

Mathematics of Big Data, I

Lecture 10: Artificial Neural Network and Graph Techniques in Big Data Analysis

Weiqing Gu

Professor of Mathematics

Director of the Mathematics Clinic

Harvey Mudd College

Summer 2017

Today's Topics

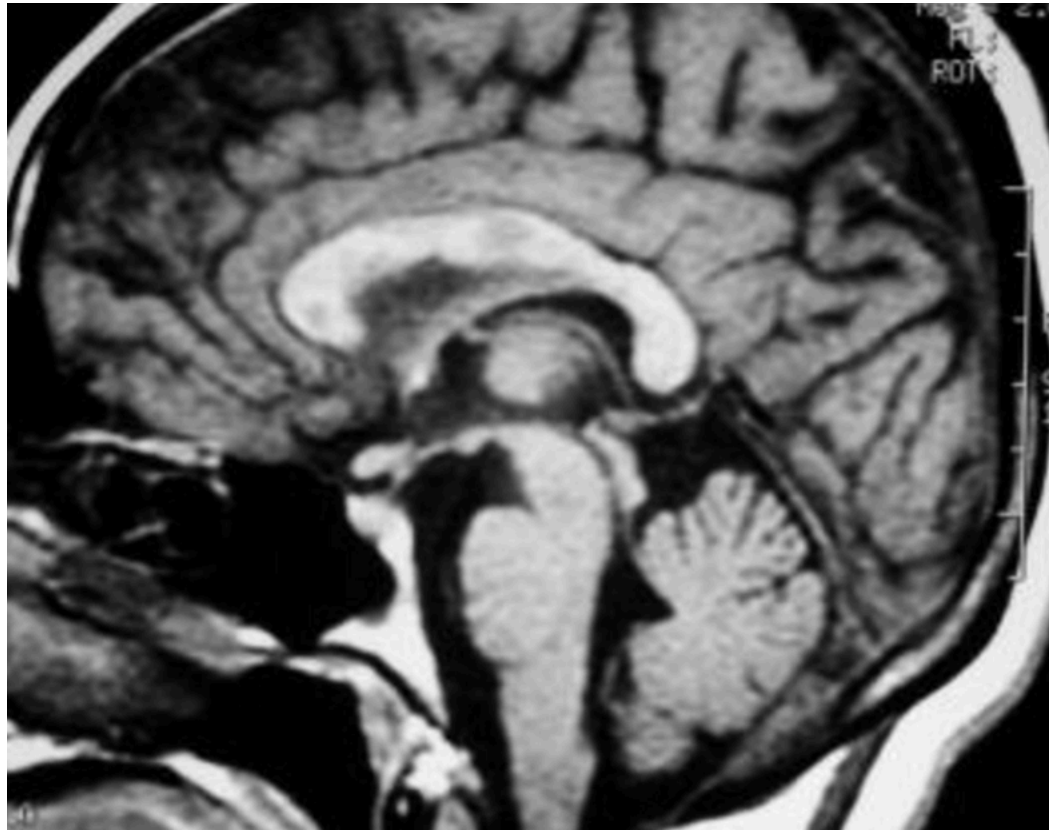
- Artificial Neural Network
- Graphical Modeling in Machine Learning
 - Page Rank
 - Graph data representations
 - Markov Chain
 - Laplacian Matrix
 - Usage of Spectral (Eigenvalue-Eigenvector) information.
- Directed Graphical Models (Bayesian Networks)
- Plate Notation (*only if time permits*)
 - Naive Bayes as a Graphical Model
 - LDA

Today's Topics

- **Artificial Neural Network**
- Graphical Modeling in Machine Learning
 - Page Rank
 - Graph data representations
 - Markov Chain
 - Laplacian Matrix
 - Usage of Spectral (Eigenvalue-Eigenvector) information.
- Directed Graphical Models (Bayesian Networks)
- Plate Notation (*only if time permits*)
 - Naive Bayes as a Graphical Model
 - LDA

An Artificial Neural Network tries to mimic some basic functions of a brain.

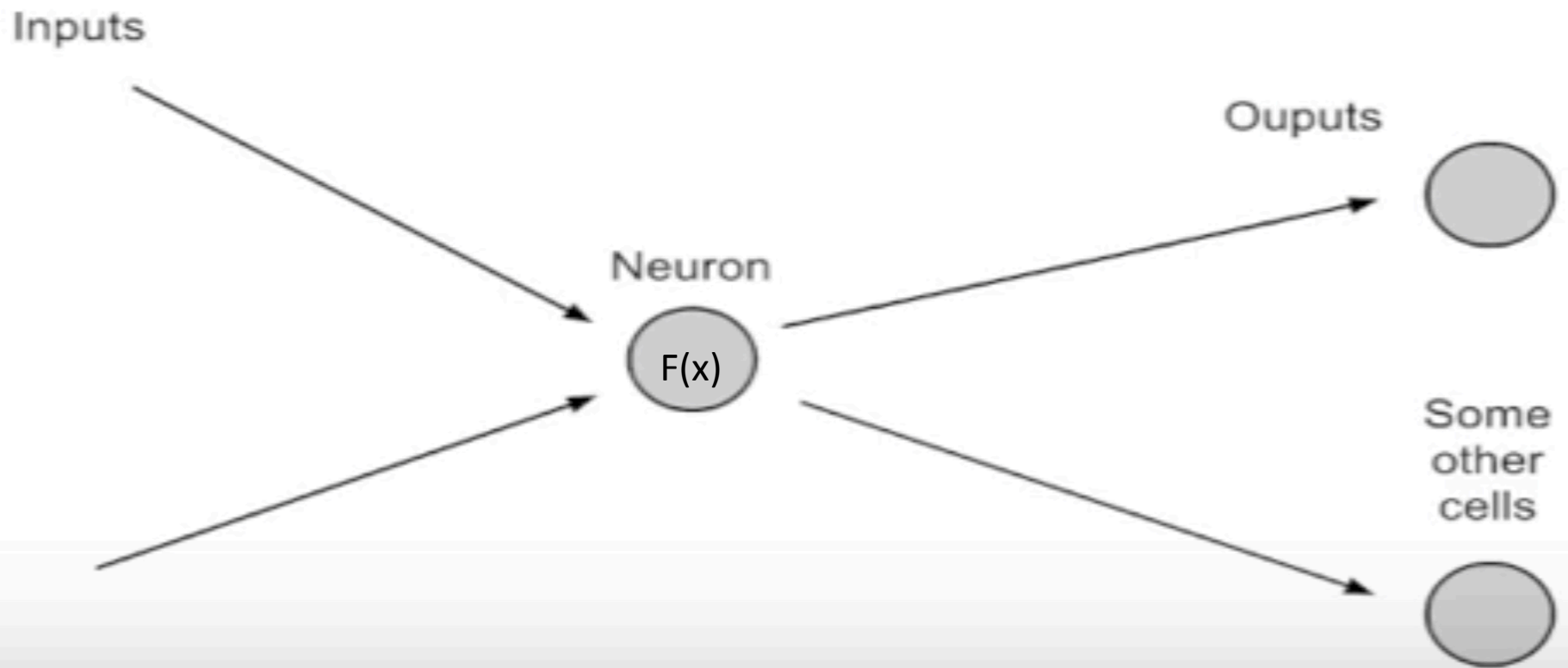
A human brain has 100 billion cells, called neurons.

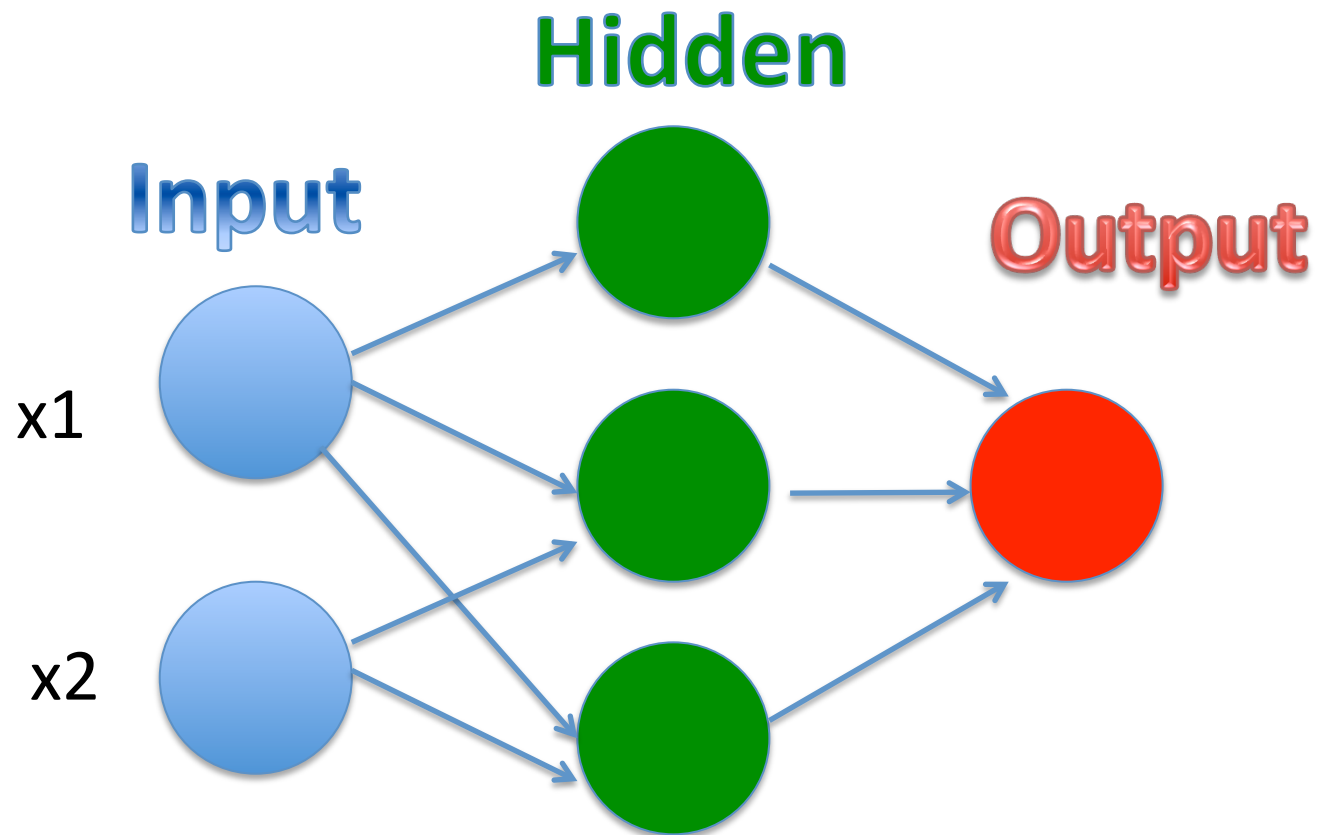


Neurons are connected each other through pathways that transmit electrical signals. These connections give neurons the ability to send and receive electrical impulse which in turn are responsible for the brain function on large scale.



Key: We can represent each neuron as a math function. It has inputs and outputs. When it receives electrical impulses from a cell it sends it to other cells it's connected to. So a neural network is just an attempt to make the computers model the brain of reasoning. If computers were more like the brain they could be good at some of things human are good at such as pattern recognition. A neural network simulated a collections of neurons just as they do in the brain. These simulated neurons take input and gives outputs through their connections.





Let's see an **Example**

- Work out mathematics details with students on the board and see the code in the following slides.

There are lots of machine algorithms for making prediction

- Regression methods
- Support Vector Machine
-
- Artificial Neural Network **ANN**

Code an ANN example

```
In [1]: X = np.array([[3,5], [5,1], [10,2]], dtype=float)
        y = np.array([75], [82], [93]), dtype=float)
```

```
In [2]: X
```

```
Out[2]: array([[ 3.,  5.],
               [ 5.,  1.],
               [10.,  2.]])
```

```
In [3]: y
```

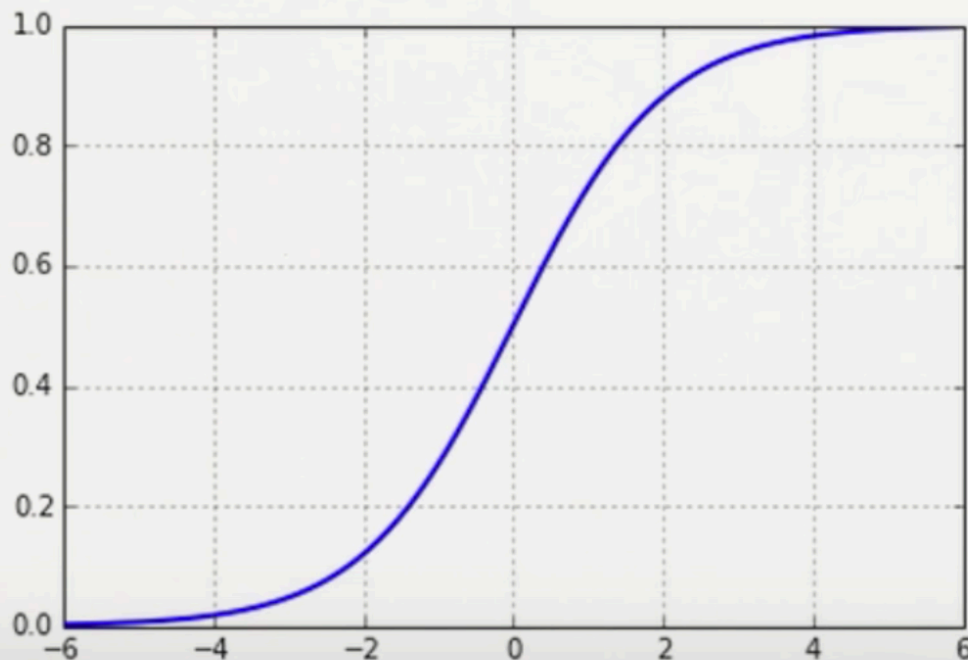
```
Out[3]: array([[ 75.],
               [ 82.],
               [ 93.]])
```

Hyper Parameters determines the structure of an ANN!

```
In [1]: class Neural_Network(object):  
        def __init__(self):  
            #Define HyperParameters  
            self.inputLayerSize = 2  
            self.outputLayerSize = 1  
            self.hiddenLayerSize = 3  
  
        def forward(self, X):  
            #Propagate inputs through network  
            |
```

ANN uses a smooth function such as sigmoid function to make the differentiation possible.

```
In [2]: testInput = np.arange(-6,6,0.01)  
plot(testInput, sigmoid(testInput), linewidth=2)  
grid(1)
```



When a Sigmoid function applies to a matrix in ANN, it applies to each entry of the matrix.

```
In [3]: sigmoid(1)
```

```
Out[3]: 0.7310585786300049
```

```
In [4]: sigmoid(np.array([-1,0,1]))
```

```
Out[4]: array([ 0.26894142,  0.5         ,  0.73105858])
```

```
In [5]: sigmoid(np.random.randn(3,3))
```

```
Out[5]: array([[ 0.71205286,  0.53868493,  0.6115578 ],
                [ 0.34019709,  0.5402725 ,  0.62380767],
                [ 0.11294651,  0.33710901,  0.67477125]])
```

```
In [ ]: |
```

Forward Propagate inputs through network

```
[1]: class Neural_Network(object):
      def __init__(self):
          #Define Hyperparameters
          self.inputLayerSize = 2
          self.outputLayerSize = 1
          self.hiddenLayerSize = 3

          #Weights (Parameters)
          self.W1 = np.random.randn(self.inputLayerSize, \
                                      self.hiddenLayerSize)
          self.W2 = np.random.randn(self.hiddenLayerSize, \
                                      self.outputLayerSize)

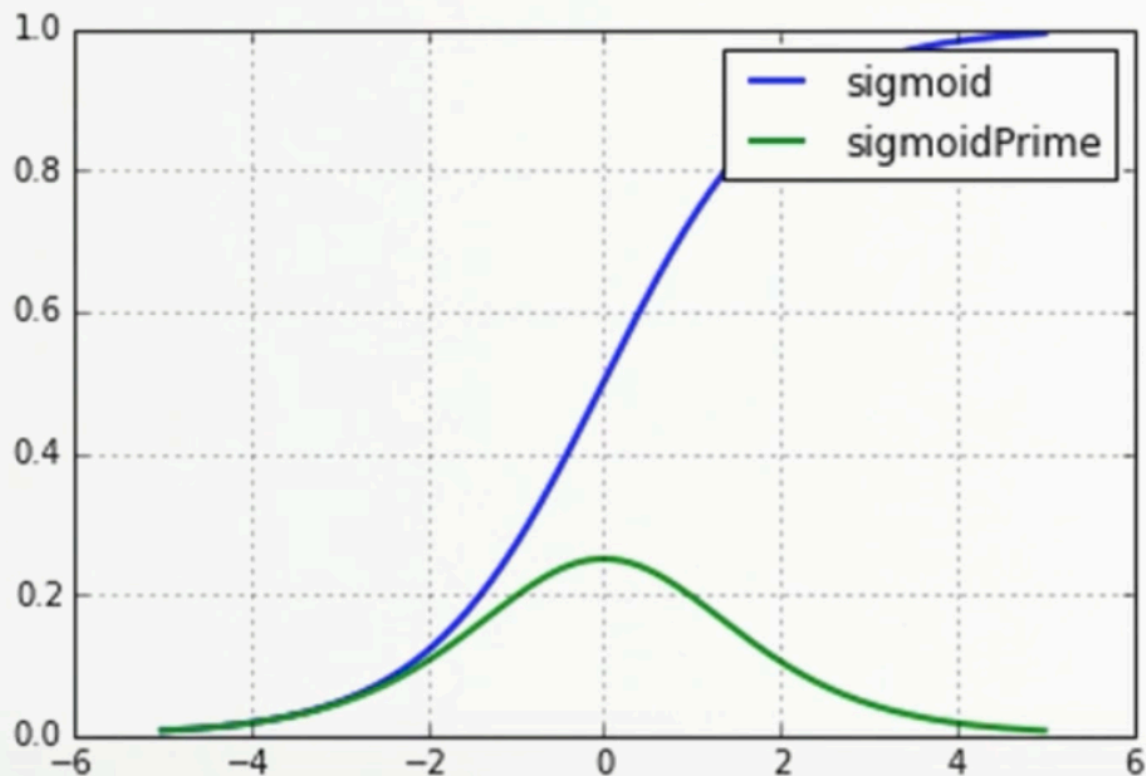
      def forward(self, X):
          #Propagate inputs though network
          self.z2 = np.dot(X, self.W1)
          self.a2 = self.sigmoid(self.z2)
          self.z3 = np.dot(self.a2, self.W2)
          yHat = self.sigmoid(self.z3)
          return yHat

      def sigmoid(self, z):
          #Apply sigmoid activation function to scalar, vector, or
          return 1/(1+np.exp(-z))
```

```
In [2]: def sigmoidPrime(z):  
        #Derivative of Sigmoid Function  
        return np.exp(-z)/((1+np.exp(-z))**2)
```

```
In [ ]: testValues = np.arange(-5,5,0.01)  
        plot(testValues, sigmoid(testValues), linewidth=2)  
        plot(testValues, sigmoidPrime(testValues), linewidth=2)  
        grid(1)  
        legend(['sigmoid', 'sigmoidPrime'])
```

Out[3]: <matplotlib.legend.Legend at 0x1068b25d0>



$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)}$$

$$\delta^{(3)} = -(y - \hat{y})f'(z^{(3)})$$

```
In [ ]: def costFunctionPrime(self, X, y):  
    #Compute derivative with respect to W1 and W2  
    self.yHat = self.forward(X)  
  
    delta3 = np.multiply(-(y-self.yHat), self.sigmoidPrime(self.z3))  
    dJdW2 = np.dot(self.a2.T, delta3)
```

$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)}$$

$$\delta^{(3)} = -(y - \hat{y})f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = X^T \delta^{(2)}$$

$$\delta^{(2)} = \delta^{(3)} (W^{(2)})^T f'(z^{(2)})$$

$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)}$$

$$\delta^{(3)} = -(y - \hat{y})f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = X^T \delta^{(2)}$$

$$\delta^{(2)} = \delta^{(3)} (W^{(2)})^T f'(z^{(2)})$$

```
In [ ]: def costFunctionPrime(self, X, y):
         #Compute derivative with respect to W1 and W2
         self.yHat = self.forward(X)

         delta3 = np.multiply(-(y-self.yHat), self.sigmoidPrime(self.z3))
         dJdW2 = np.dot(self.a2.T, delta3)

         delta2 = np.dot(delta3, self.W2.T)*self.sigmoidPrime(self.z2)
         dJdW1 = np.dot(X.T, delta2)

         return dJdW1, dJdW2
```

```
In [2]: NN = Neural_Network()
```

```
In [3]: cost1 = NN.costFunction(X,y)
```

```
In [4]: dJdW1, dJdW2 = NN.costFunctionPrime(X,y)
```

```
In [5]: dJdW1
```

```
Out[5]: array([[ -0.0071096 , -0.01059837, -0.00094283],  
               [-0.00172302, -0.00234379, -0.00019984]])
```


```
In [6]: dJdW2
```

```
Out[6]: array([[ -0.0229961 ],  
               [-0.01631712],  
               [-0.02079302]])
```


Gradient Descent Method

```
In [7]: scalar = 3  
NN.W1 = NN.W1 + scalar*dJdW1  
NN.W2 = NN.W2 + scalar*dJdW2  
cost2 = NN.costFunction(X,y)
```

```
In [8]: print cost1, cost2  
[ 0.01906658] [ 0.02396064]
```



```
In [9]: dJdW1, dJdW2 = NN.costFunctionPrime(X,y)  
NN.W1 = NN.W1 - scalar*dJdW1  
NN.W2 = NN.W2 - scalar*dJdW2  
cost3 = NN.costFunction(X,y)
```


```
In [10]: print cost2, cost3  
[ 0.02396064] [ 0.01773225]
```

```
In [ ]:
```


Today's Topics

- Artificial Neural Network
- **Graphical Modeling in Machine Learning**
 - Page Rank
 - Graph data representations
 - Markov Chain
 - Laplacian Matrix
 - Usage of Spectral (Eigenvalue-Eigenvector) information.
- Directed Graphical Models (Bayesian Networks)
- Plate Notation (*only if time permits*)
 - Naive Bayes as a Graphical Model
 - LDA

What happens when you Google Michael Jackson?



AllNewsVideosImagesBooksMore ▾Search tools

About 308,000,000 results (0.57 seconds)

Home | Michael Jackson Official Site

www.michaeljackson.com/ ▾

Sony Music site includes streaming audio and video files, discography, image gallery and competitions.


Michael Jackson - Wikipedia

https://en.wikipedia.org/wiki/Michael_Jackson ▾

Michael Joseph Jackson (August 29, 1958 – June 25, 2009) was an American singer and philanthropist. Called the "King of Pop", his contributions to music, ...

[Life and career](#) · [Death and memorial](#) · [Artistry](#) · [Legacy and influence](#)

Michael Jackson (@michaeljackson) | Twitter

<https://twitter.com/michaeljackson> 

1 day ago - [View on Twitter](#)

Today in '93 marked the final show on MJ's 17-month Dangerous World Tour. The final performance was in Mexico City,...

twitter.com/michaeljackson/status/1111111111

2 days ago - [View on Twitter](#)

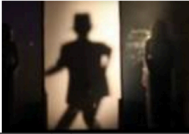
Starting tomorrow, Musical.ly will be joining in on the #Dangerous25 celebration with a special "Black or White" event! Download the app!

Michael Jackson - YouTube

<https://www.youtube.com/user/michaeljackson> ▾

The Official YouTube Channel of The King of Pop - Michael Jackson. For more info, visit www.michaeljackson.com.


In the news



Little Michael Jackson impersonator leaves crowd speechless

[East Coast Radio](#) - 2 days ago


At just seven he has already mastered some of **Michael Jackson's** iconic dance moves.





Michael Jackson

Singer

Available on

 iHeartRadio

 Pandora

 Tuneln

More music services

Michael Joseph Jackson was an American singer and philanthropist. Called the "King of Pop", his contributions to music, dance, and fashion along with his publicized personal life made him a global figure in popular culture for over four decades. [Wikipedia](#)

Died: June 25, 2009, [Holmby Hills, Los Angeles, CA](#)

Height: 5' 9"

Children: [Paris Michael Katherine Jackson](#), [Michael Joseph Jackson, Jr.](#)

Have you ever wondered...

- Why all the links on that page talk about the Pop star Michael Jackson instead of hundreds of other Michael Jackson who may be
 - a taxi driver,
 - an architect,
 - or an Engineer?
- The reason is that Google has a way of **scoring** pages, and the pages are presented from the most important one (with highest score) as the first page.
- This is called “**PageRank**”.
- Here “page” indicate the pages on the web as well as the last name of one of the funders of Google: Larry Page.

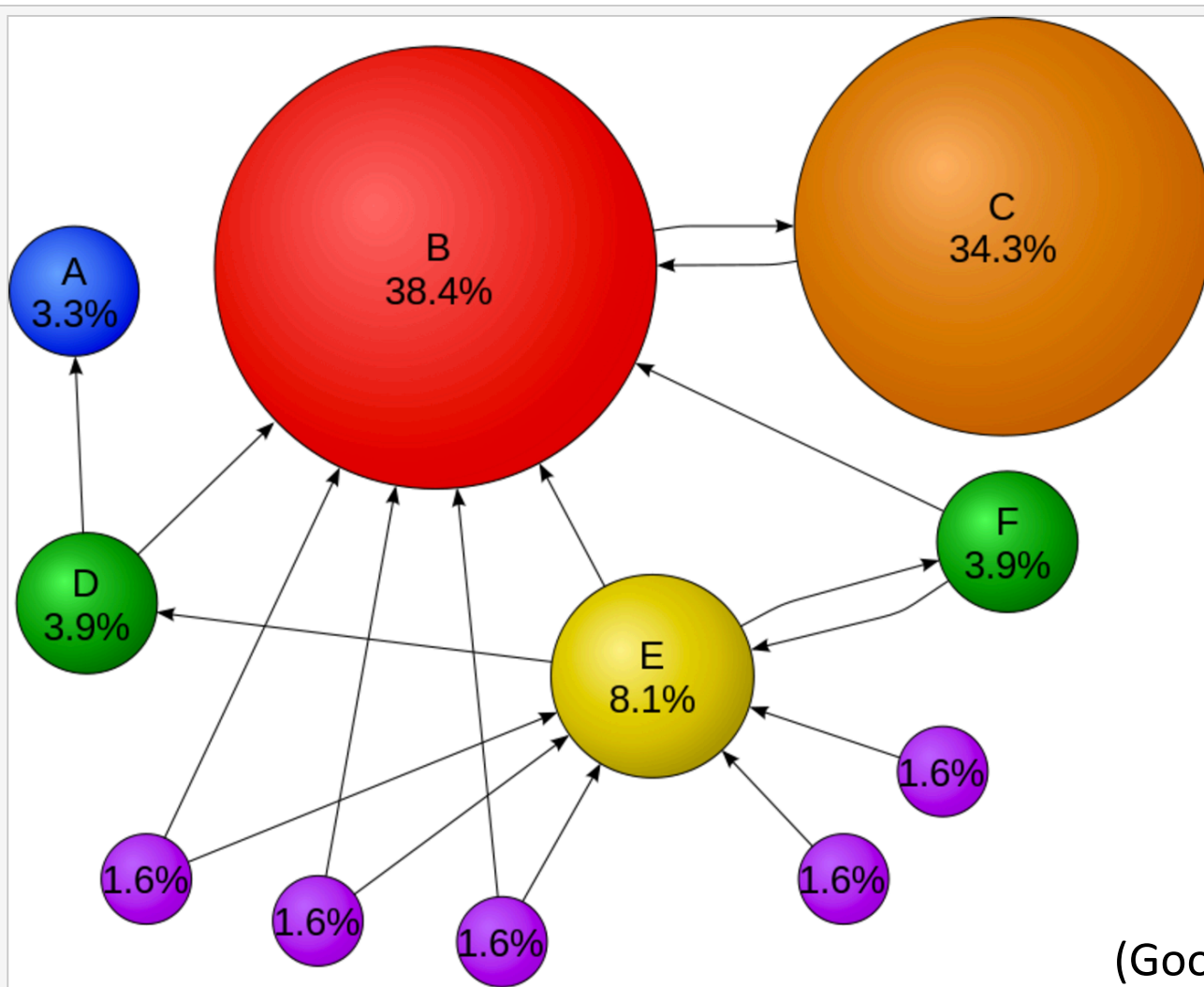
Mathematically, Web is a graph!

Web = the set of pages with links
between pages

- The Key idea in the graph theory is represent a graph as a matrix.
- Then some how make it to be symmetric.
- Then decompose it by using the method of eigenvalues and eigenvectors.
- Understand the meaning of the eigenvalues and eigenvectors related to the original graph.

- Work out details with the students on the board:
- Graph representations as data
- Adjacency matrix
- The Laplacian matrix of a graph
- Usage of Spectral (Eigenvalue-Eigenvector) information.

Famous example of using Graph Techniques in Large Data: PageRank



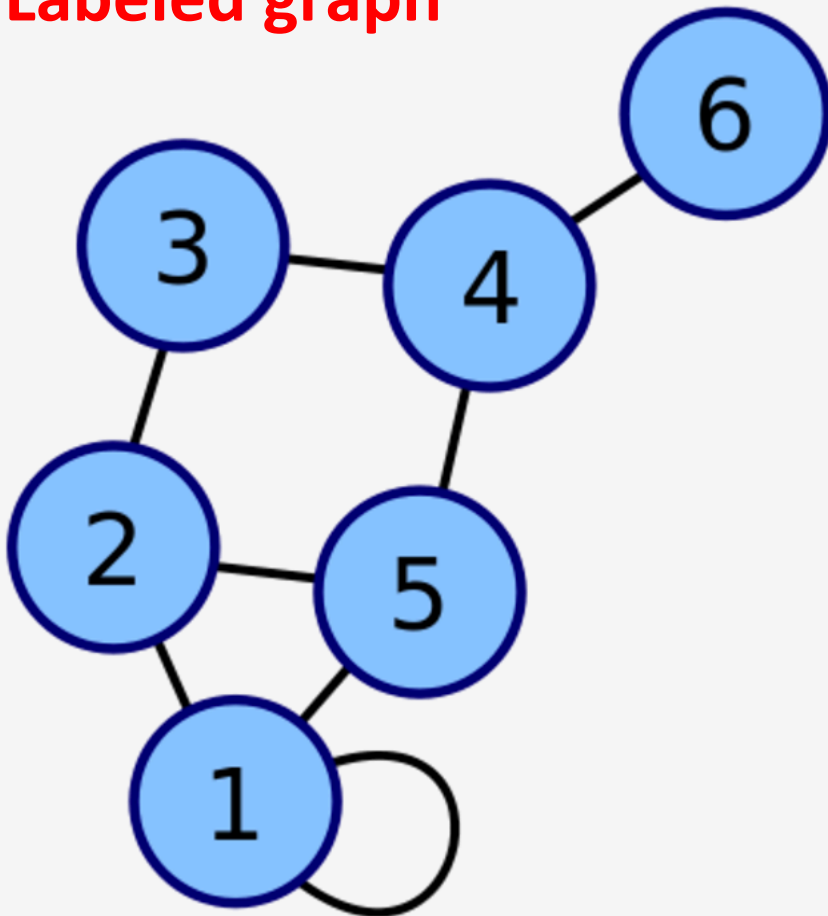
Page C has a higher PageRank than Page E, even though there are fewer links to C; the one link to C comes from an important page and hence is of high value.

(Google uses a logarithmic scale.)

Mathematical PageRanks for a simple network, expressed as percentages.

What is a graph and how to represent it mathematically?

Labeled graph

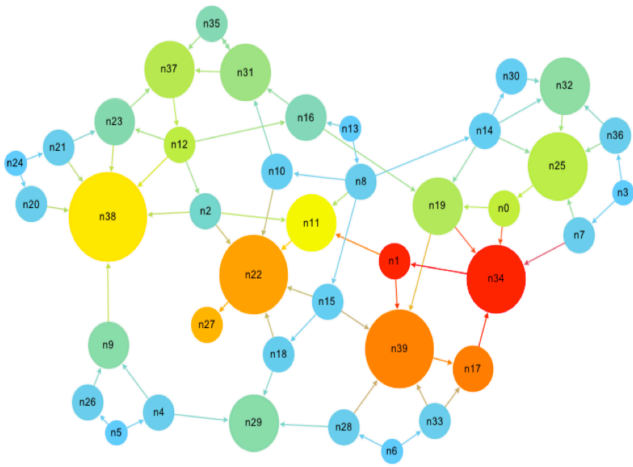
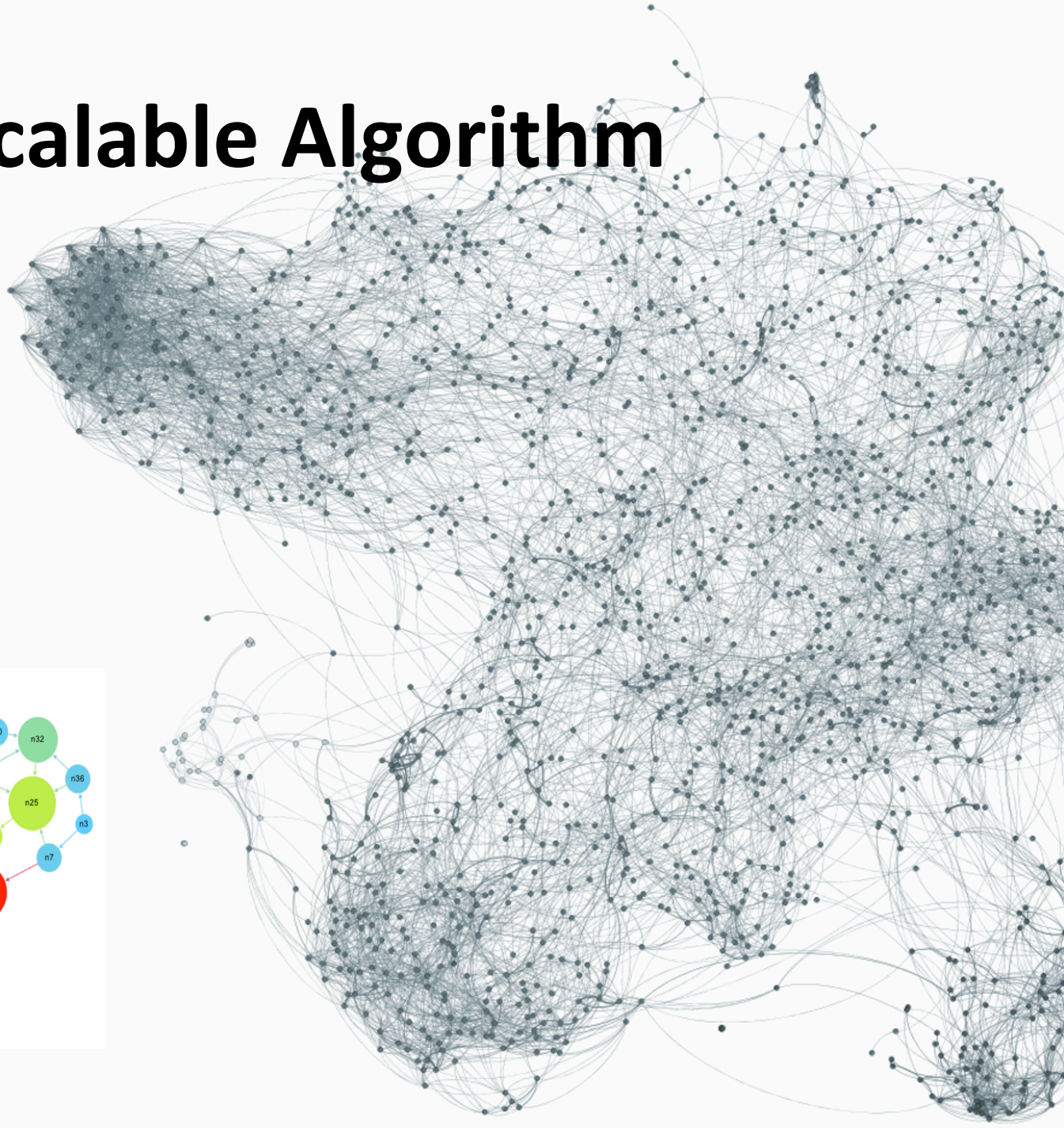
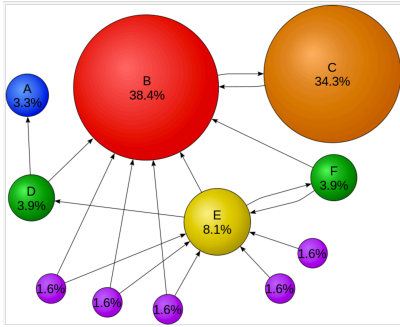


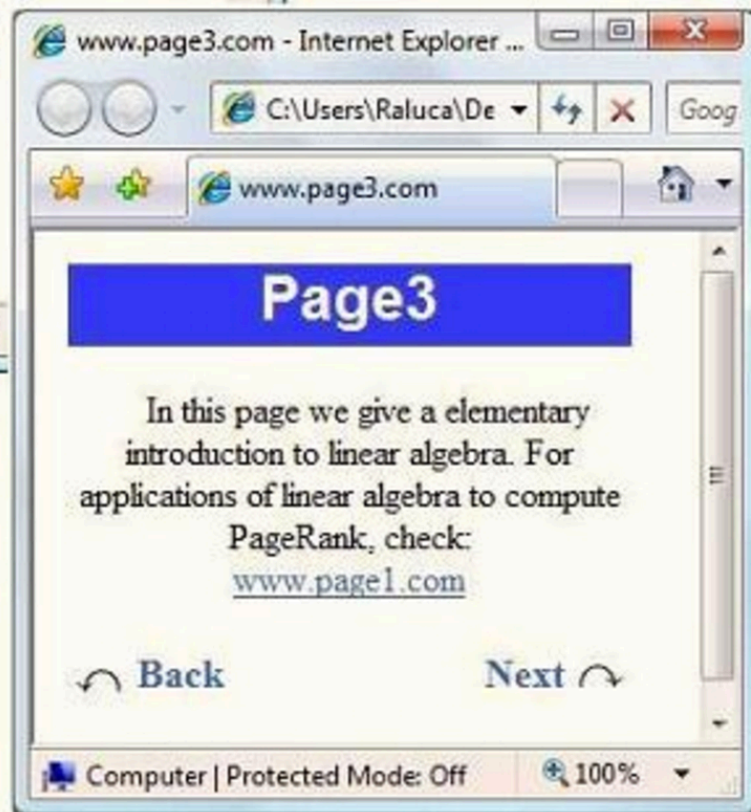
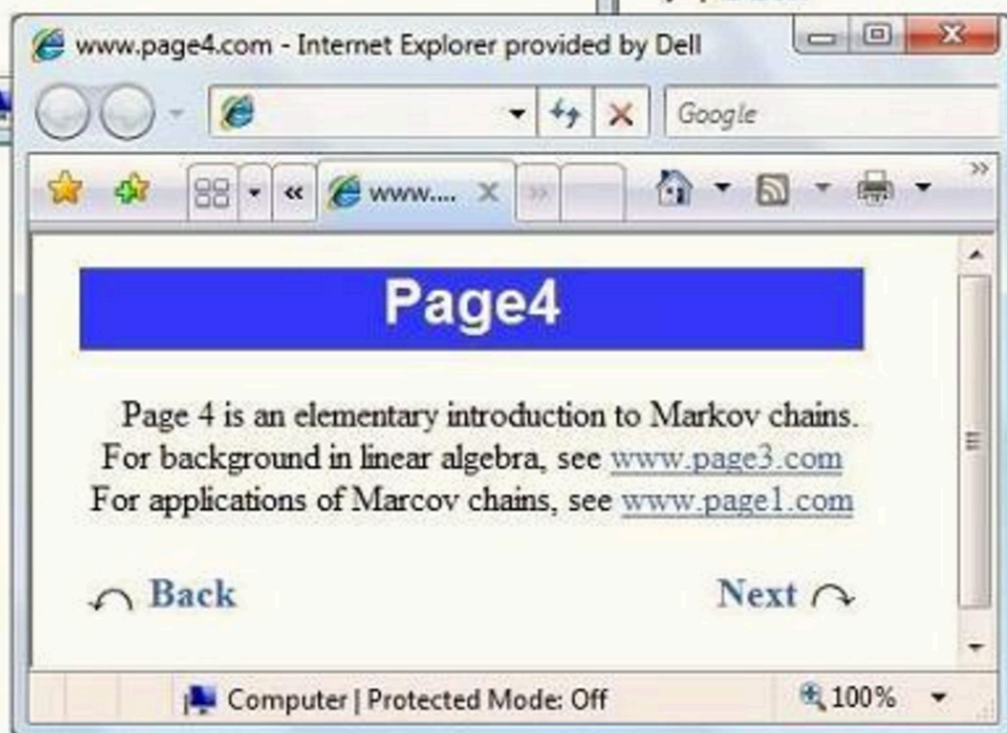
Adjacency matrix

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

Coordinates are 1–6.

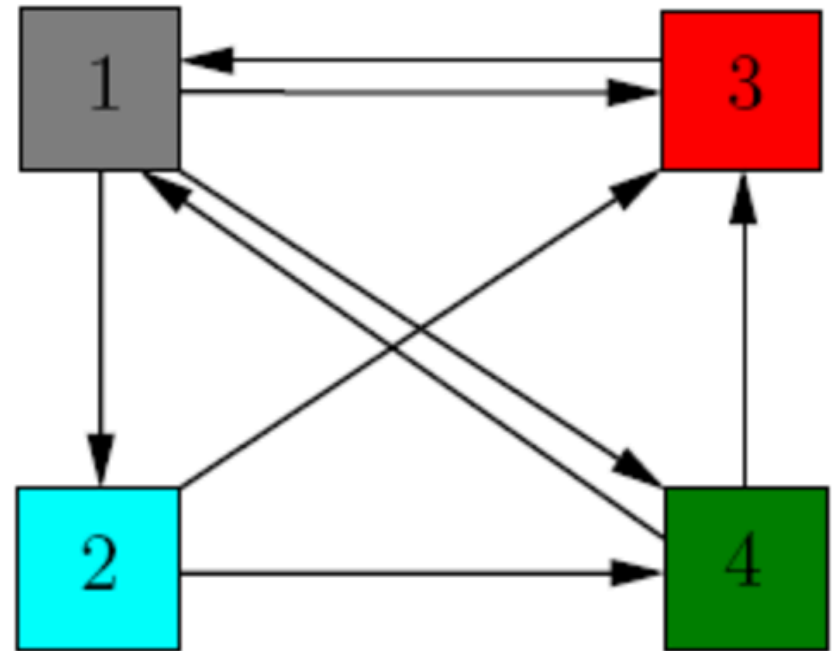
Scalable Algorithm



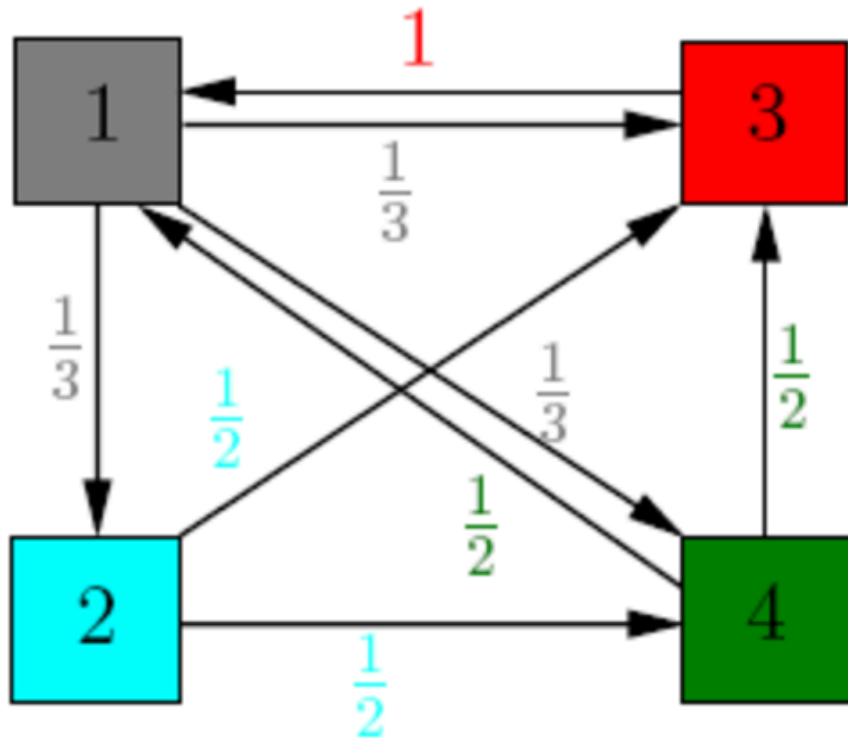


Graph Representation

We "translate" the picture into a directed graph with 4 nodes, one for each web site. When web site i references j , we add a directed edge between node i and node j in the graph. For the purpose of computing their page rank, we ignore any navigational links such as back, next buttons, as we only care about the connections between different web sites. For instance, Page1 links to all of the other pages, so node 1 in the graph will have outgoing edges to all of the other nodes. Page3 has only one link, to Page 1, therefore node 3 will have one outgoing edge to node 1. After analyzing each web page, we get the graph:



Normalize each column of the adjacency matrix, we obtain a Transition Matrix!



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

- Usually, each page transfer evenly its importance to the pages that it links to. Node 1 has 3 outgoing edges, so it will pass on $\frac{1}{3}$ of its importance to each of the other 3 nodes.
- Node 3 has only one outgoing edge, so it will pass on all of its importance to node 1.
- In general, if a node has k outgoing edges, it will pass on of its importance to each of the nodes that it links to.
- We then assigning weights to each edge.

Dynamical systems point of view

Suppose that initially the importance is uniformly distributed among the 4 nodes, each getting $\frac{1}{4}$. Denote by v the initial rank vector, having all entries equal to $\frac{1}{4}$. Each incoming link increases the importance of a web page, so at step 1, we update the rank of each page by adding to the current value the importance of the incoming links. This is the same as multiplying the matrix A with v . At step 1, the new importance vector is $v_1 = Av$. We can iterate the process, thus at step 2, the updated importance vector is $v_2 = A(Av) = A^2v$. Numeric computations give:

$$v = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}, \quad Av = \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix}, \quad A^2 v = A(Av) = A \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix} = \begin{pmatrix} 0.43 \\ 0.12 \\ 0.27 \\ 0.16 \end{pmatrix}$$

$$A^3 v = \begin{pmatrix} 0.35 \\ 0.14 \\ 0.29 \\ 0.20 \end{pmatrix}, \quad A^4 v = \begin{pmatrix} 0.39 \\ 0.11 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^5 v = \begin{pmatrix} 0.39 \\ 0.13 \\ 0.28 \\ 0.19 \end{pmatrix}$$

$$A^6 v = \begin{pmatrix} 0.38 \\ 0.13 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^7 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^8 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$$

PageRank Vector as limit of $A^k v$ as k tends to infinity

We notice that the
sequences of iterates v ,
 Av , ..., $A^k v$ tends to the
equilibrium value:

$$v^* = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$$



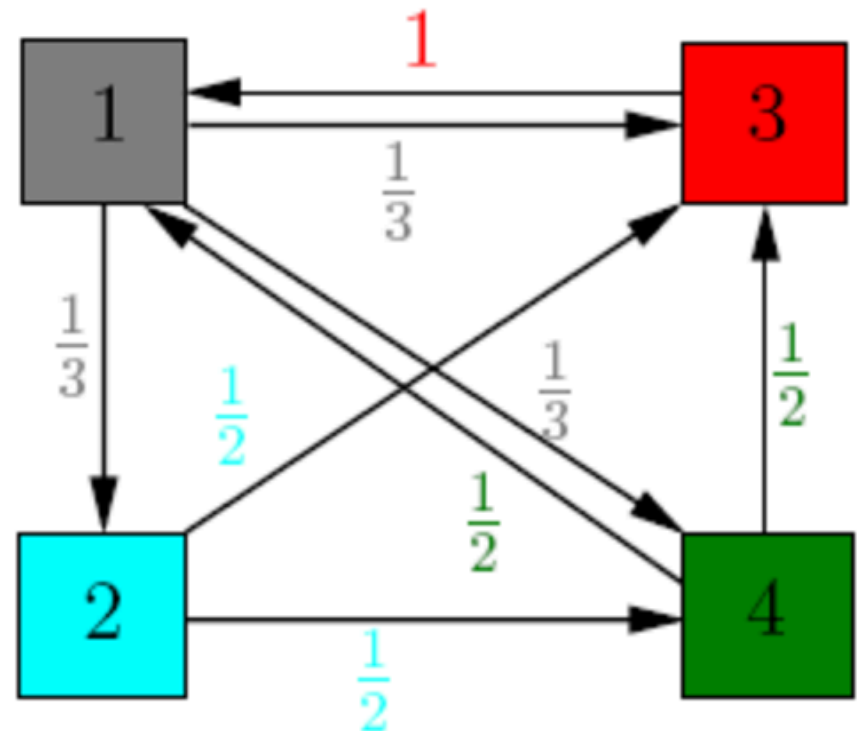
We call this the
PageRank vector
of our web graph.

Linear algebra point of view

Let us denote by x_1 , x_2 , x_3 , and x_4 the importance of the four pages. Analyzing the situation at each node we get the system:

$$\begin{cases} x_1 = 1 \cdot x_3 + \frac{1}{2} \cdot x_4 \\ x_2 = \frac{1}{3} \cdot x_1 \\ x_3 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 + \frac{1}{2} \cdot x_4 \\ x_4 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 \end{cases}$$

$$\begin{matrix} & \text{A} & & \text{X} & \\ \begin{matrix} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{matrix} & \begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix} & = & \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \end{matrix}$$



I.e. X is an eigenvector of A associ. w/ eigenvalue 1.

The Page Rank Vector is an eigenvector of A associated to eigenvalue 1.

- If we solve find the eigenvector satisfying
- $AX = 1X$, we get

$$c \cdot \begin{bmatrix} 12 \\ 4 \\ 9 \\ 6 \end{bmatrix}.$$

Here c is a constant.

We choose v^* to be the unique eigenvector with the sum of all entries equal to 1.

$$\frac{1}{31} \cdot \begin{bmatrix} 12 \\ 4 \\ 9 \\ 6 \end{bmatrix} \sim \begin{bmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{bmatrix}$$

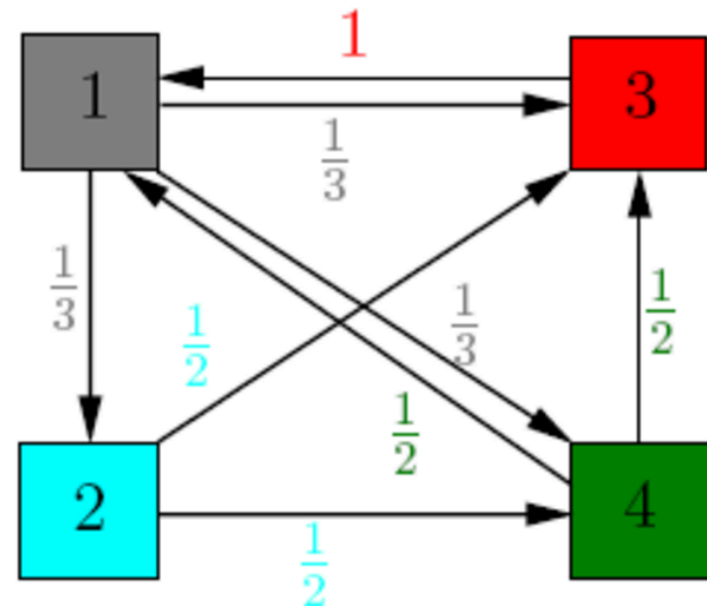
Same as the page rank vector.

Probabilistic point of view:

- Since the importance of a web page is measured by its popularity (how many incoming links it has), we can view the importance of page i as the probability that a random surfer on the Internet that opens a browser to any page and starts following hyperlinks, visits the page i . We can interpret the weights we assigned to the edges of the graph in a probabilistic way:
- A random surfer that is currently viewing web page 2, has $\frac{1}{2}$ probability to go to page 3, and $\frac{1}{2}$ probability to go to page 4.
- We can model the process as a random walk on graphs.
- Each page has equal probability $\frac{1}{4}$ to be chosen as a starting point. So, the initial probability distribution is given by the column vector $[\frac{1}{4} \ \frac{1}{4} \ \frac{1}{4} \ \frac{1}{4}]^t$.
- The probability that page i will be visited after one step is equal to Ax , and so on.
- The probability that page i will be visited after k steps is equal to A^kx .
- The sequence $Ax, A^2x, A^3x, \dots, A^kx, \dots$ converges in this case to a unique probabilistic vector v^* .
- In this context v^* is called the stationary distribution and it will be our Page Rank vector.
- Moreover, the i th entry in the vector v^* is simply the probability that at each moment a random surfer visits page i .
- The computations are identical to the ones we did in the dynamical systems interpretation, only the meaning we attribute to each step being slightly different.

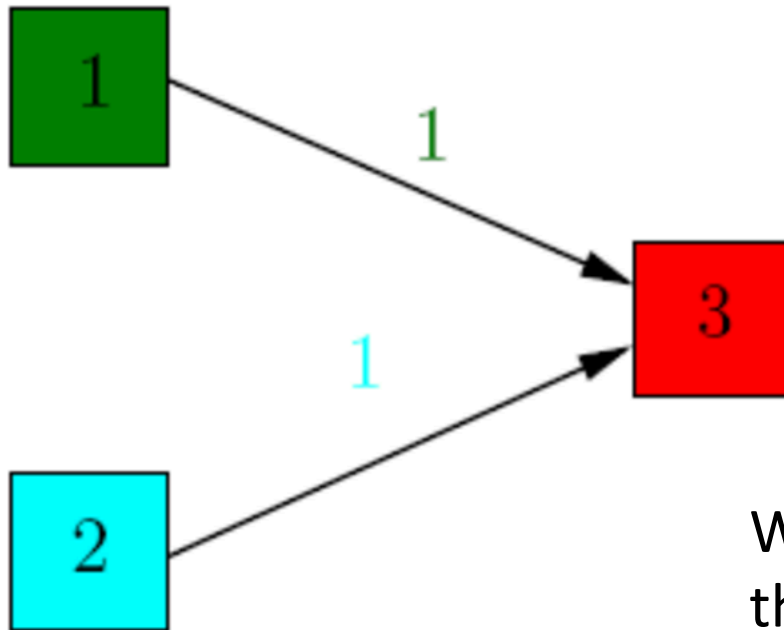
Discussion

- The Page Rank vector v^* we have computed by different methods, indicates that page 1 is the most relevant page.
- This might seem surprising since page 1 has 2 backlinks, while page 3 has 3 backlinks.
- If we take a look at the graph, we see that node 3 has only one outgoing edge to node 1, so it transfers all its importance to node 1.
- Equivalently, once a web surfer that only follows hyperlinks visits page 3, he can only go to page 1.
- Notice also how the rank of each page is not trivially just the weighted sum of the edges that enter the node.
- Intuitively, at step 1, one node receives an importance vote from its direct neighbors, at step 2 from the neighbors of its neighbors, and so on.



Subtleties and Issues

1. Nodes with no outgoing edges (dangling nodes)



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

We iteratively compute the rank of the 3 pages:

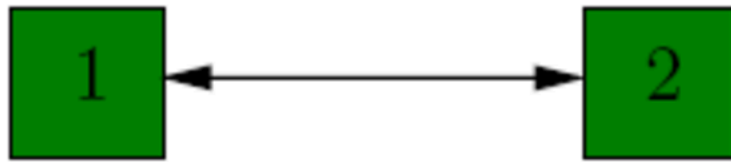
$$v_0 = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}, \quad v_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{2}{3} \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ \frac{2}{3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

So in this case the rank of every page is 0. This is counterintuitive, as page 3 has 2 incoming links, so it must have some importance!

Subtleties and Issues

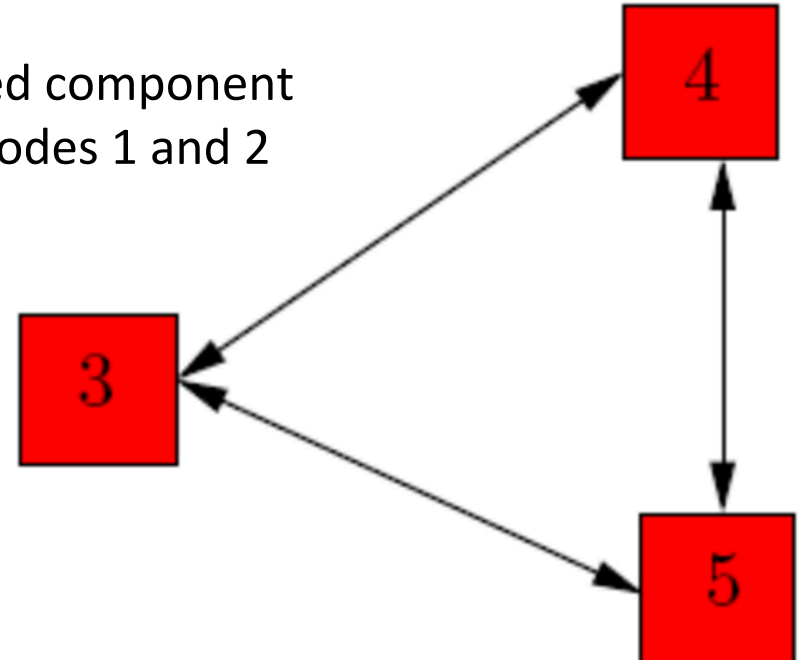
2. Disconnected components

A random surfer that starts in the first connected component has no way of getting to web page 5 since the nodes 1 and 2 have no links to node 5 that he can follow.



$$v = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad u = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

are both eigenvectors corresponding to the eigenvalue 1, and they are not just trivially one the a scalar multiple of the other.



$$A = \left[\begin{array}{cc|ccc} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{array} \right]$$

- So, both in theory and in practice, the notation of ranking pages from the first connected component relative to the ones from the second connected component is ambiguous.
- The web is very heterogeneous by its nature, and certainly huge, so we do not expect its graph to be connected.
- Likewise, there will be pages that are plain descriptive and contain no outgoing links.
- We need a non ambiguous meaning of the rank of a page, for any directed Web graph with n nodes.

How to solve the above problems?

The solution of Page and Brin

(The founders of Google.Inc)

- In order to overcome these problems, fix a positive constant p between 0 and 1, which we call the damping factor (a typical value for p is 0.15).
- Define the Page Rank matrix (also known as the Google matrix) of the graph by PageRank matrix

$$M = (1 - p) \cdot A + p \cdot B$$

- where

$$B = \frac{1}{n} \cdot \begin{bmatrix} 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}.$$

Characteristics of Google PageRank

- M remains a column stochastic matrix.
- M has only positive entries
- *Homework: Prove above factors*
- So what?
- These characteristics guarantees that M has a unique dominant eigenvalue ($= 1$) and its eigenvectors are unique (after normalize it) by **Perron-Frobenius Theorem.**

- The matrix M models the random surfer model as follows: most of the time, a surfer will follow links from a page: from a page i the surfer will follow the outgoing links and move on to one of the neighbors of i . A smaller, but positive percentage of the time, the surfer will dump the current page and choose arbitrarily a different page from the web and "teleport" there. The damping factor p reflects the probability that the surfer quits the current page and "teleports" to a new one. Since he/she can teleport to any web page, each page has p probability to be chosen. This justifies the structure of the matrix B .
- Intuitively, the matrix M "connects" the graph and gets rid of the dangling nodes. A node with no outgoing edges has now p probability to move to any other node.

Perron-Frobenius Theorem

- If M is a positive, column stochastic matrix, then:
- 1 is an eigenvalue of multiplicity one.
- 1 is the largest eigenvalue: all the other eigenvalues have absolute value smaller than 1.
- The eigenvectors corresponding to the eigenvalue 1 have either only positive entries or only negative entries.
- In particular, for the eigenvalue 1 there exists a unique eigenvector with the sum of its entries equal to 1.

Power Method Convergence Theorem

- Let M be a positive, column stochastic $n \times n$ matrix.
- Denote by v^* its probabilistic eigenvector corresponding to the eigenvalue 1.
- Let z be the column vector with all entries equal to $1/n$.
- Then the sequence $z, Mz, \dots, M^k z$ converges to the vector v^* .

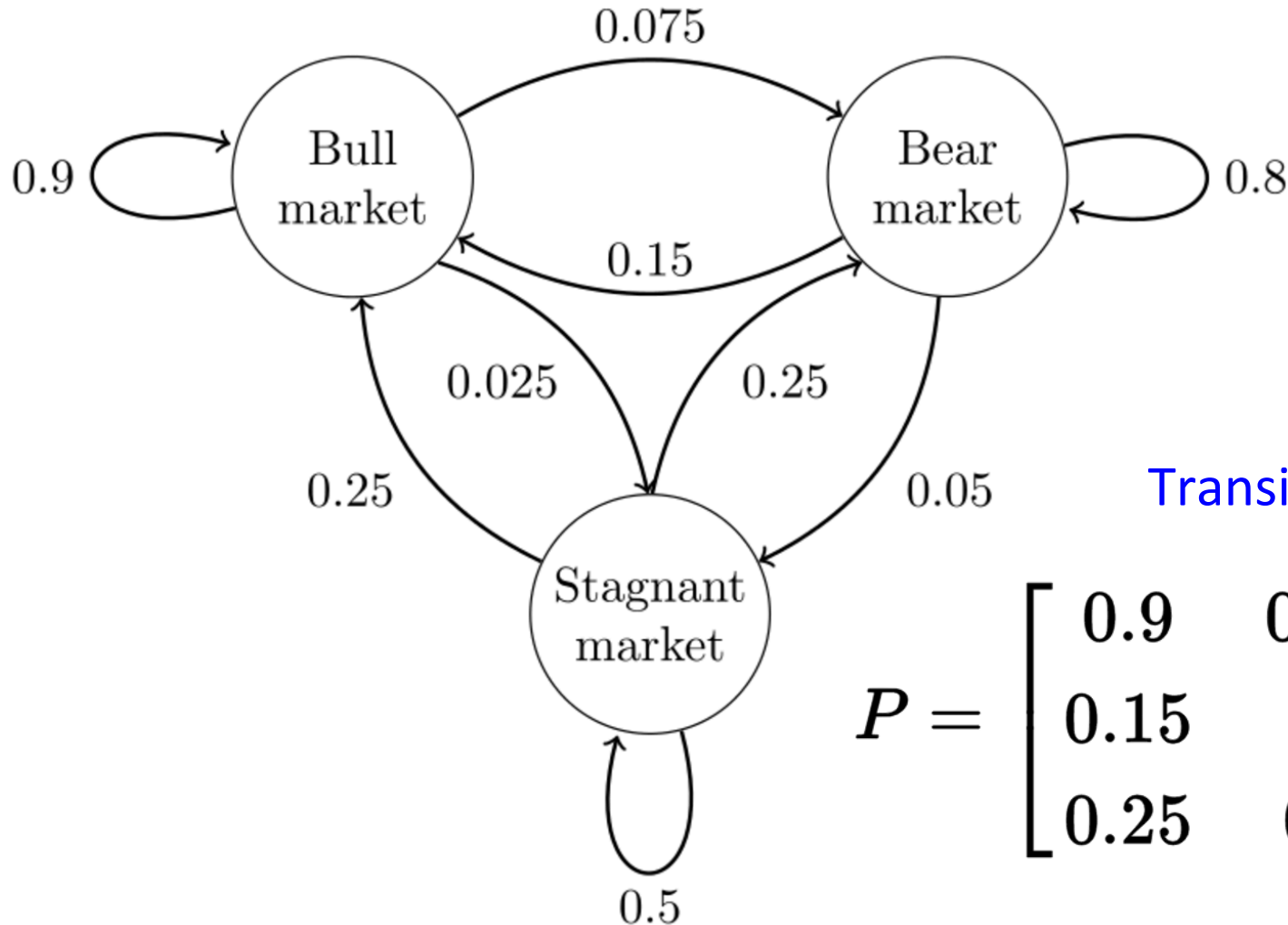
Why does PageRank method work?

- Fact: The PageRank vector for a web graph with transition matrix A , and damping factor p , is the unique probabilistic eigenvector of the matrix M , corresponding to the eigenvalue 1.

Today's Topics

- Artificial Neural Network
- Graphical Modeling in Machine Learning
 - Page Rank
 - Graph data representations
 - **Markov Chain**
 - Laplacian Matrix
 - Usage of Spectral (Eigenvalue-Eigenvector) information.
- Directed Graphical Models (Bayesian Networks)
- Plate Notation (*only if time permits*)
 - Naive Bayes as a Graphical Model
 - LDA

Markov Chain



Transition Matrix:

$$P = \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}$$

Stochastic Matrix = probability matrix = transition matrix = Markov matrix

- A stochastic matrix is a matrix used to describe the transitions of a Markov chain. Each of its entries is a nonnegative real number representing a probability.
- There are several different definitions and types of stochastic matrices:
 - A right stochastic matrix is a real square matrix, with each row summing to 1.
 - A left stochastic matrix is a real square matrix, with each column summing to 1.
 - A doubly stochastic matrix is a square matrix of nonnegative real numbers with each row and column summing to 1.

- The state space $\{1 = \text{bull}, 2 = \text{bear}, 3 = \text{stagnant}\}$
- The distribution over states can be written as a stochastic row vector x with the relation

$$x^{(n+1)} = x^{(n)} P.$$

- So if at time n the system is in state $x^{(n)}$, then three time periods later, at time $n + 3$ the distribution is

$$x^{(n+3)} = x^{(n+2)} P = \left(x^{(n+1)} P \right) P$$

$$= x^{(n+1)} P^2 = \left(x^{(n)} P \right) P^2$$

$$= x^{(n)} P^3$$

In particular, if at time n the system is in state 2 (bear), then at time $n + 3$ the distribution is:

$$\begin{aligned} x^{(n+3)} &= [0 \quad 1 \quad 0] \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}^3 \\ &= [0 \quad 1 \quad 0] \begin{bmatrix} 0.7745 & 0.17875 & 0.04675 \\ 0.3575 & 0.56825 & 0.07425 \\ 0.4675 & 0.37125 & 0.16125 \end{bmatrix} \\ &= [0.3575 \quad 0.56825 \quad 0.07425]. \end{aligned}$$

Using the transition matrix it is possible to calculate, for example, the long-term fraction of weeks during which the market is stagnant, or the average number of weeks it will take to go from a stagnant to a bull market. Using the transition probabilities, the steady-state probabilities indicate that 62.5% of weeks will be in a bull market, 31.25% of weeks will be in a bear market and 6.25% of weeks will be stagnant, since:

$$\lim_{N \rightarrow \infty} P^N = \begin{bmatrix} 0.625 & 0.3125 & 0.0625 \\ 0.625 & 0.3125 & 0.0625 \\ 0.625 & 0.3125 & 0.0625 \end{bmatrix}$$

Today's Topics

- Artificial Neural Network
- Graphical Modeling in Machine Learning
 - Page Rank
 - Graph data representations
 - Markov Chain
 - **Laplacian Matrix**
 - Usage of Spectral (Eigenvalue-Eigenvector) information.
- Directed Graphical Models (Bayesian Networks)
- Plate Notation (*only if time permits*)
 - Naive Bayes as a Graphical Model
 - LDA

Laplacian Matrix

- Given a simple graph G with n vertices, its Laplacian matrix is defined as

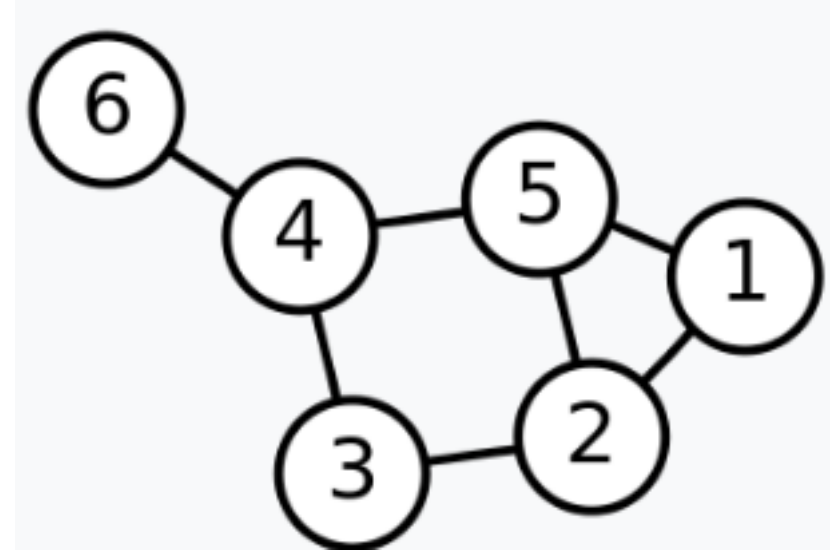
$$L = D - A,$$

where D is the degree matrix and A is the adjacency matrix of the graph.

Since G is a simple graph, A only contains 1s or 0s and its diagonal elements are all 0s.

- Notice L is symmetric, so there exists a matrix of same size which is orthogonal so that $P^t L P = D$, where D is diagonal matrix with diagonal elements are eigenvalues of L .

Example



Degree matrix	Adjacency matrix	Laplacian matrix
$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Today's Topics

- Artificial Neural Network
- Graphical Modeling in Machine Learning
 - Page Rank
 - Graph data representations
 - Markov Chain
 - Laplacian Matrix
 - **Usage of Spectral (Eigenvalue-Eigenvector) information.**
- Directed Graphical Models (Bayesian Networks)
- Plate Notation (*only if time permits*)
 - Naive Bayes as a Graphical Model
 - LDA

Two most important Properties: Assume the eigenvalues of L is arranged at $\lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{n-1}$

- **The number of connected components in the graph is the algebraic multiplicity of the 0 eigenvalue, i.e. the dimension of the nullspace of the Laplacian.**
- **The second smallest eigenvalue of L is the approximates the sparsest cut of a graph.**

[There is a name for the second smallest eigenvalue: algebraic connectivity (or Fiedler value) of G .]

Other properties of Laplacian Matrix

- L is positive semi-definite, i.e. all the eigenvalues are ≥ 0 . (We know L is symmetric and diagonally dominant.)
- Every row sum and column sum of L is zero.

This implies:

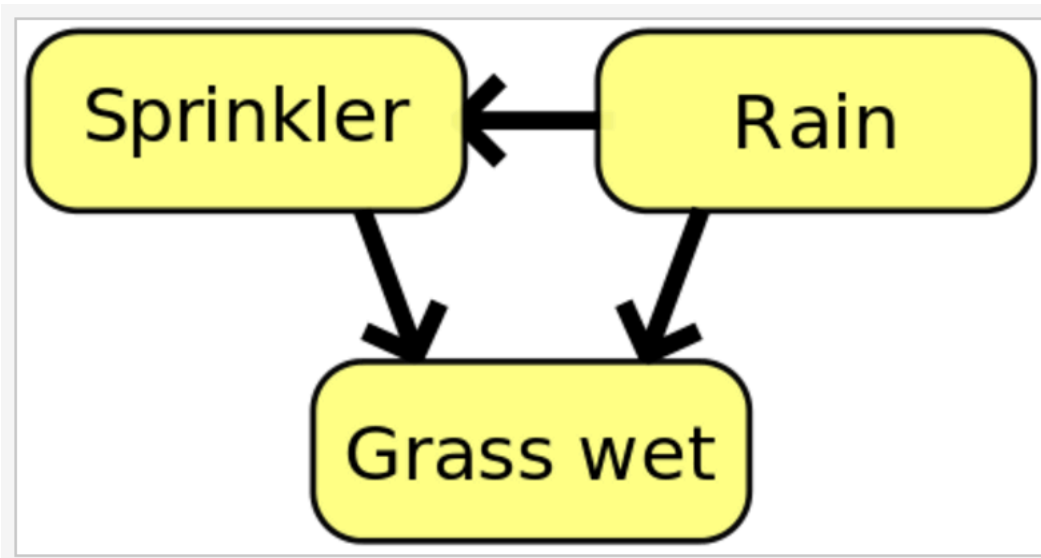
$\lambda_0 = 0$, because the vector $\mathbf{v}_0 = (1, 1, \dots, 1)$ satisfies $L\mathbf{v}_0 = \mathbf{0}$.

- Laplacian matrix is singular.
- And more.

Today's Topics

- Artificial Neural Network
- Graphical Modeling in Machine Learning
 - Page Rank
 - Graph data representations
 - Markov Chain
 - Laplacian Matrix
 - Usage of Spectral (Eigenvalue-Eigenvector) information.
- **Directed Graphical Models (Bayesian Networks)**
- Plate Notation (*only if time permits*)
 - Naive Bayes as a Graphical Model
 - LDA

An Example of Bayesian Network



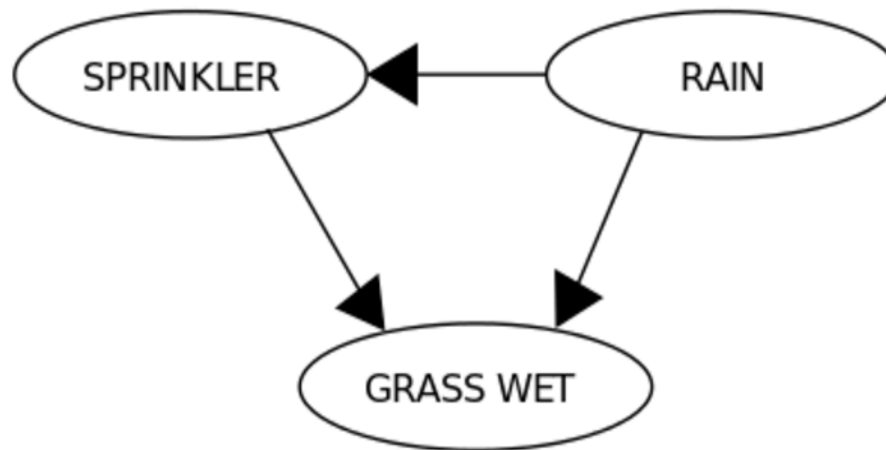
Rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet.

A simple Bayesian network

- Bayesian network
- = Bayes network
- = belief network
- = Bayes(ian) model
- = probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG).

A simple **Bayesian Network** with conditional probability tables

RAIN	SPRINKLER	
	T	F
F	0.4	0.6
T	0.01	0.99



	RAIN	
	T	F
	0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

The **joint probability function** is:

$$\Pr(G, S, R) = \Pr(G|S, R) \Pr(S|R) \Pr(R)$$

where the names of the variables have been abbreviated to $G = \text{Grass wet (yes/no)}$, $S = \text{Sprinkler turned on (yes/no)}$, and $R = \text{Raining (yes/no)}$.

The model can answer questions like "What is the probability that it is raining, given the grass is wet?" by using the **conditional probability** formula and summing over all **nuisance variables**:

$$\Pr(R = T|G = T) = \frac{\Pr(G = T, R = T)}{\Pr(G = T)} = \frac{\sum_{S \in \{T, F\}} \Pr(G = T, S, R = T)}{\sum_{S, R \in \{T, F\}} \Pr(G = T, S, R)}$$

$$\Pr(R = T|G = T) = \frac{\Pr(G = T, R = T)}{\Pr(G = T)} = \frac{\sum_{S \in \{T, F\}} \Pr(G = T, S, R = T)}{\sum_{S, R \in \{T, F\}} \Pr(G = T, S, R)}$$

Using the expansion for the joint probability function $\Pr(G, S, R)$ and the conditional probabilities from the [conditional probability tables \(CPTs\)](#) stated in the diagram, one can evaluate each term in the sums in the numerator and denominator. For example,

$$\begin{aligned}\Pr(G = T, S = T, R = T) &= \Pr(G = T|S = T, R = T) \Pr(S = T|R = T) \Pr(R = T) \\ &= 0.99 \times 0.01 \times 0.2 \\ &= 0.00198.\end{aligned}$$

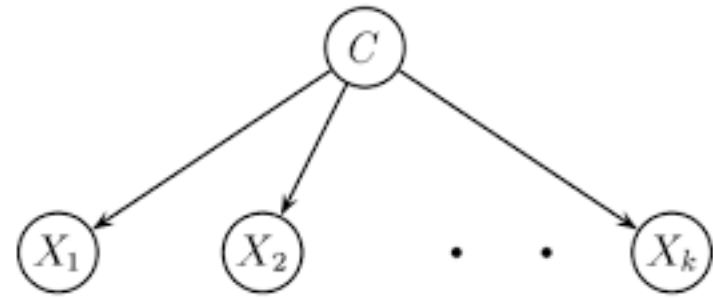
Then the numerical results (subscripted by the associated variable values) are

$$\Pr(R = T|G = T) = \frac{0.00198_{TTT} + 0.1584_{TFT}}{0.00198_{TTT} + 0.288_{TTF} + 0.1584_{TFT} + 0.0_{TFF}} = \frac{891}{2491} \approx 35.77\%.$$

Today's Topics

- Artificial Neural Network
- Graphical Modeling in Machine Learning
 - Page Rank
 - Graph data representations
 - Markov Chain
 - Laplacian Matrix
 - Usage of Spectral (Eigenvalue-Eigenvector) information.
- Directed Graphical Models (Bayesian Networks)
- **Plate Notation (*only if time permits*)**
 - Naive Bayes as a Graphical Model
 - LDA

Plate Notation of Naïve Bayes



$$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$$

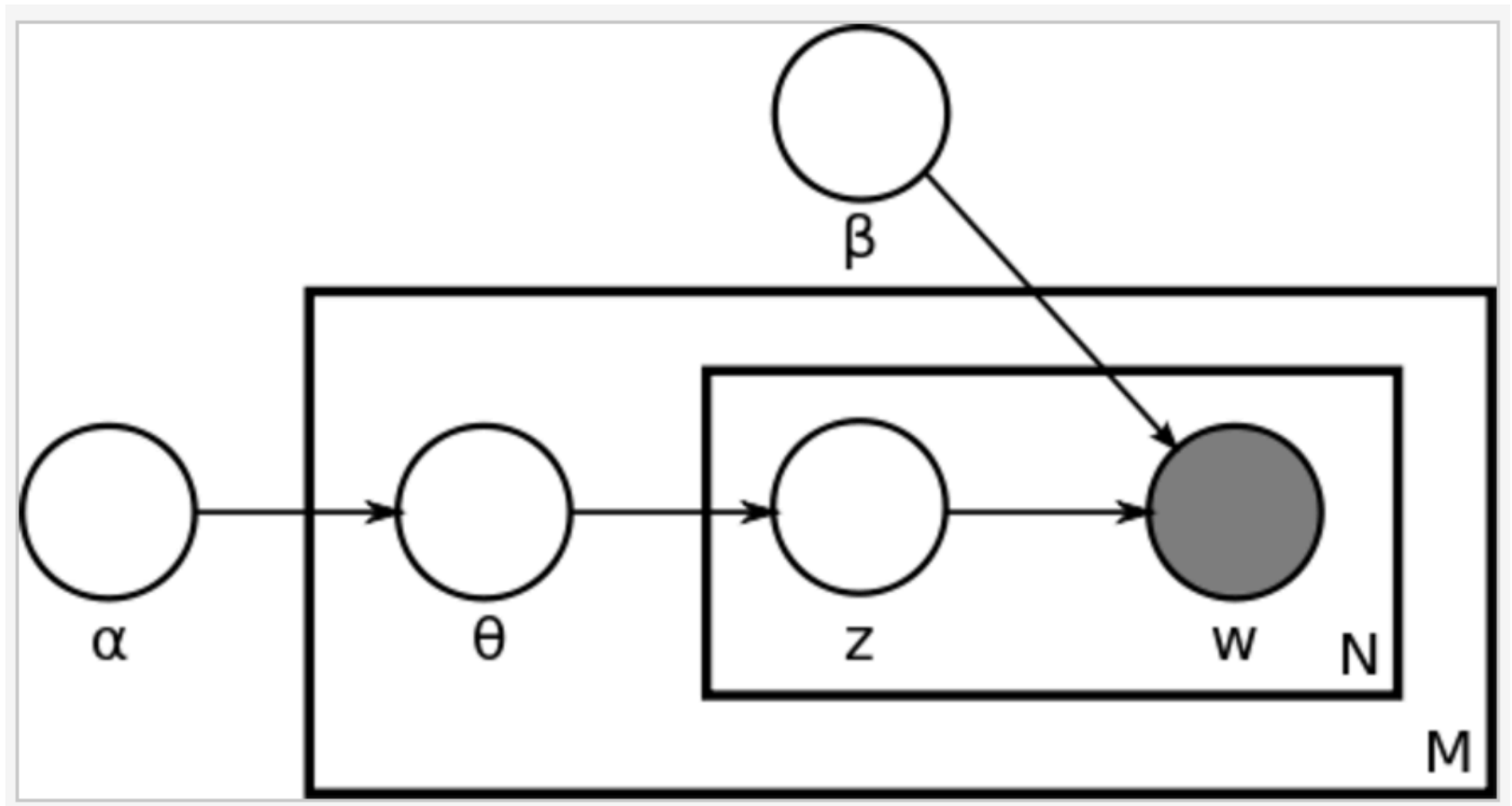
Diagram illustrating the components of the Naïve Bayes formula:

- $P(c | x)$ is labeled **Posterior Probability** (indicated by a downward arrow).
- $P(x | c)$ is labeled **Likelihood** (indicated by an upward arrow).
- $P(c)$ is labeled **Class Prior Probability** (indicated by an upward arrow).
- $P(x)$ is labeled **Predictor Prior Probability** (indicated by a downward arrow).

$$P(c | X) \propto \underbrace{P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c)}_{\text{Likelihood}} \times P(c)$$

To maximize this product, we take log of it

Plate Notation representing the LDA model



Virtual way to see it

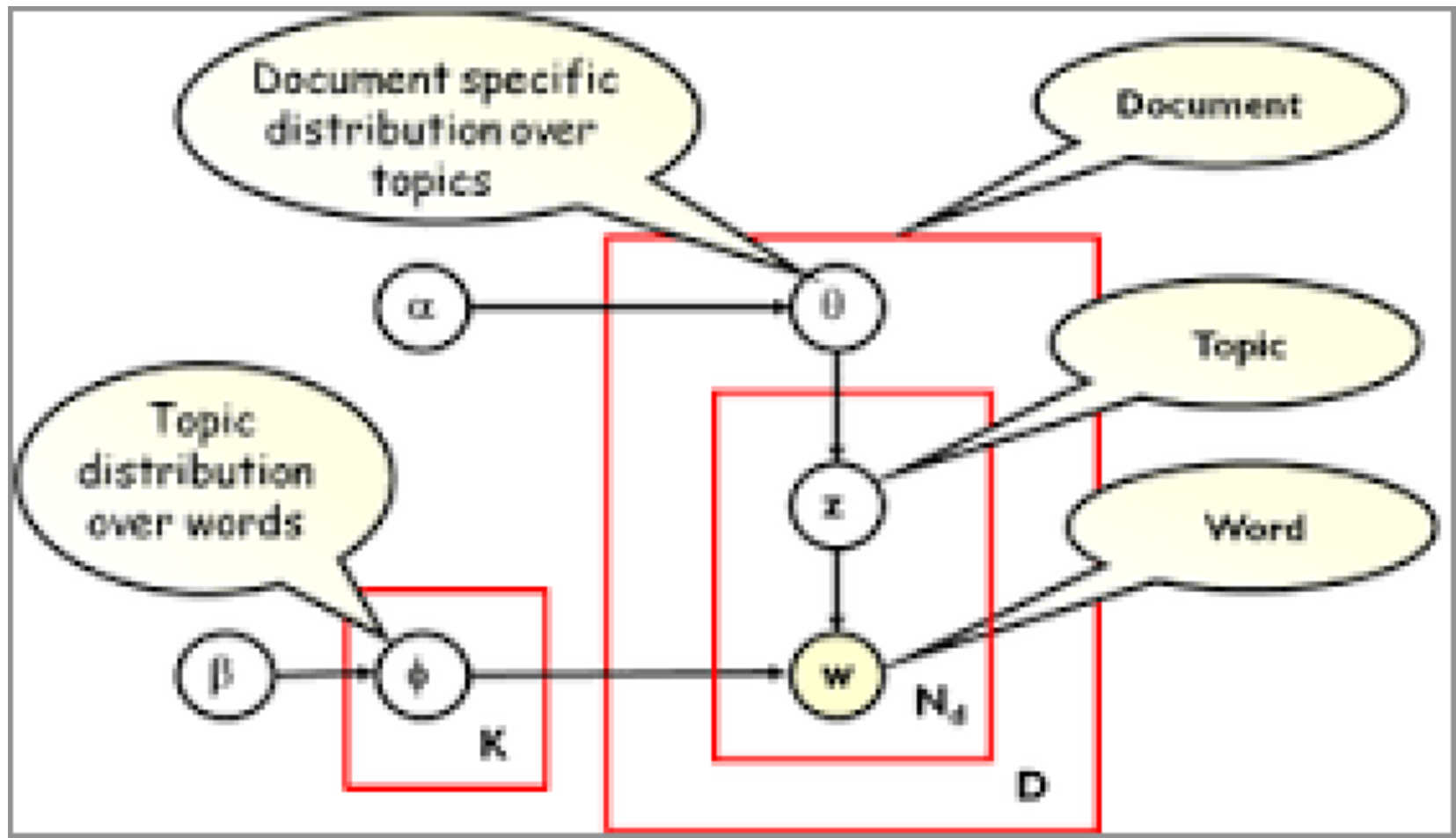
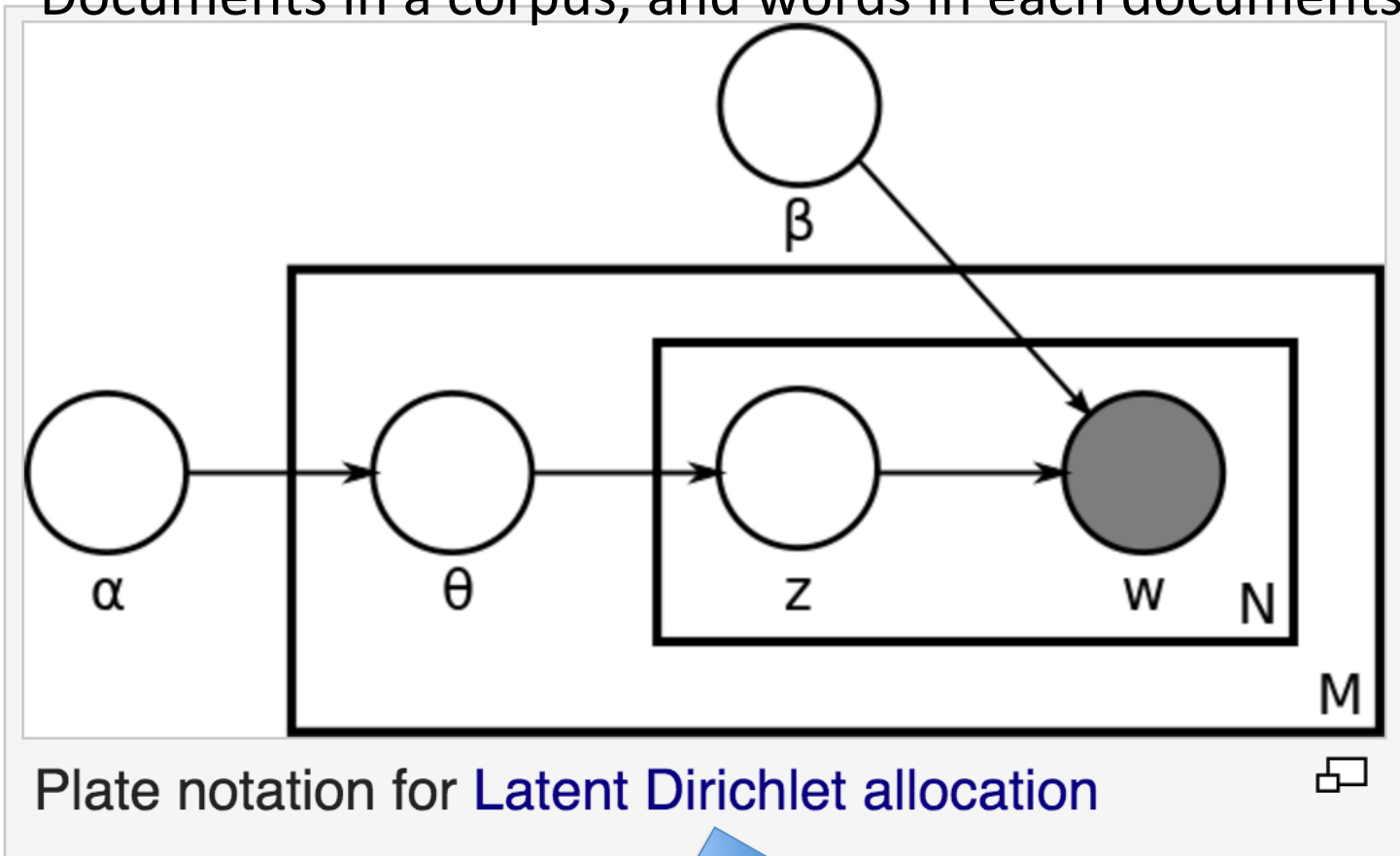


Plate Notation

Example: Modeling topically related:
Documents in a corpus, and words in each documents.



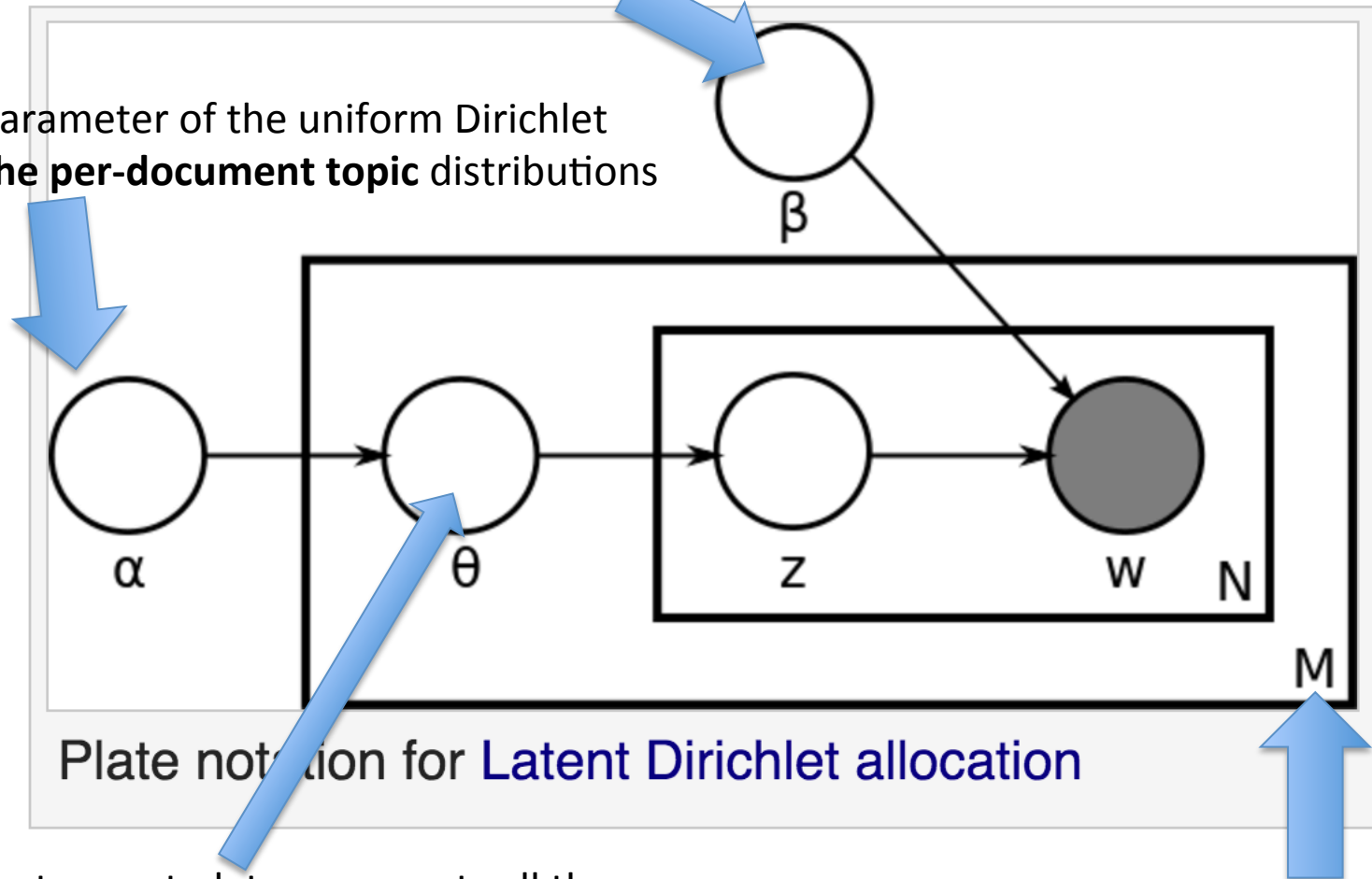
A Bayesian network that models how documents in a corpus are topically related.

Plate Notation

Note: α and β not in any plate.

β is the parameter of the uniform Dirichlet prior **on the per-topic word** distribution.

α is the parameter of the uniform Dirichlet prior **on the per-document topic** distributions



The outermost plate represents all the variables related to a specific document, including θ_i , the topic distribution for document i .

The M indicates that the variables inside are repeated M times, once for each document.

Plate Notation

The inner plate represents the variables associated with each of N_i words in document i : z_{ij} is the topic for the j^{th} word in document i , and w_{ij} is the actual word used.

The circle representing the individual words is shaded, indicating that each w_{ij} is observable.

The other circles are empty, indicating that the other variables are latent variables.

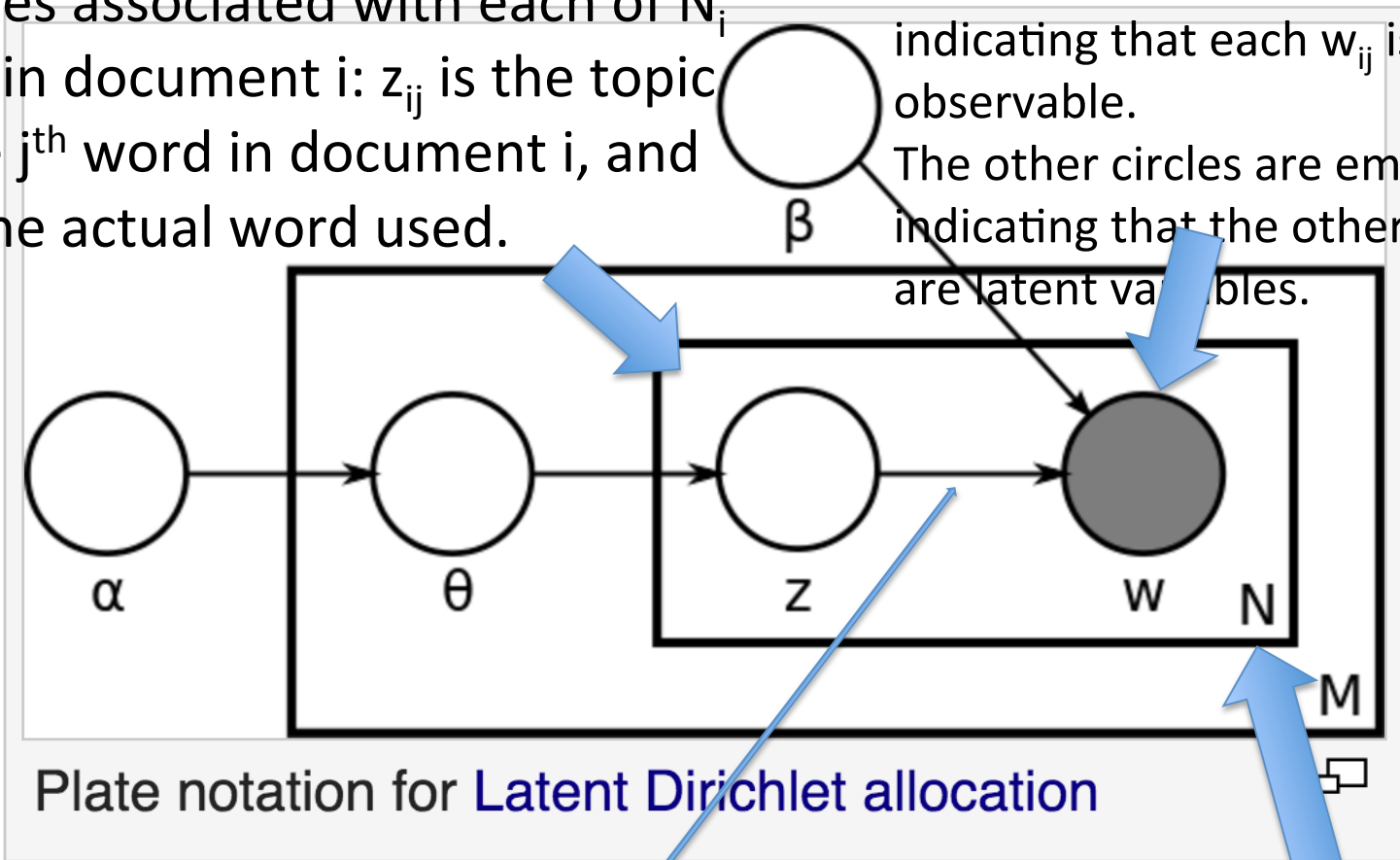


Plate notation for **Latent Dirichlet allocation**

The directed edges between variables indicate dependencies between the variables: for example, each w_{ij} depends z_{ij} and β .

The N represents the repetition of the variables in the inner plate N_i times, once for each word in document i .

Suppose you have the following set of sentences:

- I like to eat broccoli and bananas.
- I ate a banana and spinach smoothie for breakfast.
- Chinchillas and kittens are cute.
- My sister adopted a kitten yesterday.
- Look at this cute hamster munching on a piece of broccoli.

What is latent Dirichlet allocation? It's a way of automatically discovering topics that these sentences contain. For example, given these sentences and asked for 2 topics, LDA might produce something like

- Sentences 1 and 2: 100% Topic A
- Sentences 3 and 4: 100% Topic B
- Sentence 5: 60% Topic A, 40% Topic B
- Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, ... (at which point, you could interpret topic A to be about food)
- Topic B: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, ... (at which point, you could interpret topic B to be about cute animals)

How does LDA perform this discovery?

LDA Model

LDA represents documents as mixtures of topics that spit out words with certain probabilities.

It assumes that documents are produced in the following fashion: when writing each document, you

- Decide on the number of words N the document will have (say, according to a Poisson distribution).
- Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics). *For example, assuming that we have the two food and cute animal topics above, you might choose the document to consist of $1/3$ food and $2/3$ cute animals.*
- Generate each word w_i in the document by:
 - First picking a topic (according to the multinomial distribution that you sampled above; for example, you might pick the food topic with $1/3$ probability and the cute animals topic with $2/3$ probability).
 - Using the topic to generate the word itself (according to the topic's multinomial distribution). For example, if we selected the food topic, we might generate the word “broccoli” with 30% probability, “bananas” with 15% probability, and so on.

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

Example

Let's make an example. According to the above process, when generating some particular document D , you might

- Pick 5 to be the number of words in D .
- Decide that D will be $1/2$ about food and $1/2$ about cute animals.
- Pick the first word to come from the food topic, which then gives you the word “broccoli”.
- Pick the second word to come from the cute animals topic, which gives you “panda”.
- Pick the third word to come from the cute animals topic, giving you “adorable”.
- Pick the fourth word to come from the food topic, giving you “cherries”.
- Pick the fifth word to come from the food topic, giving you “eating”.

So the document generated under the LDA model will be “broccoli panda adorable cherries eating” (*note that LDA is a bag-of-words model*).

Learning

So now suppose you have a set of documents. You've chosen some fixed number of K topics to discover, and want to use LDA to learn the topic representation of each document and the words associated to each topic. How do you do this? One way (known as collapsed Gibbs sampling) is the following:

- Go through each document, and randomly assign each word in the document to one of the K topics.
- Notice that this random assignment already gives you both topic representations of all the documents and word distributions of all the topics (albeit not very good ones).
- So to improve on them, for each document d ...

Learning

So now suppose you have a set of documents. You've chosen some fixed number of K topics to discover, and want to use LDA to learn the topic representation of each document and the words associated to each topic. How do you do this? One way (known as collapsed Gibbs sampling) is the following:

- Go through each document, and randomly assign each word in the document to one of the K topics.
 - Notice that this random assignment already gives you both topic representations of all the documents and word distributions of all the topics (albeit not very good ones).
 - So to improve on them, for each document d ...
- Go through each word w in d ...
- And for each topic t , compute two things:
- 1) $p(\text{topic } t \mid \text{document } d)$ = the proportion of words in document d that are currently assigned to topic t , and
 - 2) $p(\text{word } w \mid \text{topic } t)$ = the proportion of assignments to topic t over all documents that come from this word w .
- Reassign w a new topic, where we choose topic t with probability $p(\text{topic } t \mid \text{document } d) * p(\text{word } w \mid \text{topic } t)$ (according to our generative model, this is essentially the probability that topic t generated word w , so it makes sense that we resample the current word's topic with this probability)
- In other words, in this step, we're assuming that all topic assignments except for the current word in question are correct, and then updating the assignment of the current word using our model of how documents are generated.
- After repeating the previous step a large number of times, you'll eventually reach a roughly steady state where your assignments are pretty good. So use these assignments to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).