{t_i} 1/2$$

Thus

$$\sum_{j=1}^{t_i} 1/2 \sim \sum_{j \text{ opponent of } i} r_j \quad \text{and} \quad w_i \sim \frac{w_i - l_i}{2}$$

Colley's method continued

- We get

$$r_i = \frac{1 + \frac{w_i - l_i}{2} + \sum r_j}{2 + t_i}$$

Colley's method continued

- We get
$$r_i = \frac{1 + \frac{w_i - l_i}{2} + \sum r_j}{2 + t_i}$$
- Rewriting in terms of linear equations and solving for the ratings r_k's, we get a matrix C

$$C = \begin{pmatrix} t_1 + 2 & -n_{12} & \cdots & -n_{1n} \\ -n_{21} & t_2 + 2 & \cdots & -n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -n_{n1} & -n_{n2} & \cdots & t_n + 2 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Colley's method continued

- We get

$$r_i = \frac{1 + \frac{w_i - l_i}{2} + \sum r_j}{2 + t_i}$$

- Rewriting in terms of linear equations and solving for the ratings r_k's, we get a matrix C

$$C = \begin{pmatrix} t_1 + 2 & -n_{12} & \cdots & -n_{1n} \\ -n_{21} & t_2 + 2 & \cdots & -n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -n_{n1} & -n_{n2} & \cdots & t_n + 2 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- Where we are trying to find the rating r_i and t_i is the total number of games, $-n_{ij}$ is the number of games team i played against j and $b_i = 1 + \frac{1}{2}(w_i - l_i)$.

Colley's method continued

- We get

$$r_i = \frac{1 + \frac{w_i - l_i}{2} + \sum r_j}{2 + t_i}$$

- Rewriting in terms of linear equations and solving for the ratings r_k's, we get a matrix C

$$C = \begin{pmatrix} t_1 + 2 & -n_{12} & \cdots & -n_{1n} \\ -n_{21} & t_2 + 2 & \cdots & -n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -n_{n1} & -n_{n2} & \cdots & t_n + 2 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- Where we are trying to find the rating r_i and t_i is the total number of games, $-n_{ij}$ is the number of games team i played against j and $b_i = 1 + \frac{1}{2}(w_i - l_i)$.
- The matrix C is a real symmetric matrix and $Cr = b$ is the system we need to solve.
- In the example of candidates: Hillary Clinton played forty games in total: ten games against Bernie Sanders, ten games against John Kasich, ten games against Ted Cruz, and ten games against Donald Trump. Each voter rated five candidates which is equivalent to four games since each candidate was indirectly being compared to the other four candidates.

	HC	BS	JK	TC	DT
HC	42	-10	-10	-10	-10
BS	-10	42	-10	-10	-10
Jk	-10	-10	42	-10	-10
TC	-10	-10	-10	42	-10
DT	-10	-10	-10	-10	42

Table: C Matrix.

Games	HC	BS	JK	TC	DT
Won	27	37	14	9	13
Lost	13	3	26	31	27
Total	40	40	40	40	40

Table: Candidate's Scoreboard.

$$\begin{bmatrix} 42 & -10 & -10 & -10 & -10 \\ -10 & 42 & -10 & -10 & -10 \\ -10 & -10 & 42 & -10 & -10 \\ -10 & -10 & -10 & 42 & -10 \\ -10 & -10 & -10 & -10 & 42 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix} = \begin{bmatrix} 8 \\ 18 \\ -5 \\ -10 \\ -6 \end{bmatrix}$$

After inputting all the matrices into MATLAB we use $\vec{r}$ = linsolve(C, $\vec{b}$) to get:
$\vec{r}$ =

$$\begin{bmatrix} 0.6346 \\ 0.8269 \\ 0.3846 \\ 0.2885 \\ 0.3654 \end{bmatrix}$$

Is the solution unique? Why?

The largest rating of 0.8269 corresponds to Bernie Sanders. Therefore, Bernie Sanders is the winner for Colley's method.

A classic application: Regression Analysis

- We have data points $x_1, x_2, \ldots, x_m$ each has n input variables x_{ij}, also called **features**.
  Example: measurements such as living-space area, number of rooms, etc.

A classic application: Regression Analysis

- We have data points $x_1, x_2, \ldots, x_m$ each has n input variables x_{ij}, also called **features**.
  Example: measurements such as living-space area, number of rooms, etc.
- We have y_i to denote the response, training or target variable that we are *trying to predict.*
  Example: We wish to predict the price of the house.

A classic application: Regression Analysis

- We have data points $x_1, x_2, \ldots, x_m$ each has n input variables x_{ij}, also called **features**.
  Example: measurements such as living-space area, number of rooms, etc.
- We have y_i to denote the response, training or target variable that we are *trying to predict.*
  Example: We wish to predict the price of the house.
- A pair (x_i, y_i) is called *a training example*. The dataset that we use to learn a list of m training examples. $(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)$ is called a *training set* We will also use X denote the space of input values, and Y the space of output values.

A classic application: Regression Analysis

- We have data points $x_1, x_2, \ldots, x_m$ each has n input variables x_{ij}, also called **features**.
  Example: measurements such as living-space area, number of rooms, etc.
- We have y_i to denote the response, training or target variable that we are *trying to predict.*
  Example: We wish to predict the price of the house.
- A pair (x_i, y_i) is called *a training example*. The dataset that we use to learn a list of m training examples. $(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)$ is called a *training set* We will also use X denote the space of input values, and Y the space of output values.
- **GOAL** Given a training set, to *learn* a function $h : X \to Y$ so that $h(x)$ is a "good" predictor for the corresponding value of y. This is the *supervised learning problem*

A classic application: Regression Analysis

- We have data points $x_1, x_2, \ldots, x_m$ each has n input variables x_{ij}, also called **features**.
  Example: measurements such as living-space area, number of rooms, etc.
- We have y_i to denote the response, training or target variable that we are *trying to predict.*
  Example: We wish to predict the price of the house.
- A pair (x_i, y_i) is called *a training example*. The dataset that we use to learn a list of m training examples. $(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)$ is called a *training set* We will also use X denote the space of input values, and Y the space of output values.
- **GOAL** Given a training set, to *learn* a function $h : X \to Y$ so that $h(x)$ is a "good" predictor for the corresponding value of y. This is the *supervised learning problem*
- When target variables are continuous, we call the supervised learning a **regression problem**.
- When y can take on discrete values (e.g., want to predict if a dwelling is a house or an apartment), we call it a **classification problem**.

A classic application: Regression Analysis

- We have data points $x_1, x_2, \ldots, x_m$ each has n input variables x_{ij}, also called **features**.
  Example: measurements such as living-space area, number of rooms, etc.
- We have y_i to denote the response, training or target variable that we are *trying to predict.*
  Example: We wish to predict the price of the house.
- A pair (x_i, y_i) is called *a training example*. The dataset that we use to learn a list of m training examples. $(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)$ is called a *training set* We will also use X denote the space of input values, and Y the space of output values.
- **GOAL** Given a training set, to *learn* a function $h : X \rightarrow Y$ so that $h(x)$ is a "good" predictor for the corresponding value of y. This is the *supervised learning problem*
- When target variables are continuous, we call the supervised learning a **regression problem**.
- When y can take on discrete values (e.g., want to predict if a dwelling is a house or an apartment), we call it a **classification problem**.
  To perform supervised learning, we must decide how to represent functions/hypotheses h in a computer!

Basic Regression analysis

- LINEAR CHOICE: Approximate y as a linear function of x:

$$h(x) = \sum_{k}^{n} w_i x_{ik} = w^T x_i$$

Basic Regression analysis

- LINEAR CHOICE: Approximate y as a linear function of x:

$$h(x) = \sum_{k}^{n} w_i x_{ik} = w^T x_i$$

- Formally we wish to find a function that measures, for each value of the w, how close the $h(x_i)$'s are to the corresponding y_i's.
  We define the cost function that minimizes the error.

$$\min \sum_{i}^{m} (h(x_i) - y_i)^2 = \min_{w \in \mathbf{R}^m} ||X^T w - \hat{y}||_2$$

- We use the notation X to denote the $m \times n$ matrix whose rows are the data points x_i. Let $\hat{y}$ the row vector with y_i's. This yields the traditional LEAST SQUARES PROBLEM:
- **Proposition** The optimal solution is the solution of a linear system of equations

$$X^T X w = X^T \hat{y}$$

Polynomial Regression

- We may add nonlinear relations on the features by considering a polynomial of degree M over n variables as the learning function:

$$h(x) = \sum_{a_1+a_2+\cdots+a_n \leq M} c_{i_1,i_2,\ldots,i_n} x_{i1}^{a_1} x_{i2}^{a_2} \cdots x_{in}^{a_n}$$

Polynomial Regression

- We may add nonlinear relations on the features by considering a polynomial of degree M over n variables as the learning function:

$$h(x) = \sum_{a_1+a_2+\cdots+a_n \leq M} c_{i_1,i_2,\ldots,i_n} x_{i1}^{a_1} x_{i2}^{a_2} \cdots x_{in}^{a_n}$$

- The monomials can be used to represent all such polynomials as a vector space basis (of which dimension??)

Polynomial Regression

- We may add nonlinear relations on the features by considering a polynomial of degree M over n variables as the learning function:

$$h(x) = \sum_{a_1+a_2+\cdots+a_n \leq M} c_{i_1,i_2,\ldots,i_n} x_{i1}^{a_1} x_{i2}^{a_2} \cdots x_{in}^{a_n}$$

- The monomials can be used to represent all such polynomials as a vector space basis (of which dimension??)
- ANSWER:

$$m = \binom{M+n}{M}$$

.

- Thus we have a least squares problem over a matrix $\Phi(X)$ with m columns and r rows (number of data points). This is a **polynomial learning model of degree M**

$$\min \sum_{i}^{m} (h(x_i) - y_i)^2 = \min_{w \in \mathbf{R}^m} ||\Phi(X)^T w - \hat{y}||_2.$$

Polynomial Regression

- We may add nonlinear relations on the features by considering a polynomial of degree M over n variables as the learning function:

$$h(x) = \sum_{a_1+a_2+\cdots+a_n \leq M} c_{i_1,i_2,\ldots,i_n} x_{i1}^{a_1} x_{i2}^{a_2} \cdots x_{in}^{a_n}$$

- The monomials can be used to represent all such polynomials as a vector space basis (of which dimension??)
- ANSWER:

$$m = \binom{M+n}{M}$$

.

- Thus we have a least squares problem over a matrix $\Phi(X)$ with m columns and r rows (number of data points). This is a **polynomial learning model of degree M**

$$\min \sum_{i}^{m} (h(x_i) - y_i)^2 = \min_{w \in \mathbf{R}^m} ||\Phi(X)^T w - \hat{y}||_2.$$

- It might seem that the more features we add, the better. However, there is a danger in adding too many features! We have to be careful with **underfitting** or **overfitting**.

Evaluating the regression result

- How to compare different regression models? How good is the prediction?

Evaluating the regression result

- How to compare different regression models? How good is the prediction?
- **Leave-one-out approach** Instead of using the entire data set to fit our model we use all but one point x_j. Missing point will be our test

Evaluating the regression result

- How to compare different regression models? How good is the prediction?
- **Leave-one-out approach** Instead of using the entire data set to fit our model we use all but one point x_j. Missing point will be our test
- Solve the smaller Least squares problem, we get a prediction for x_j, $\hat{y}(x_j)$. We can repeat the above process for all points (leave a different one out). Compute the *test set error*:

$$1/m \sum_{j=1}^{m} (\hat{y}(x_j) - y_j).$$

Evaluating the regression result

- How to compare different regression models? How good is the prediction?
- **Leave-one-out approach** Instead of using the entire data set to fit our model we use all but one point x_j. Missing point will be our test
- Solve the smaller Least squares problem, we get a prediction for x_j, $\hat{y}(x_j)$. We can repeat the above process for all points (leave a different one out). Compute the *test set error*:

$$1/m \sum_{j=1}^{m} (\hat{y}(x_j) - y_j).$$

- As the degree M of the polynomial grows the training error decreases, while the test set error typically decreases and then then increases. This is *overfitting*.

Evaluating the regression result

- How to compare different regression models? How good is the prediction?
- **Leave-one-out approach** Instead of using the entire data set to fit our model we use all but one point x_j. Missing point will be our test
- Solve the smaller Least squares problem, we get a prediction for x_j, $\hat{y}(x_j)$. We can repeat the above process for all points (leave a different one out). Compute the *test set error*:

$$1/m \sum_{j=1}^{m} (\hat{y}(x_j) - y_j).$$

- As the degree M of the polynomial grows the training error decreases, while the test set error typically decreases and then then increases. This is *overfitting*.
- Models of higher M complexity fit the training data better, but a model that is too complex (high M) does not behave well on unseen data. We are looking for a compromise.

Evaluating the regression result

- How to compare different regression models? How good is the prediction?
- **Leave-one-out approach** Instead of using the entire data set to fit our model we use all but one point x_j. Missing point will be our test
- Solve the smaller Least squares problem, we get a prediction for x_j, $\hat{y}(x_j)$. We can repeat the above process for all points (leave a different one out). Compute the *test set error*:

$$1/m \sum_{j=1}^{m} (\hat{y}(x_j) - y_j).$$

- As the degree M of the polynomial grows the training error decreases, while the test set error typically decreases and then then increases. This is *overfitting*.
- Models of higher M complexity fit the training data better, but a model that is too complex (high M) does not behave well on unseen data. We are looking for a compromise.
- **leave-more out**

Introduction to Eigenvalues

- **Eigenvalues** appear in many problems of applied mathematics: Google search, differential equations, control theory etc.

Introduction to Eigenvalues

- **Eigenvalues** appear in many problems of applied mathematics: Google search, differential equations, control theory etc.
- **Definition** A non-zero vector x is an **Eigenvector** of A if $Ax = \lambda x$ and then we say λ is an **eigenvalue**.

Introduction to Eigenvalues

- **Eigenvalues** appear in many problems of applied mathematics: Google search, differential equations, control theory etc.
- **Definition** A non-zero vector x is an **Eigenvector** of A if $Ax = \lambda x$ and then we say λ is an **eigenvalue**.
- Clearly the eigenvectors of A make up the nullspace of $A - \lambda I$. So if we know the eigenvalues then we can find eigenvectors!! So how to find the eigenvalues???

Introduction to Eigenvalues

- **Eigenvalues** appear in many problems of applied mathematics: Google search, differential equations, control theory etc.
- **Definition** A non-zero vector x is an **Eigenvector** of A if $Ax = \lambda x$ and then we say λ is an **eigenvalue**.
- Clearly the eigenvectors of A make up the nullspace of $A - \lambda I$. So if we know the eigenvalues then we can find eigenvectors!! So how to find the eigenvalues???
- **Theorem:** A number λ is an eigenvalue of A if and only if $A - \lambda I$ is singular, namely, $det(A - \lambda I) = 0$.

Introduction to Eigenvalues

- **Eigenvalues** appear in many problems of applied mathematics: Google search, differential equations, control theory etc.
- **Definition** A non-zero vector x is an **Eigenvector** of A if $Ax = \lambda x$ and then we say λ is an **eigenvalue**.
- Clearly the eigenvectors of A make up the nullspace of $A - \lambda I$. So if we know the eigenvalues then we can find eigenvectors!! So how to find the eigenvalues???
- **Theorem:** A number λ is an eigenvalue of A if and only if $A - \lambda I$ is singular, namely, $det(A - \lambda I) = 0$.
- **Corollary** The eigenvalues of an $n \times n$ matrix A are the roots of a polynomial of degree n. This is HARD TO DO!!!!

Introduction to Eigenvalues

- **Eigenvalues** appear in many problems of applied mathematics: Google search, differential equations, control theory etc.
- **Definition** A non-zero vector x is an **Eigenvector** of A if $Ax = \lambda x$ and then we say λ is an **eigenvalue**.
- Clearly the eigenvectors of A make up the nullspace of $A - \lambda I$. So if we know the eigenvalues then we can find eigenvectors!! So how to find the eigenvalues???
- **Theorem:** A number λ is an eigenvalue of A if and only if $A - \lambda I$ is singular, namely, $det(A - \lambda I) = 0$.
- **Corollary** The eigenvalues of an $n \times n$ matrix A are the roots of a polynomial of degree n. This is HARD TO DO!!!!
- **Theorem** An $n \times n$ matrix A has n eigenvalues (with multiplicity).

Two key results

Theorem

Let λ_i *for* $i = 1 \dots k$ *the distinct eigenvalues of a matrix* A*. Let* Φ_i *be the* **eigenspace** *of* λ_i*. Then a set of eigenvectors, one for each eigenspace is a set of linearly independent vectors.*

Theorem

Let A *have* n *distinct eigenvalues* λ_i *for* $i = 1 \dots k$*. Let* μ_i *the* **algebraic multiplicity** *of* λ_i*. Moreover, let* Φ_i *the eigenspace of* λ_i *and* $v_i = dim(\Phi_i)$*. Let* $U^{(i)}$ *be the matrix containing as columns a basis of* Φ_i *and* $U = [U^{(1)} U^{(2)} \cdots U^{(k)}]$*. Then*

- $v_i \leq \mu_i$.
- *if* $v_i = \mu_i$ *then* A *is* **diagonalizable***:*

$$A = U \Lambda U^{-1}$$

MARKOV PROCESSES

- Suppose a Taxi company serves Davis, Sacramento, and Woodland. Records of the company indicate:
  - 10% of the costumers taking a taxi in Davis go to Sacramento and 30% to Woodland.

MARKOV PROCESSES

- Suppose a Taxi company serves Davis, Sacramento, and Woodland. Records of the company indicate:
  - 10% of the costumers taking a taxi in Davis go to Sacramento and 30% to Woodland.
  - 30% of costumers taking a taxi in Sacramento go to Davis and 30% go to Woodland.

MARKOV PROCESSES

- Suppose a Taxi company serves Davis, Sacramento, and Woodland. Records of the company indicate:
  - 10% of the costumers taking a taxi in Davis go to Sacramento and 30% to Woodland.
  - 30% of costumers taking a taxi in Sacramento go to Davis and 30% go to Woodland.
  - 40% of costumers in Woodland go to Davis and to Sacramento 30%
- The company wants to know where the taxis will end up on average. We can write this as a **Markov Process**.

MARKOV PROCESSES

- Suppose a Taxi company serves Davis, Sacramento, and Woodland. Records of the company indicate:
  - 10% of the costumers taking a taxi in Davis go to Sacramento and 30% to Woodland.
  - 30% of costumers taking a taxi in Sacramento go to Davis and 30% go to Woodland.
  - 40% of costumers in Woodland go to Davis and to Sacramento 30%
- The company wants to know where the taxis will end up on average. We can write this as a **Markov Process**.
- The entries of a **State vector** $u^{(k)} = (u_1^{(k)}, u_2^{(k)}, u_3^{(k)})^T$ tell what proportion of taxis is in Davis, Sacramento, Woodland.

MARKOV PROCESSES

- Suppose a Taxi company serves Davis, Sacramento, and Woodland. Records of the company indicate:
  - 10% of the costumers taking a taxi in Davis go to Sacramento and 30% to Woodland.
  - 30% of costumers taking a taxi in Sacramento go to Davis and 30% go to Woodland.
  - 40% of costumers in Woodland go to Davis and to Sacramento 30%
- The company wants to know where the taxis will end up on average. We can write this as a **Markov Process**.
- The entries of a **State vector** $u^{(k)} = (u_1^{(k)}, u_2^{(k)}, u_3^{(k)})^T$ tell what proportion of taxis is in Davis, Sacramento, Woodland.
- The next stage of the evolution of the process will be given by a **Transition matrix** T.

MARKOV PROCESSES

- Suppose a Taxi company serves Davis, Sacramento, and Woodland. Records of the company indicate:
  - 10% of the costumers taking a taxi in Davis go to Sacramento and 30% to Woodland.
  - 30% of costumers taking a taxi in Sacramento go to Davis and 30% go to Woodland.
  - 40% of costumers in Woodland go to Davis and to Sacramento 30%
- The company wants to know where the taxis will end up on average. We can write this as a **Markov Process**.
- The entries of a **State vector** $u^{(k)} = (u_1^{(k)}, u_2^{(k)}, u_3^{(k)})^T$ tell what proportion of taxis is in Davis, Sacramento, Woodland.
- The next stage of the evolution of the process will be given by a **Transition matrix** T.
- Then $u^{(k+1)} = Tu^{(k)}$, with $u^{(0)}$ being the initial state at time zero (MARKOV CHAIN)

- We interpret the entry $u_i^{(k)}$ as the probability that the system is in state i (e.g., in the city of Sacramento)

- We interpret the entry $u_i^{(k)}$ as the probability that the system is in state i (e.g., in the city of Sacramento)
- The initial vector has the proportions of taxis at the beginning.

- We interpret the entry $u_i^{(k)}$ as the probability that the system is in state i (e.g., in the city of Sacramento)
- The initial vector has the proportions of taxis at the beginning.
- The entries t_{ij} of the transition matrix are represent the **transitional probabilities** that the system will switch from state j to state i (note the reversal of indices). For example:

$$\begin{bmatrix} 0.6 & 0.3 & 0.4 \\ 0.1 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix}$$

- We interpret the entry $u_i^{(k)}$ as the probability that the system is in state i (e.g., in the city of Sacramento)
- The initial vector has the proportions of taxis at the beginning.
- The entries t_{ij} of the transition matrix are represent the **transitional probabilities** that the system will switch from state j to state i (note the reversal of indices). For example:

$$\begin{bmatrix} 0.6 & 0.3 & 0.4 \\ 0.1 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix}$$

- Note: $0 \leq t_{ij} \leq 1$ and $t_{1j} + t_{2j} + \cdots + t_{jn} = 1$.

- We interpret the entry $u_i^{(k)}$ as the probability that the system is in state i (e.g., in the city of Sacramento)
- The initial vector has the proportions of taxis at the beginning.
- The entries t_{ij} of the transition matrix are represent the **transitional probabilities** that the system will switch from state j to state i (note the reversal of indices). For example:

$$\begin{bmatrix} 0.6 & 0.3 & 0.4 \\ 0.1 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix}$$

- Note: $0 \leq t_{ij} \leq 1$ and $t_{1j} + t_{2j} + \cdots + t_{jn} = 1$.
- The equation $u^{(k+1)} = Tu^{(k)}$ represents a chain of probability vectors.

- We interpret the entry $u_i^{(k)}$ as the probability that the system is in state i (e.g., in the city of Sacramento)
- The initial vector has the proportions of taxis at the beginning.
- The entries t_{ij} of the transition matrix are represent the **transitional probabilities** that the system will switch from state j to state i (note the reversal of indices). For example:

$$\begin{bmatrix} 0.6 & 0.3 & 0.4 \\ 0.1 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix}$$

- Note: $0 \leq t_{ij} \leq 1$ and $t_{1j} + t_{2j} + \cdots + t_{jn} = 1$.
- The equation $u^{(k+1)} = Tu^{(k)}$ represents a chain of probability vectors.
- T having non-zero entries means there is a positive probability to from A to B in the system (matrix is regular!)

- We interpret the entry $u_i^{(k)}$ as the probability that the system is in state i (e.g., in the city of Sacramento)
- The initial vector has the proportions of taxis at the beginning.
- The entries t_{ij} of the transition matrix are represent the **transitional probabilities** that the system will switch from state j to state i (note the reversal of indices). For example:

$$\begin{bmatrix} 0.6 & 0.3 & 0.4 \\ 0.1 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix}$$

- Note: $0 \leq t_{ij} \leq 1$ and $t_{1j} + t_{2j} + \cdots + t_{jn} = 1$.
- The equation $u^{(k+1)} = Tu^{(k)}$ represents a chain of probability vectors.
- T having non-zero entries means there is a positive probability to from A to B in the system (matrix is regular!)
- **Theorem (Perron-Frobenius)** If T is a regular transition matrix, then it admits a unique probability eigenvector u^* with eigenvalue $\lambda = 1$. Moreover $Tu^{(k)} \longrightarrow u^*$ as k increases.

- For our example, the eigenvector is $(.4714, .2286, .3)^T$.

- For our example, the eigenvector is $(.4714, .2286, .3)^T$.
- So eventually, no matter how the taxis were initially distributed, 47$\%$ of taxis will be in Davis, 23$\%$ in Sacramento and 30$\%$ in Woodland.

- For our example, the eigenvector is $(.4714, .2286, .3)^T$.
- So eventually, no matter how the taxis were initially distributed, 47% of taxis will be in Davis, 23% in Sacramento and 30% in Woodland.
- How fast do we converge to this stable situation? Given by the second largest eigenvalue $|\lambda_2|$, the smaller it is the faster the process goes. In our example $\lambda_2 = .3$ and $\lambda_3 = 0$. So the convergence to steady state is fairly rapid.

- For our example, the eigenvector is $(.4714, .2286, .3)^T$.
- So eventually, no matter how the taxis were initially distributed, 47% of taxis will be in Davis, 23% in Sacramento and 30% in Woodland.
- How fast do we converge to this stable situation? Given by the second largest eigenvalue $|\lambda_2|$, the smaller it is the faster the process goes. In our example $\lambda_2 = .3$ and $\lambda_3 = 0$. So the convergence to steady state is fairly rapid.
- **Lemma** All eigenvalues of a Markov matrix satisfy $|\lambda_i| \leq 1$.

- For our example, the eigenvector is $(.4714, .2286, .3)^T$.
- So eventually, no matter how the taxis were initially distributed, 47$\%$ of taxis will be in Davis, 23$\%$ in Sacramento and 30$\%$ in Woodland.
- How fast do we converge to this stable situation? Given by the second largest eigenvalue $|\lambda_2|$, the smaller it is the faster the process goes. In our example $\lambda_2 = .3$ and $\lambda_3 = 0$. So the convergence to steady state is fairly rapid.
- **Lemma** All eigenvalues of a Markov matrix satisfy $|\lambda_i| \leq 1$.
- So eigenvalues can be used to predict dynamic behavior.

PageRank Algorithm

- How do we rank web sites? Pagerank is the algorithm used by Google, it was invented by Larry Page (but based on the Mathematical work on many before him!!).

PageRank Algorithm

- How do we rank web sites? Pagerank is the algorithm used by Google, it was invented by Larry Page (but based on the Mathematical work on many before him!!).
- One can start with a simple, but bad, idea: Assign to a web page the score or ranking equal to the number of web pages that link to it.

PageRank Algorithm

- How do we rank web sites? Pagerank is the algorithm used by Google, it was invented by Larry Page (but based on the Mathematical work on many before him!!).
- One can start with a simple, but bad, idea: Assign to a web page the score or ranking equal to the number of web pages that link to it.
- It is not good since it does not take into account WHO is linking into the web page, e.g., not the same Yahoo.com has a link to your webpage, than nobody.com links to it. Importance should be rewarded. Also, important players do not have to link to low-rank places, they have few outgoing links!

PageRank Algorithm

- How do we rank web sites? Pagerank is the algorithm used by Google, it was invented by Larry Page (but based on the Mathematical work on many before him!!).
- One can start with a simple, but bad, idea: Assign to a web page the score or ranking equal to the number of web pages that link to it.
- It is not good since it does not take into account WHO is linking into the web page, e.g., not the same Yahoo.com has a link to your webpage, than nobody.com links to it. Importance should be rewarded. Also, important players do not have to link to low-rank places, they have few outgoing links!
- A better idea: Each page has a score x_j that indicates its importance, but if that webpage has n_j outgoing arcs (or links) then its vote to each is only $\frac{x_j}{n_j}$.

PageRank Algorithm

- How do we rank web sites? Pagerank is the algorithm used by Google, it was invented by Larry Page (but based on the Mathematical work on many before him!!).
- One can start with a simple, but bad, idea: Assign to a web page the score or ranking equal to the number of web pages that link to it.
- It is not good since it does not take into account WHO is linking into the web page, e.g., not the same Yahoo.com has a link to your webpage, than nobody.com links to it. Importance should be rewarded. Also, important players do not have to link to low-rank places, they have few outgoing links!
- A better idea: Each page has a score x_j that indicates its importance, but if that webpage has n_j outgoing arcs (or links) then its vote to each is only $\frac{x_j}{n_j}$.
- Final reputation of the k-th webpage is given by

$$x_k = \sum_{j \in \text{with arc going to k}} \frac{x_j}{n_j}$$

PageRank Algorithm

- How do we rank web sites? Pagerank is the algorithm used by Google, it was invented by Larry Page (but based on the Mathematical work on many before him!!).
- One can start with a simple, but bad, idea: Assign to a web page the score or ranking equal to the number of web pages that link to it.
- It is not good since it does not take into account WHO is linking into the web page, e.g., not the same Yahoo.com has a link to your webpage, than nobody.com links to it. Importance should be rewarded. Also, important players do not have to link to low-rank places, they have few outgoing links!
- A better idea: Each page has a score x_j that indicates its importance, but if that webpage has n_j outgoing arcs (or links) then its vote to each is only $\frac{x_j}{n_j}$.
- Final reputation of the k-th webpage is given by

$$x_k = \sum_{j \in \text{with arc going to k}} \frac{x_j}{n_j}$$

- **Lemma** Resulting matrix equation $Ax = x$ is an eigenvalue problem! Matrix A is column stochastic matrix Thus Perron-Frobenius says there must be an eigenvector. Vector will be unique (up to scaling) if the problem is *irreducible*.

PageRank Algorithm

- How do we rank web sites? Pagerank is the algorithm used by Google, it was invented by Larry Page (but based on the Mathematical work on many before him!!).
- One can start with a simple, but bad, idea: Assign to a web page the score or ranking equal to the number of web pages that link to it.
- It is not good since it does not take into account WHO is linking into the web page, e.g., not the same Yahoo.com has a link to your webpage, than nobody.com links to it. Importance should be rewarded. Also, important players do not have to link to low-rank places, they have few outgoing links!
- A better idea: Each page has a score x_j that indicates its importance, but if that webpage has n_j outgoing arcs (or links) then its vote to each is only $\frac{x_j}{n_j}$.
- Final reputation of the k-th webpage is given by

$$x_k = \sum_{j \in \text{with arc going to k}} \frac{x_j}{n_j}$$

- **Lemma** Resulting matrix equation $Ax = x$ is an eigenvalue problem! Matrix A is column stochastic matrix Thus Perron-Frobenius says there must be an eigenvector. Vector will be unique (up to scaling) if the problem is *irreducible*.
- Intuitively irreducible means you cannot get stuck while walking the network.

What if A is not irreducible? Modify the matrix to be

$$A = (1 - \alpha)A + \alpha E, \quad ,\text{where } E = [1/n]$$

.

What if A is not irreducible? Modify the matrix to be

$$A = (1-\alpha)A + \alpha E, \quad \text{,where } E = [1/n]$$

.

Page Rank Eigenvalue problem

Determine the final ratings by solving for the dominant eigenvector $\vec{r}$ (associated with eigenvalue 1) in the following irreducible matrix,

$$\tilde{A} = \alpha \cdot A + \frac{(1-\alpha)}{n} \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \text{ for } 0 < \alpha < 1.$$

What if A is not irreducible? Modify the matrix to be

$$A = (1-\alpha)A + \alpha E, \quad ,\text{where } E = [1/n]$$

.

Page Rank Eigenvalue problem

Determine the final ratings by solving for the dominant eigenvector $\vec{r}$ (associated with eigenvalue 1) in the following irreducible matrix,

$$\tilde{A} = \alpha \cdot A + \frac{(1-\alpha)}{n} \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \text{ for } 0 < \alpha < 1.$$

NOTE: For games of basketball or voting A is a stochastic matrix with entries composed of 0's and $\frac{1}{j_i}$'s where j_i represents that number of loses candidate i obtained (pointing an arrow to someone means you lost to him/her).
If candidate i is undefeated replace the $\mathbf{0}^T$ with $\begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$ where n is the number of candidates.

What if A is not irreducible? Modify the matrix to be

$$A = (1-\alpha)A + \alpha E, \quad \text{,where } E = [1/n]$$

.

Page Rank Eigenvalue problem

Determine the final ratings by solving for the dominant eigenvector $\vec{r}$ (associated with eigenvalue 1) in the following irreducible matrix,

$$\tilde{A} = \alpha \cdot A + \frac{(1-\alpha)}{n} \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \text{ for } 0 < \alpha < 1.$$

NOTE: For games of basketball or voting A is a stochastic matrix with entries composed of 0's and $\frac{1}{j_i}$'s where j_i represents that number of loses candidate i obtained (pointing an arrow to someone means you lost to him/her).
If candidate i is undefeated replace the $\mathbf{0}^T$ with $\begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$ where n is the number of candidates.
How do you solve such a large-scale problem? Use the **power method**!!

What if A is not irreducible? Modify the matrix to be

$$A = (1-\alpha)A + \alpha E, \quad ,\text{where } E = [1/n]$$

.

Page Rank Eigenvalue problem

Determine the final ratings by solving for the dominant eigenvector $\vec{r}$ (associated with eigenvalue 1) in the following irreducible matrix,

$$\tilde{A} = \alpha \cdot A + \frac{(1-\alpha)}{n} \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \text{ for } 0 < \alpha < 1.$$

NOTE: For games of basketball or voting A is a stochastic matrix with entries composed of 0's and $\frac{1}{j_i}$'s where j_i represents that number of loses candidate i obtained (pointing an arrow to someone means you lost to him/her).
If candidate i is undefeated replace the $\mathbf{0}^T$ with $\begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$ where n is the number of candidates.
How do you solve such a large-scale problem? Use the **power method**!!
$A^k(r) \to t$ when $k \to \infty$ converges to the 1 eigenvalue we need. WHY??

- Given an $m \times n$ matrix A, its **singular values** are the positive square roots of the eigenvalues of the matrix $A^T A$ (the **Gram matrix**). The corresponding eigenvectors are called the singular vectors of A.
- By Cholesky's decomposition $A^T A$ is positive semidefinite its eigenvalues are always non-negative and real.
- **Example** Say A is a the square matrix $\begin{bmatrix} 3 & 5 \\ 4 & 0 \end{bmatrix}$ Thus $A^T A = \begin{bmatrix} 25 & 15 \\ 15 & 25 \end{bmatrix}$ This matrix has eigenvalues 40 and 10. and the corresponding eigenvectors are $(1,1)^T$ and $(1,-1)^T$. Thus the singular values are $\sigma_1 = \sqrt{40} = 6.3246$ and $\sigma_2 = \sqrt{10} = 3.1623$. Note singular values are NOT the same as the eigenvalues of A.

- There are a number of properties we will need to understand the importance of the singular values
- **Theorem** Every real symmetric matrix A can be diagonalized by an orthogonal matrix Q, i.e., $A = Q\Lambda Q^T$ where Q is orthogonal and Λ is a diagonal matrix.
- **Proposition** If $A^T = A$ then its singular values are the absolute values of its non-zero eigenvalues and its singular vectors coincide with the associated non-null eigenvectors.

- **Theorem** Any non-zero real $(m \times n)$ matrix A of rank $r > 0$ can be factored

$$A = P \Sigma Q^T$$

  - P is an $m \times r$ matrix with orthonormal columns ($P^T P = I$).
  - Σ is an $r \times r$ diagonal matrix with the singular values of A in the diagonal.
  - Q^T is an $r \times n$ matrix with orthonormal rows ($Q^T Q = I$).
- **Proof** Same as proving $AQ = P\Sigma$. So find orthonormal vectors $q_1, \ldots, q_r$ for Q and $p_1, \ldots, p_r$!!!
- Let $q_1, \ldots, q_r$ be the orthonormal eigenvectors of Gram matrix $A^T A$ corresponding to the **non-zero eigenvalues**.

$$A^T A q_i = \lambda_i q_i = \sigma_i^2 q_i$$

- **Claim 1:** $w_i = Aq_i$ are orthogonal.
- **Claim 2:** $\|w_i\| = \sqrt{w_i^T w_i} = \sigma$
- **Claim 3:** $p_i = \frac{w_i}{\sigma_i} = \frac{Aq_i}{\sigma}$.

- **Example (continued)** For $A = \begin{bmatrix} 3 & 5 \\ 4 & 0 \end{bmatrix}$, the orthogonal eigenvector basis of A^TA is $q_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^T, q_2 = (\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^T$

- Thus, following the method of the algorithm, $Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$ and $P = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{bmatrix}$

- Thus $A = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{40} & 0 \\ 0 & \sqrt{10} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$

- **Theorem** Any non-zero real $m \times n$ A can be factored

$$A = P\Sigma Q^T$$

  where
  - P is an $m \times m$ matrix with orthonormal columns ($P^T P = I$).
  - Σ is an $m \times n$ diagonal matrix with the singular values of A in the diagonal.
  - Q is an $n \times n$ matrix with orthonormal rows ($Q^T Q = I$).
- WHAT IS THE GEOMETRIC MEANING?
- Let us look at what happens to a unit sphere?

Theory Consequences of SVD decomposition:

- **Theorem** Let $A = P\Sigma Q^T$ be the SVD of A.
  Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n$ be the singular values (first r non-zero!) Then
  - The (left) singular vectors $p_1, p_2, \ldots, p_r$ for an orthonormal basis in the range of A and $rank(A) = r$
  - The (right) singular vectors $q_{r+1}, q_{r+2}, \ldots, q_n$ are an orthonormal basis in $N(A)$ and $dim(N(A)) = n - r$.
  - The (right) singular vectors $q_1, q_2, \ldots, q_r$ are an orthonormal basis in the range of A^T.
  - The (left) singular vectors $p_{r+1}, p_{r+2}, \ldots, p_m$ are an orthonormal basis for $N(A^T)$.
  - Moreover, he columns of P are the eigenvectors of AA^T and the columns of Q are the eigenvectors of $A^T A$.
  - The r singular values on the diagonal of σ are the square roots of the non-zero eigenvalues of both $A^T A$ and AA^T.
- **Proposition:** The largest singular value is equal to the 2-norm of A.
- The Frobenius norm of a matrix A, (denoted $\|A\|_F$ is $\sqrt{\sum_{i,j} a_{i,j}^2}$ or $\sqrt{Tr(A^T A)}$. We have:
- The ratio $\sigma_{\mathbf{max}} / \sigma_{\mathbf{min}}$ is the **Condition number**.

WHY CARE? Applications: Numerics, Image Compression, Data Analysis

- The ratio $\sigma_{\mathbf{max}}/\sigma_{\mathbf{min}}$ is the **Condition number**. A matrix with very large condition number is **ill-conditioned**. This is trouble for calculations when the condition number is larger than the reciprocal of the machine precision.
- The singular vectors indicate the **principal components**. One key idea is to discard singular values and vectors that are too small
- The columns of the matrix A represent data vectors (normalized to have mean 0). The matrix A^TA can be thought of as the **covariance matrix**. Its eigenvectors indicate directions of correlation and clustering in the data.
- For Images, instead of sending a 1000×1000 pixels, compute SVD, $A = P\Sigma Q^T = p_1\sigma_1 q_1^T + p_2\sigma_2 q_2^T + \cdots + p_r\sigma_r q_r^T$, we see A is a sum of rank 1 matrices!
  we only keep those pieces with σ_i of "good" size, throw away rest.

Low Rank Approximations

- Given two vectors u, v, What kind of matrix is (uv^T)?
- It is a **rank one** matrix!

$$\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix} = \begin{pmatrix} u_1 v_1 & \dots & u_1 v_n \\ \vdots & & \vdots \\ u_n v_1 & \dots & u_n v_n \end{pmatrix}$$

- **Corollary:** A matrix A of rank r can be written as a sum of r, rank one matrices using the SVD decomposition: $A = \sum_{i=1}^{n} \sigma_i (u_i v_i^T)$.
- **Theorem:** For the $m \times n$ matrix A call the truncated sum of matrices $A_k = \sum_{i=1}^{k} \sigma_i (u_i v_i^T)$. Then the optimization problem:

$$\text{Minimize}_{B\ m\times n, rank(B) \leq k} \|A - B\|_2 = \sigma_{k+1}$$

It is achieved by $B = A_k$. Same holds for the Frobenius norm.

Applications: Image Compression

- **key idea for image compression:** discard singular values and vectors that are too small
- For images, instead of sending a 1000×1000 pixels over the internet.
- compute SVD,

$$A = P\Sigma Q^T = p_1\sigma_1 q_1^T + p_2\sigma_2 q_2^T + \cdots + p_r\sigma_r q_r^T$$

We see A is a sum of rank 1 matrices, keep those pieces with σ_i of "good" size, throw away rest.

Principal Component Analysis of Data

- **Goal:** Discover the most important *directions* where data varies the most!!
- **Answer:** The (left) singular vectors are principal components. Why?
  NOTE: If not mean zero, simply translate to the average $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$. Assume data is centered!!
- **WISH** Find z with $||z||_2 = 1$ such that the variance of the projection of the data points onto the line defined by z
- Components of the data along z are given by $a_i = x_i^T z$.
- The **mean square variation of data** equals:

$$\frac{1}{m}\sum_{i=1}^{m} a_i^2 = \sum_{i=1} mz^T x_i x^T z = z^T AA^T z$$

- Desired direction is z that maximizes:

$$\max_{z \in R^n} z^T AA^T z, \quad ||z||_2 = 1$$

  But by SVD equals

$$\max_{z \in R^n} z^T P_r \Sigma^2 P_r^T z, \quad ||z||_2 = 1$$

- Optimal value $z = p_1$ first column of P and largest variation is σ_1^2