DATA MINING LECTURE 11

Classification

Naïve Bayes

Graphs And Centrality

NAÏVE BAYES CLASSIFIER

Bayes Classifier

- A probabilistic framework for solving classification problems
- **A, C** random variables
- Joint probability: **Pr(A=a,C=c)**
- Conditional probability: **Pr(C=c | A=a)**
- Relationship between joint and conditional probability distributions

$$\Pr(C, A) = \Pr(C \mid A) \times \Pr(A) = \Pr(A \mid C) \times \Pr(C)$$

- **Bayes Theorem**:

$$P(C \mid A) = \frac{P(A \mid C) P(C)}{P(A)}$$

Bayesian Classifiers

- Consider each attribute and class label as random variables

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Evade C
Event space: {Yes, No}
P(C) = (0.3,0.7}

Refund A_1
Event space: {Yes, No}
$P(A_1)$ = (0.3,0.7)

Martial Status A_2
Event space: {Single, Married, Divorced}
$P(A_2)$ = (0.4,0.4,0.2)

Taxable Income A_3
Event space: R
$P(A_3) \sim \text{Normal}(\mu,\sigma)$

Bayesian Classifiers

- Given a record X over attributes $(A_1, A_2, \ldots, A_n)$
  - E.g., X = (‘Yes’, ‘Single’, 125K)
- The goal is to predict class C
  - Specifically, we want to find the value c of C that maximizes P(C=c| X)
- Can we estimate P(C| X) directly from data?
  - This means that we estimate the probability for all possible values of the class variable.

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$\boxed{P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$
- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n | C) = P(A_1|C)\, P(A_2|C) \cdots P(A_n|C)$

  - We can estimate $P(A_i|\ C)$ for all values of A_i and C.

  - New point X is classified to class c if

$$P(C = c|X) = P(C = c) \prod_i P(A_i|c)$$

  is maximal over all possible values of C.

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class Prior Probability:

$$P(C = c) = \frac{N_c}{N}$$

e.g., P(C = No) = 7/10,
P(C = Yes) = 3/10

- For discrete attributes:

$$P(A_i = a|C = c) = \frac{N_{a,c}}{N_c}$$

where $N_{a,c}$ is number of instances having attribute $A_i = a$ and belongs to class c

  - Examples:

P(Status=Married|No) = 4/7
P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: (A < v) or (A > v)
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean μ and standard deviation σ)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i = a \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(a-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (a_i, c_i) pair
- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No) $\times$ P(Married| Class=No) $\times$ P(Income=120K| Class=No)
  = 4/7 $\times$ 4/7 $\times$ 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes) $\times$ P(Married| Class=Yes) $\times$ P(Income=120K| Class=Yes)
  = 1 $\times$ 0 $\times$ 1.2 $\times$ 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)
=> Class = No

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No) $\times$ P(Married| Class=No) $\times$ P(Income=120K| Class=No) = 4/7 $\times$ 4/7 $\times$ 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes) $\times$ P(Married| Class=Yes) $\times$ P(Income=120K| Class=Yes) = 1 $\times$ 0 $\times$ 1.2 $\times$ 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original}: P(A_i = a \mid C = c) = \frac{N_{ac}}{N_c}$$

$$\text{Laplace}: P(A_i = a \mid C = c) = \frac{N_{ac} + 1}{N_c + N_i}$$

$$\text{m - estimate}: P(A_i = a \mid C = c) = \frac{N_{ac} + mp}{N_c + m}$$

N_i: number of attribute values for attribute A_i

p: prior probability

m: parameter

Implementation details

- Computing the conditional probabilities involves multiplication of many very small numbers
  - Numbers get very close to zero, and there is a danger of numeric instability
- We can deal with this by computing the logarithm of the conditional probability

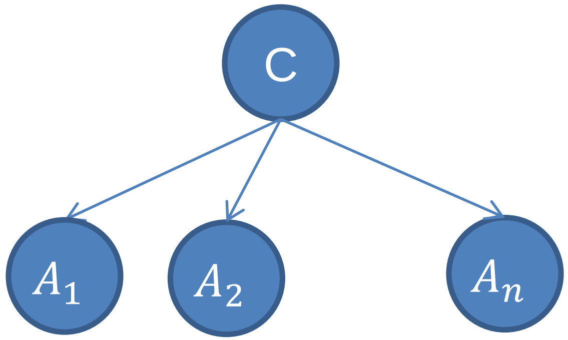
$$\log P(C|A) \sim \log P(A|C) + \log P(A)$$
$$= \sum_i \log(A_i|C) + \log P(A)$$

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)
- Naïve Bayes can produce a probability estimate, but it is usually a very biased one
  - Logistic Regression is better for obtaining probabilities.

Generative vs Discriminative models

- Naïve Bayes is a type of a generative model
  - Generative process:
    - First pick the category of the record
    - Then given the category, generate the attribute values from the distribution of the category
    - Conditional independence given C



- We use the training data to learn the distribution of the values in a class

Generative vs Discriminative models

- Logistic Regression and SVM are discriminative models
  - The goal is to find the boundary that discriminates between the two classes from the training data

- In order to classify the language of a document, you can
  - Either learn the two languages and find which is more likely to have generated the words you see
  - Or learn what differentiates the two languages.

SUPERVISED LEARNING

Learning

- Supervised Learning: learn a model from the data using labeled data.
  - Classification and Regression are the prototypical examples of supervised learning tasks. Other are possible (e.g., ranking)
- Unsupervised Learning: learn a model – extract structure from unlabeled data.
  - Clustering and Association Rules are prototypical examples of unsupervised learning tasks.
- Semi-supervised Learning: learn a model for the data using both labeled and unlabeled data.

Supervised Learning Steps

- Model the problem
  - What is you are trying to predict? What kind of optimization function do you need? Do you need classes or probabilities?
- Extract Features
  - How do you find the right features that help to discriminate between the classes?
- Obtain training data
  - Obtain a collection of labeled data. Make sure it is large enough, accurate and representative. Ensure that classes are well represented.
- Decide on the technique
  - What is the right technique for your problem?
- Apply in practice
  - Can the model be trained for very large data? How do you test how you do in practice? How do you improve?

Modeling the problem

- Sometimes it is not obvious. Consider the following three problems
  - Detecting if an email is spam
  - Categorizing the queries in a search engine
  - Ranking the results of a web search

Feature extraction

- Feature extraction, or feature engineering is the most tedious but also the most important step
  - How do you separate the players of the Greek national team from those of the Swedish national team?
- One line of thought: throw features to the classifier and the classifier will figure out which ones are important
  - More features, means that you need more training data
- Another line of thought: select carefully the features using various functions and techniques
  - Computationally intensive

Training data

- An overlooked problem: How do you get labeled data for training your model?
  - E.g., how do you get training data for ranking?
- Usually requires a lot of manual effort and domain expertise and carefully planned labeling
  - Results are not always of high quality (lack of expertise)
  - And they are not sufficient (low coverage of the space)
- Recent trends:
  - Find a source that generates the labeled data for you.
  - Crowd-sourcing techniques

Dealing with small amount of labeled data

- Semi-supervised techniques have been developed for this purpose.

- Self-training: Train a classifier on the data, and then feed back the high-confidence output of the classifier as input

- Co-training: train two “independent” classifiers and feed the output of one classifier as input to the other.

- Regularization: Treat learning as an optimization problem where you define relationships between the objects you want to classify, and you exploit these relationships
  - Example: Image restoration

Technique

- The choice of technique depends on the problem requirements (do we need a probability estimate?) and the problem specifics (does independence assumption hold? Do we think classes are linearly separable?)
- For many cases finding the right technique may be trial and error
- For many cases the exact technique does not matter.

Big Data Trumps Better Algorithms

- If you have enough data then the algorithms are not so important

- The web has made this possible.
  - Especially for text-related tasks
  - Search engine uses the collective human intelligence

http://www.youtube.com/watch?v=nU8DcBF-qo4

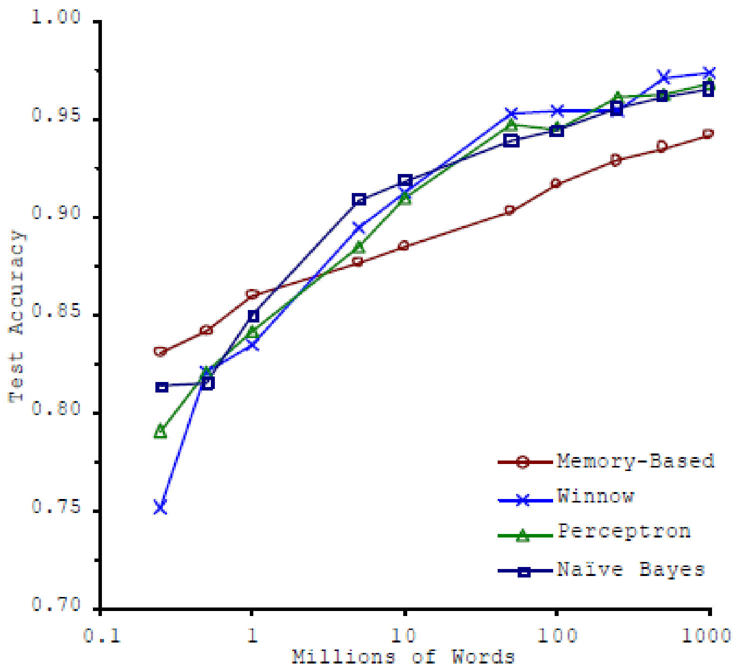


Figure 1. Learning Curves for Confusion Set Disambiguation

Apply-Test

- How do you scale to very large datasets?
  - Distributed computing – map-reduce implementations of machine learning algorithms (Mahut, over Hadoop)

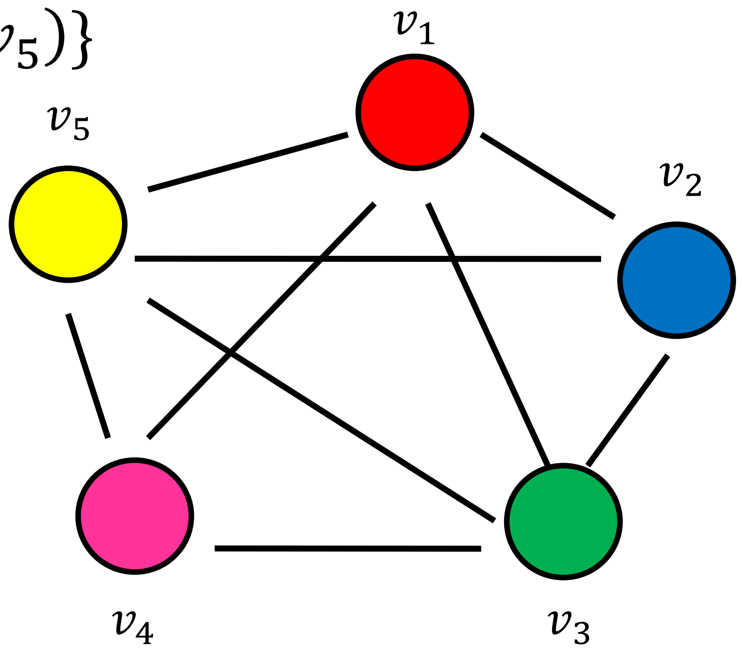
- How do you test something that is running online?
  - You cannot get labeled data in this case
  - A/B testing

- How do you deal with changes in data?
  - Active learning

GRAPHS AND LINK ANALYSIS RANKING

Graphs - Basics

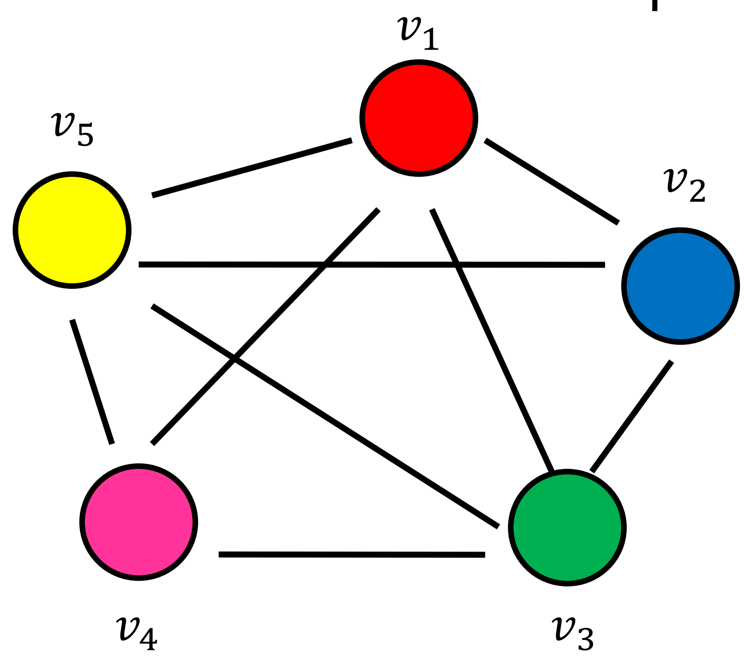
- A graph is a powerful abstraction for modeling entities and their pairwise relationships.
- G = (V,E)
  - Set of nodes $V = \{v_1, \ldots, v_5\}$
  - Set of edges $E = \{(v_1, v_2), \ldots (v_4, v_5)\}$
- Examples:
  - Social network
  - Twitter Followers
  - Web
  - Collaboration graphs



Undirected Graphs

- Undirected Graph: The edges are undirected pairs – they can be traversed in any direction.
- Degree of node: Number of edges incident on the node
- Path: A sequence of edges from one node to another
  - We say that the node is reachable
- Connected Component: A set of nodes such that there is a path between any two nodes in the set

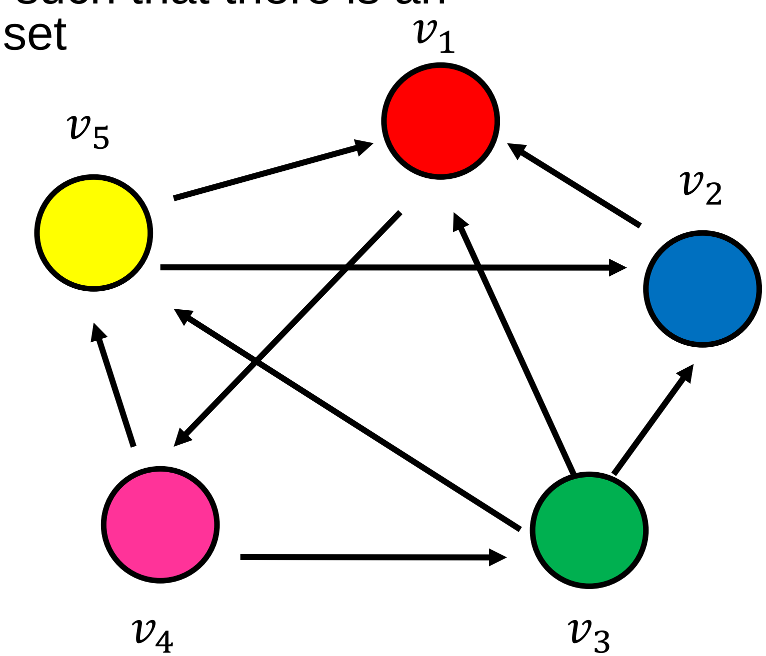
$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$



Directed Graphs

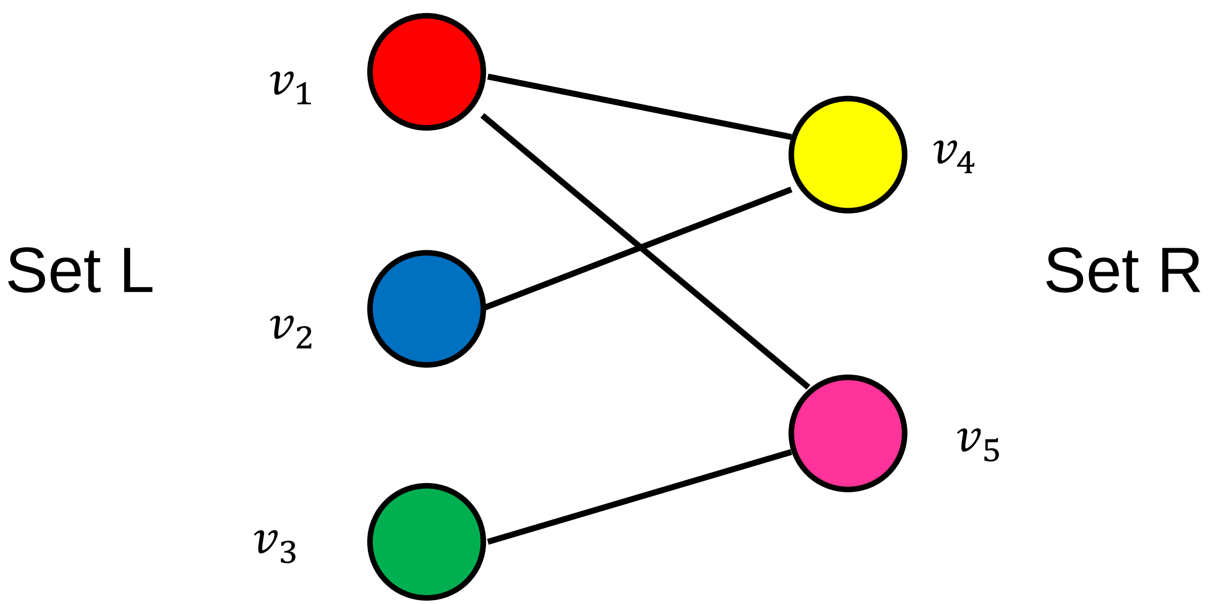
- Directed Graph: The edges are ordered pairs – they can be traversed in the direction from first to second.
- In-degree and Out-degree of a node.
- Path: A sequence of directed edges from one node to another
  - We say that the node is reachable
- Strongly Connected Component: A set of nodes such that there is a directed path between any two nodes in the set
- Weakly Connected Component: A set of nodes such that there is an undirected path between any two nodes in the set

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



Bipartite Graph

- A graph where the vertex set V is partitioned into two sets V = {L,R}, of size greater than one, such that there is no edge within each set.



Importance problem

- What are the most important nodes in the graph?
  - What are the most authoritative pages on the web
  - Who are the important users in Facebook?
  - What are the most influential Twitter accounts?

Why is this important?

- When you make a query “microsoft” to Google why do you get the home page of Microsoft as the first result?

Link Analysis

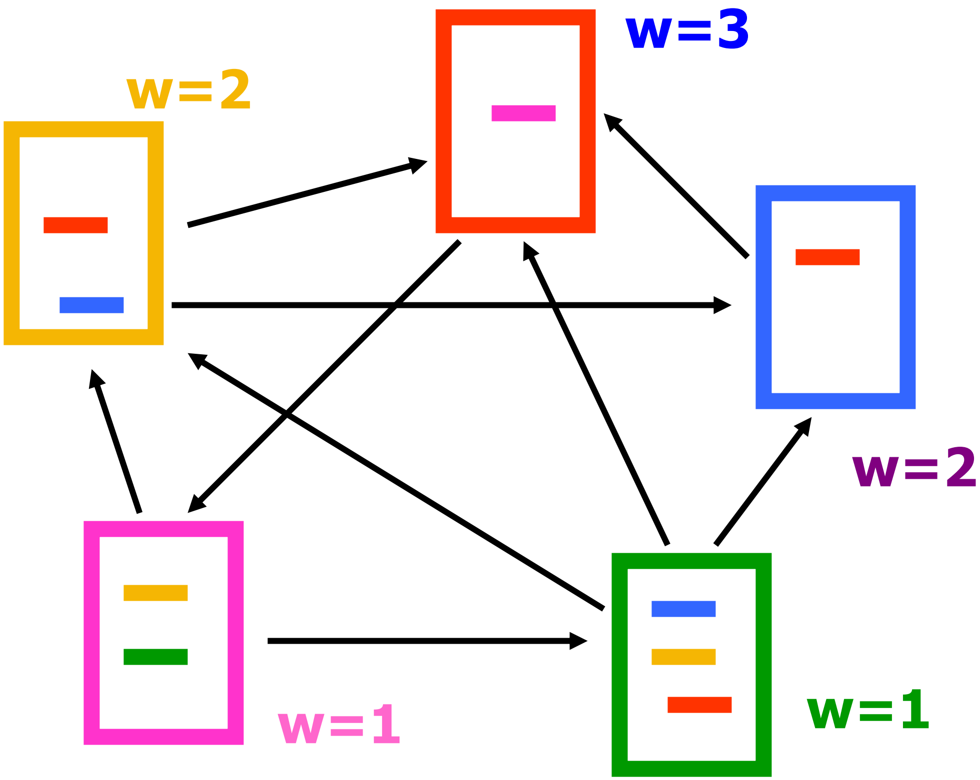
- First generation search engines
  - view documents as flat text files
  - could not cope with size, spamming, user needs
- Second generation search engines
  - Ranking becomes critical
  - use of Web specific data: Link Analysis
  - shift from relevance to authoritativeness
  - a success story for the network analysis

Link Analysis: Intuition

- A link from page p to page q denotes endorsement
  - page p considers page q an authority on a subject
  - use the graph of recommendations
  - assign an authority value to every page

Popularity: InDegree algorithm

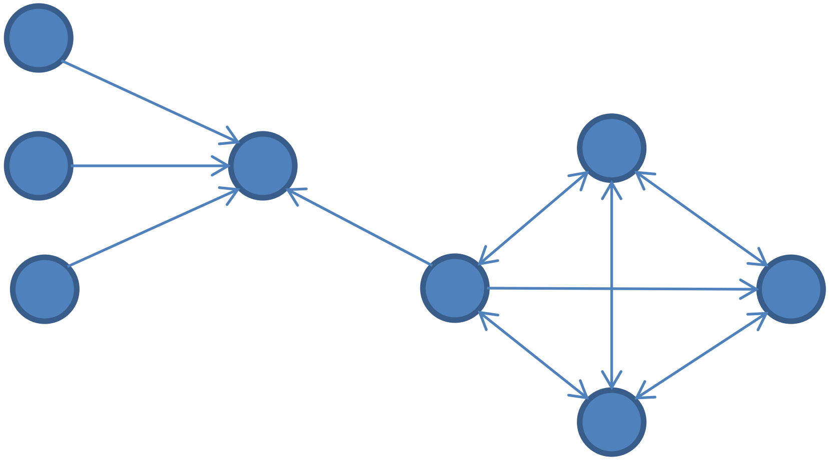
- Rank pages according to the popularity of incoming edges



1. Red Page
2. Yellow Page
3. Blue Page
4. Purple Page
5. Green Page

Popularity

- Could you think of the case where this could be a problem?



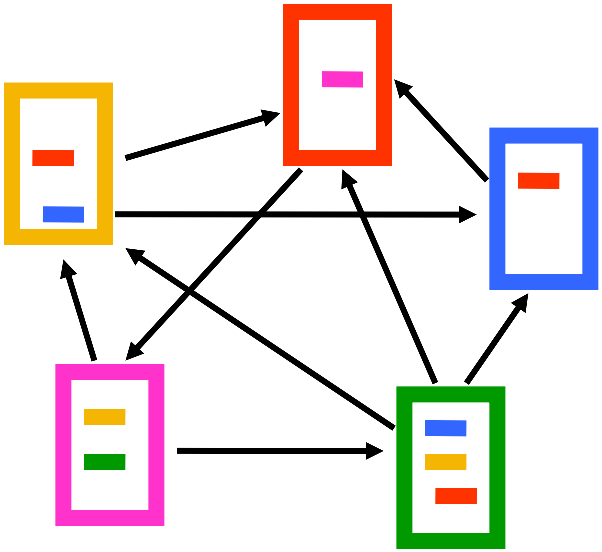
- It is not important only how many link to you, but how important are the people that link to you.

PageRank algorithm [BP98]

- Good authorities should be pointed by good authorities
  - The value of a page is the value of the people that link to you

- How do we implement that?
  - Each page has a value.
  - Proceed in iterations,
    - in each iteration every page distributes the value to the neighbors
  - Continue until there is convergence.

$$PR(p) = \sum_{q \to p} \frac{PR(q)}{|F(q)|}$$



1. Red Page
2. Purple Page
3. Yellow Page
4. Blue Page
5. Green Page

Random Walks on Graphs

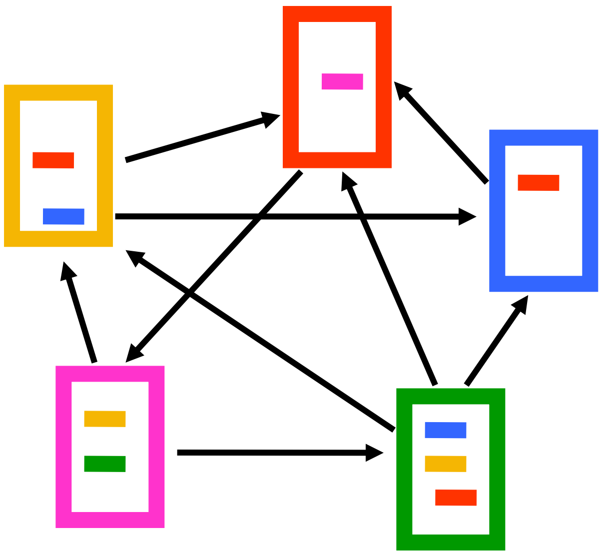
- What we described is equivalent to a random walk on the graph

- Random walk:
  - Pick a node uniformly at random
  - Pick one of the outgoing edges uniformly at random
  - Repeat.

- Question:
  - What is the probability that after N steps you will be at node x? Or, after N steps, what is the fraction of times times have you visited node x?
    - The answer is the same for these two questions
  - When $N \rightarrow \infty$ this number converges to a single value regardless of the starting point!

PageRank algorithm [BP98]

- Random walk on the web graph (the Random Surfer model)
  - pick a page at random
  - with probability $1-\alpha$ jump to a random page
  - with probability α follow a random outgoing link
- Rank according to the stationary distribution
- $$PR(p) = \alpha \sum_{q \to p} \frac{PR(q)}{|F(q)|} + (1-\alpha)\frac{1}{n}$$



1. Red Page
2. Purple Page
3. Yellow Page
4. Blue Page
5. Green Page

Markov chains

- A Markov chain describes a discrete time stochastic process over a set of states

$$S = \{s_1, s_2, \dots s_n\}$$

according to a transition probability matrix

$$P = \{P_{ij}\}$$

  - P_{ij} = probability of moving to state j when at state i
    - $\sum_j P_{ij} = 1$ (stochastic matrix)

- Memorylessness property: The next state of the chain depends only at the current state and not on the past of the process (first order MC)
  - higher order MCs are also possible

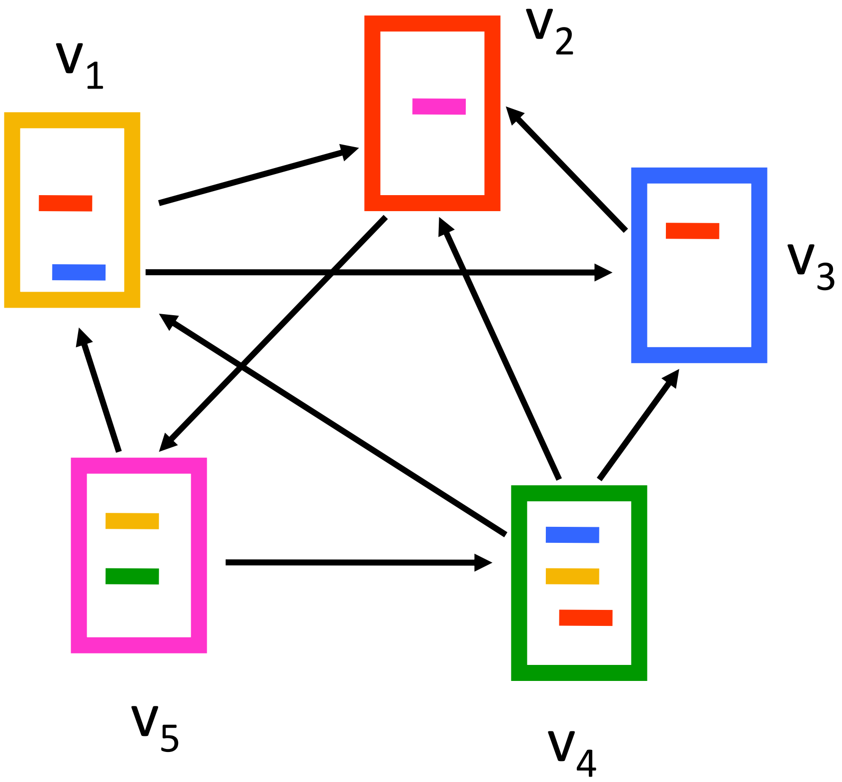
Random walks

- Random walks on graphs correspond to Markov Chains
  - The set of states S is the set of nodes of the graph G
  - The transition probability matrix is the probability that we follow an edge from one node to another

An example

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

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$



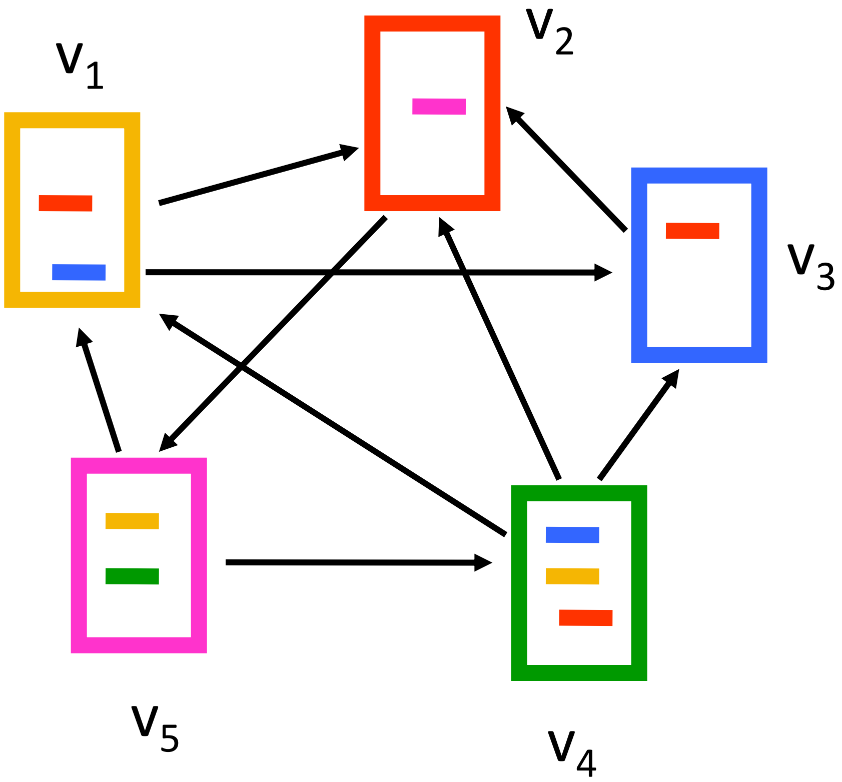
State probability vector

- The vector $q^t = (q^t_1, q^t_2, \ldots, q^t_n)$ that stores the probability of being at state i at time t
  - q^0_i = the probability of starting from state i

$$q^t = q^{t-1} P$$

An example

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



$q^{t+1}_1 = 1/3\ q^t_4 + 1/2\ q^t_5$

$q^{t+1}_2 = 1/2\ q^t_1 + q^t_3 + 1/3\ q^t_4$

$q^{t+1}_3 = 1/2\ q^t_1 + 1/3\ q^t_4$

$q^{t+1}_4 = 1/2\ q^t_5$

$q^{t+1}_5 = q^t_2$

Stationary distribution

- A stationary distribution for a MC with transition matrix P, is a probability distribution π, such that $\pi = \pi P$

- A MC has a unique stationary distribution if
  - it is irreducible
    - the underlying graph is strongly connected
  - it is aperiodic
    - for random walks, the underlying graph is not bipartite
- The probability π_i is the fraction of times that we visited state i as $t \to \infty$
- The stationary distribution is an eigenvector of matrix P
  - the principal left eigenvector of P – stochastic matrices have maximum eigenvalue 1

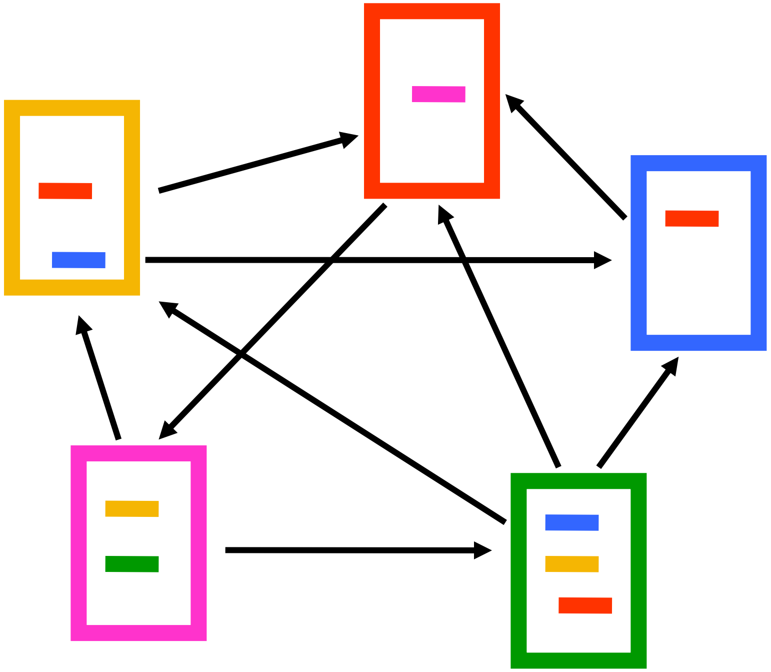
Computing the stationary distribution

- The Power Method
  - Initialize to some distribution q^0
  - Iteratively compute $q^t = q^{t-1}P$
  - After enough iterations $q^t \approx \pi$
  - Power method because it computes $q^t = q^0P^t$
- Why does it converge?
  - follows from the fact that any vector can be written as a linear combination of the eigenvectors
    - $q^0 = v_1 + c_2v_2 + \ldots c_nv_n$
- Rate of convergence
  - determined by λ_2^t

The PageRank random walk

- Vanilla random walk
  - make the adjacency matrix stochastic and run a random walk

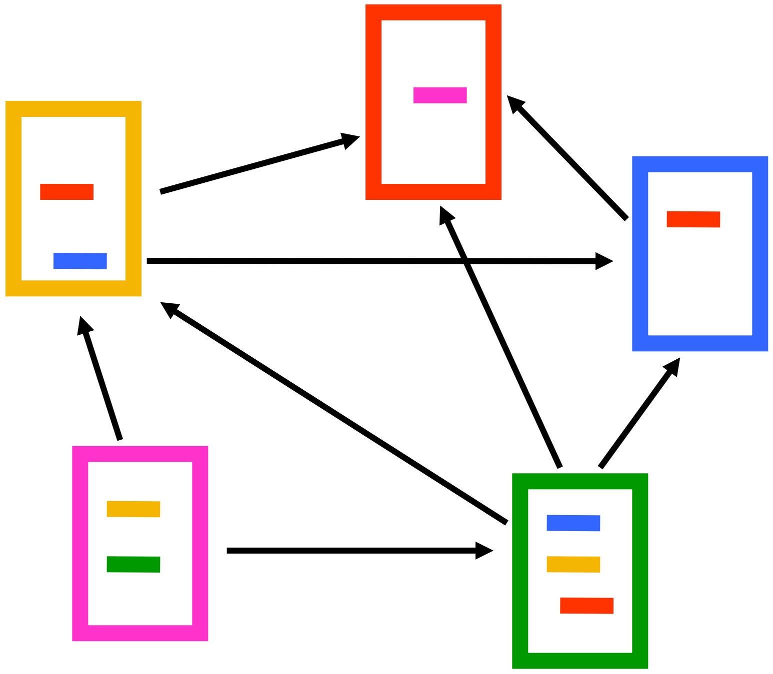
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

- What about sink nodes?
  - what happens when the random walk moves to a node without any outgoing inks?

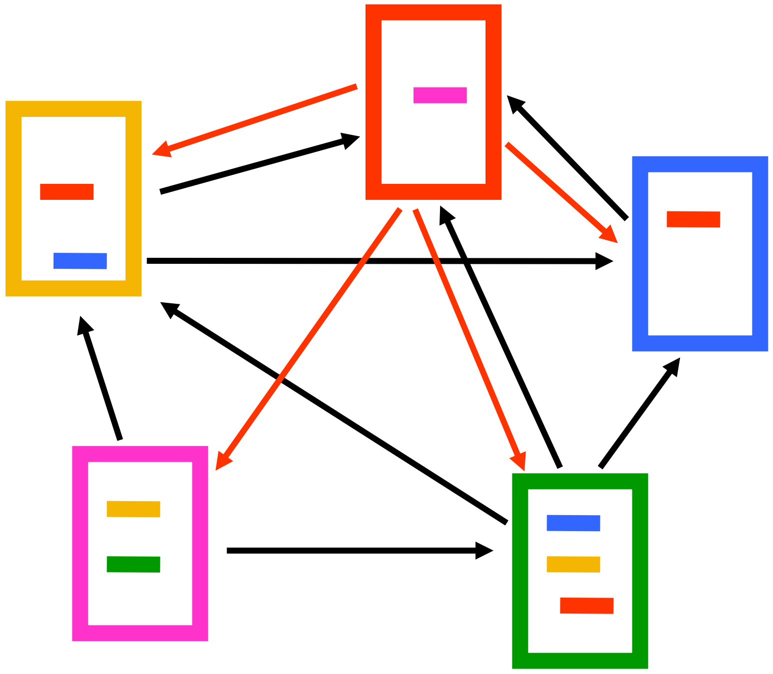
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

- Replace these row vectors with a vector v
  - typically, the uniform vector

$$P' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



$P' = P + dv^T$

$$d = \begin{cases} 1 & \text{if } i \text{ is sink} \\ 0 & \text{otherwise} \end{cases}$$

The PageRank random walk

- How do we guarantee irreducibility?
  - add a random jump to vector v with prob α
    - typically, to a uniform vector

$$P'' = \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} + (1-\alpha) \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}$$

$P'' = \alpha P' + (1-\alpha)uv^T$, where u is the vector of all 1s

Effects of random jump

- Guarantees irreducibility
- Motivated by the concept of random surfer
- Offers additional flexibility
  - personalization
  - anti-spam
- Controls the rate of convergence
  - the second eigenvalue of matrix P'' is α

A PageRank algorithm

- Performing vanilla power method is now too expensive – the matrix is not sparse

$q^0 = v$

$t = 1$

repeat

$\quad q^t = (P'')^T q^{t-1}$

$\quad \delta = \|q^t - q^{t-1}\|$

$\quad t = t + 1$

until $\delta < \varepsilon$

Efficient computation of $y = (P'')^T x$

$$y = \alpha P'^T x$$

$$\beta = \|x\|_1 - \|y\|_1$$

$$y = y + \beta v$$

Random walks on undirected graphs

- In the stationary distribution of a random walk on an undirected graph, the probability of being at node i is proportional to the (weighted) degree of the vertex

- Random walks on undirected graphs are not so "interesting"