COMP9727: Recommender Systems

Lecture 5: Social Recommender Systems

Wayne Wobcke

e-mail:w.wobcke@unsw.edu.au

This Lecture

Recommendation using social media data, often on social media

- Social Filtering, using real social networks (not “neighbourhoods”)
  - Facebook, Instagram, YouTube, Twitter/X, Reddit, Quora, ...
  - “Web 2.0” = “User Generated Content” = “Wisdom of the Crowd”
  - Use “likes”, retweets, upvotes, tags, comments, reviews, ...
- Applications
  - Feeds, News, Blogs, Photos, Videos, Restaurants, Q&A, ...
  - Tag-aware recommendation (from a “folksonomy”)
- Research Problems
  - Networks: “Trust” (Reputation) Propagation (c.f. Amazon)
  - NLP: Sentiment Analysis, Stance Detection

YouTube Recommender (2010)

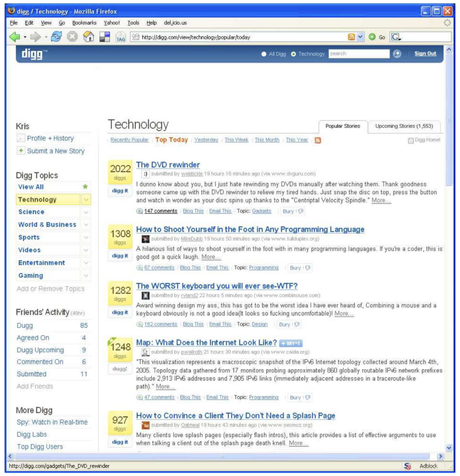
1. Use co-visitation (association rule lift) in last 24 hours (offline)
   - Seed video gives up to N related videos with score > threshold
2. Get seed set = watched/liked videos from user activity
3. Generate candidate set by applying co-visitation
4. Expand a few more times to get more “diverse” candidates
5. Rank candidates using linear function for this user
   - Quality: views, ratings, comments, favourites, shares, upload time
   - Specificity: views, watch time of seed video in user history
6. Diversity: Remove videos in candidate set too similar to each other
   - Same seed video, same uploader, same content, etc.

Digg News Aggregator

- Can “follow” other Digg users: follower/followee (like Twitter)
  - MySpace (music), Flickr (photos), del.icio.us (web bookmarks)
  - Quiz Question: How many of these still exist?
- Has same “front page” for all users
- Rank users by number of stories on front page
- Rank stories by number of diggs and rate of diggs, etc.
  - Open to manipulation by coordinated teams (extremists)
- Allow users to view articles based on friends’ activity
  - In turn boosting those articles to the front page ...

Similar to Reddit: feed not personalized

Digg Front Page



Trust Networks

- How much do you “trust” your “friends”?
- Network with arrow $A \rightarrow B$ as A trusts B (to some degree)
- Trust propagation
  - e.g. transitivity: $A \rightarrow B$ and $B \rightarrow C$ implies $A \rightarrow C$
- Questions
  - What is the role of social media “influencers”?
  - Who influences the influencers?
  - Lazarsfeld & Katz (1955) *Personal Influence*: opinion leaders

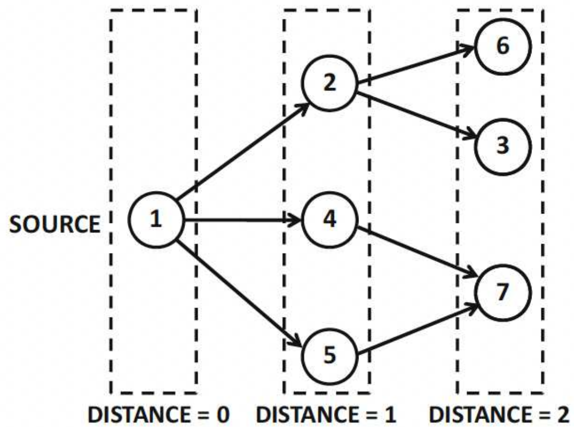
Quiz Question: What is the #1 predictor of whether you own an iPhone?

Quiz Question: Is influence domain specific or general?

Trust Propagation

- Suppose values are between 0 and 1
- Multiply values along a path such as $A \rightarrow B \rightarrow C$
  - Or discount $B \rightarrow C$ by some decay factor β
- Aggregate values across multiple paths
  - Such as max, min, weighted average, etc., with or without decay
  - With multiple paths between two nodes, choose all shortest ones?
- Application
  - Use trust rather than similarity in user-based CF
  - Determine neighbourhoods using trust in item-based CF
  - Can have a different trust network for each topic (domain experts!)

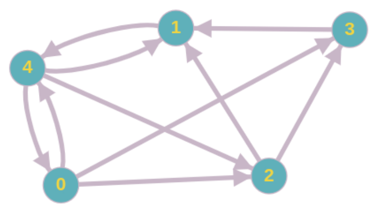
MoleTrust



Google PageRank

- Importance “flows” from one node to another
- Importance of a node derived from in-coming links
- $rank(j) = \sum_i \frac{rank(i)}{d_i}$ over nodes $i \to j$ where d_i is out-degree of i
- In matrix form, solution is π such that $P\pi = \pi$ (eigenvector!)
- Solve by repeated powers $P\pi, P^2\pi, \cdots$ starting with random π
- Add damping factor β to avoid “dead ends”

PageRank Example



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

$$R = \begin{pmatrix} \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{2}\beta + \frac{1-\beta}{5} & \beta + \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{2}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} \end{pmatrix}$$

e.g. $v_1 = \frac{1}{2}v_2 + v_3 + \frac{1}{3}v_4$

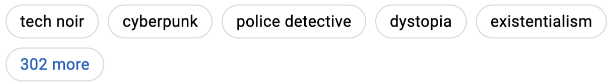
Tag-Aware Recommendation

Folksonomy = Collection of tags (over a set of items)

- Recommend items to user based on tagging behaviour
- Recommend items to user for a given tag
- Recommend tags to user based on searching
- Recommend users to users based on similar tags used
- Recommend tags to user for a given item

Use content-based recommendation and/or CF – for which scenarios?

Problem: Folksonomies are large, complex and messy

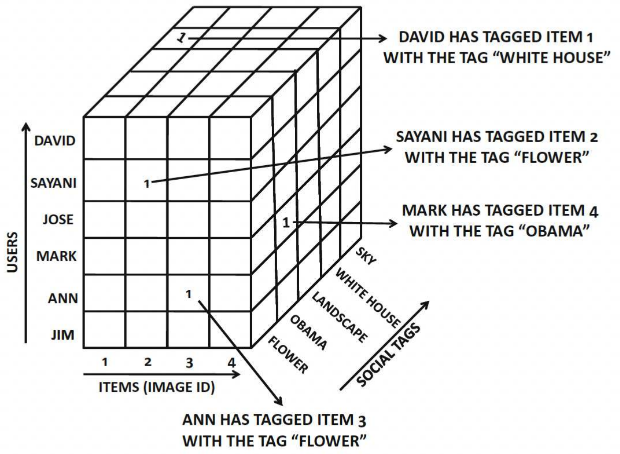


Tag-Aware Recommendation Example

Content-based recommendation applied to movie ratings

- Movies are documents – tag clouds
- User profiles are tag clouds for each rating value
- Match user profiles with tf-idf vectors for movies
- Can use weighted sum by similarity over rating values
- Can also cluster tags using similarity of tf-idf vectors
  - ... to improve recall and maybe address cold start problem

Tag Cube



Tag-Aware Recommendation Examples

CF applied to movie ratings and tags (c.f. hybrid methods)

- With limited data ... (here 0/1 matrix)
  - Add tags into the ratings matrix as if they are items (pseudo-items)
  - Add tags into the ratings matrix as if they are users (pseudo-users)
  - Use weighted sum of user-based and item-based predictions
  - c.f. feature augmentation in hybrid recommender systems
- With more data ...
  - Apply "query expansion" by tag clusters to user's tags
  - Add expanded tags into the ratings matrix for each movie
  - Augment user similarity with tf-idf similarity on tags
  - c.f. meta-features in hybrid recommender systems

Sentiment Analysis

Determine sentiment of language: positive/neutral/negative

- Hand-curated lexicons (accurate but incomplete)
- Topic modelling (more complete)
  - Topics are probability distributions over words
  - Need good "seed words" to define topics
- Neural networks with Word2Vec and Multi-Layer Perceptron
  - Word2Vec is "self supervised"; training data for sentiment
  - Example in tutorials uses only positive/negative (easier!)

Artificial Neural Networks

(Artificial) Neural Networks are made up of nodes which have

- Input edges, each with some weight
- Output edges (with weights)
- An activation level (a function of the inputs)

Weights can be positive or negative and may change over time (learning)

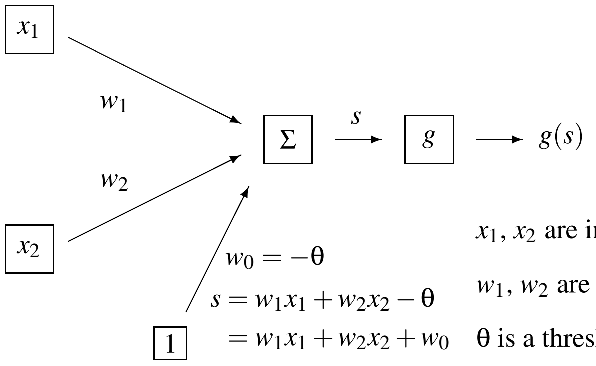
The input function is the weighted sum of the activation levels of inputs

The activation level is a non-linear transfer function g of this input

$$\text{activation}_i = g(s_i) = g(\sum_j w_{ij} x_j)$$

Some nodes are inputs (sensing), some are outputs (action)

McCulloch & Pitts Model of a Single Neuron



x_1, x_2 are inputs

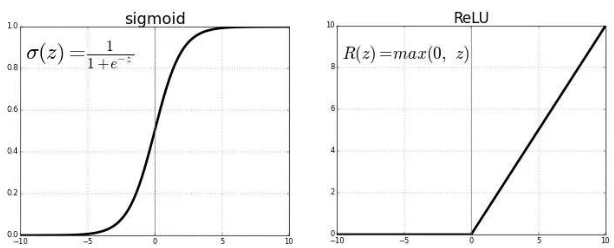
w_1, w_2 are synaptic weights

θ is a threshold

w_0 is a **bias** weight

g is transfer function

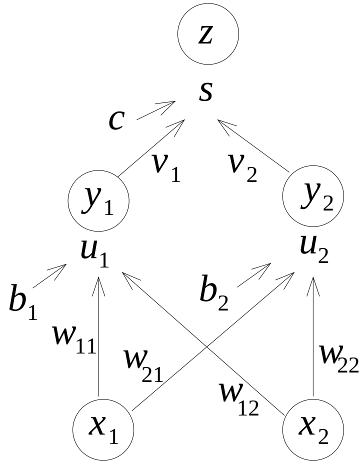
Transfer Functions



Note: if $g(z) = \dfrac{1}{1+e^{-z}}$ $\quad g'(z) = g(z)(1-g(z))$

$$g(z) = \max(z, 0) \qquad g'(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Simple Neural Network



$$\begin{aligned} u_1 &= b_1 + w_{11}x_1 + w_{12}x_2 \\ y_1 &= g(u_1) \\ s &= c + v_1y_1 + v_2y_2 \\ z &= g(s) \\ E &= \frac{1}{2}\sum(z-t)^2 \end{aligned}$$

Gradient Descent

Define an **error** (or "loss") function E as (half) the sum over all input patterns of the square of the difference between actual output and desired output

$$E = \frac{1}{2}\sum(z-t)^2$$

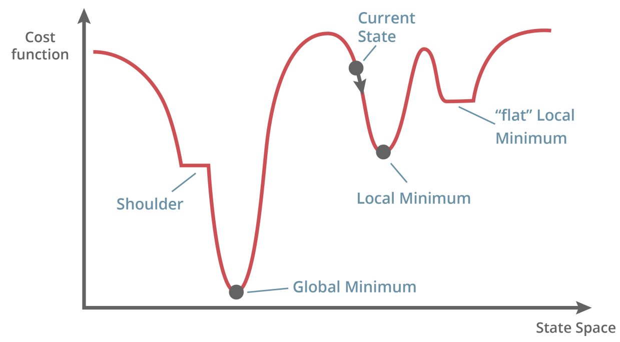
The aim is to find a set of weights for which E is very low.

If the functions are smooth, use multi-variate calculus to define how to adjust the weights so error moves in steepest downhill direction

$$w \leftarrow w - \eta\frac{\partial E}{\partial w}$$

Parameter η is called the learning rate

Local Search in Weight Space



Problem: Because of the step function, the landscape will not be smooth but will instead consist almost entirely of flat local regions and "shoulders", with occasional discontinuous jumps

Chain Rule

If

$$y = y(u)$$

$$u = u(x)$$

Then

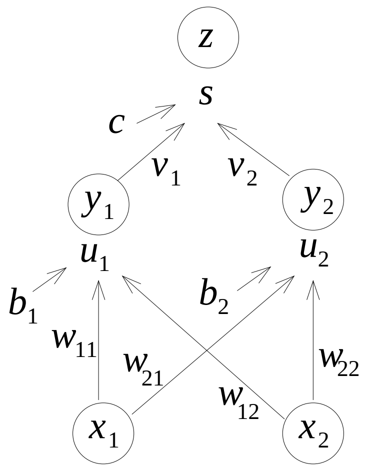
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$

This principle can be used to compute the partial derivatives in an efficient and localized manner. The transfer function must be differentiable.

Note: if $z(s) = \dfrac{1}{1+e^{-s}}$ $\quad \dfrac{\partial z}{\partial s} = z(1-z)$

if $z(s) = \tanh(s)$ $\quad \dfrac{\partial z}{\partial s} = 1 - z^2$

Forward Pass



$$\begin{aligned}
u_1 &= b_1 + w_{11}x_1 + w_{12}x_2 \\
y_1 &= g(u_1) \\
s &= c + v_1 y_1 + v_2 y_2 \\
z &= g(s) \\
E &= \frac{1}{2}\sum (z-t)^2
\end{aligned}$$

Backpropagation

Partial Derivatives

$$\begin{aligned}
\frac{\partial E}{\partial z} &= z - t \\
\frac{dz}{ds} &= g'(s) = z(1-z) \\
\frac{\partial s}{\partial y_1} &= v_1 \\
\frac{dy_1}{du_1} &= y_1(1-y_1)
\end{aligned}$$

Useful notation

$$\delta_{\text{out}} = \frac{\partial E}{\partial s} \quad \delta_1 = \frac{\partial E}{\partial u_1} \quad \delta_2 = \frac{\partial E}{\partial u_2}$$

Then

$$\begin{aligned}
\delta_{\text{out}} &= (z-t)\, z\, (1-z) \\
\frac{\partial E}{\partial v_1} &= \delta_{\text{out}}\, y_1 \\
\delta_1 &= \delta_{\text{out}}\, v_1\, y_1\, (1-y_1) \\
\frac{\partial E}{\partial w_{11}} &= \delta_1\, x_1
\end{aligned}$$

Partial derivatives can be calculated efficiently by backpropagating deltas through the network

Cross Entropy

For classification tasks, target t is either 0 or 1, so better to use

$$E = -t\,\log(z) - (1-t)\,\log(1-z)$$

This can be justified mathematically, and works well in practice – especially when negative examples vastly outweigh positive ones.
It also makes the backpropagation computations simpler:

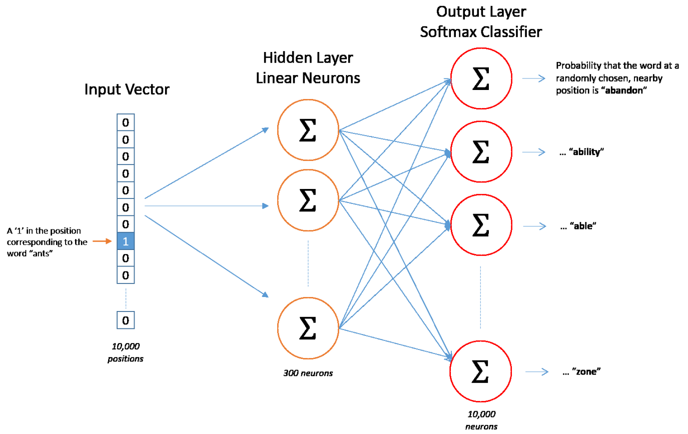
$$\begin{aligned}
\frac{\partial E}{\partial z} &= \frac{z-t}{z(1-z)} \\
\text{If} \quad z &= \frac{1}{1+e^{-s}} \\
\frac{\partial E}{\partial s} &= \frac{\partial E}{\partial z}\frac{\partial z}{\partial s} = z-t
\end{aligned}$$

Word2Vec

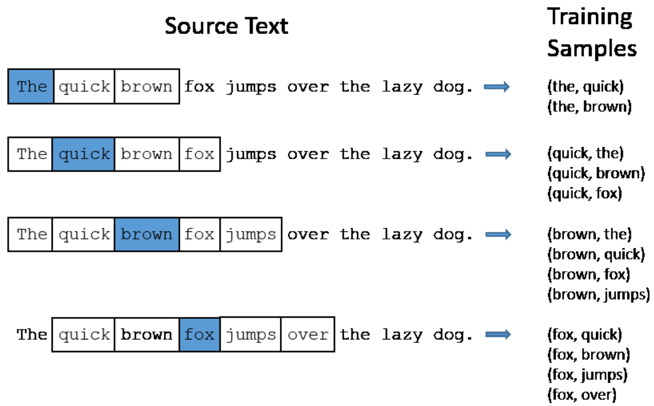
- Word "embeddings" as vectors ($\approx$ 300 dimensions)
- Training: In effect, inputs and outputs are single words
  - CBOW: Context words (words in window) predict word
  - Skip-gram: Word predicts context words (words in window)
  - Add negative training examples
- Embeddings are first layer of the network, ignore second layer
- Words with "similar" contexts have "similar" embeddings
- Alternative uses word and context word vectors (wrong?)
- Can use pretrained model (Google News) or train new model

Somewhat similar to matrix factorization with 300 dimensions

Word2Vec Architecture



Word2Vec Training Data



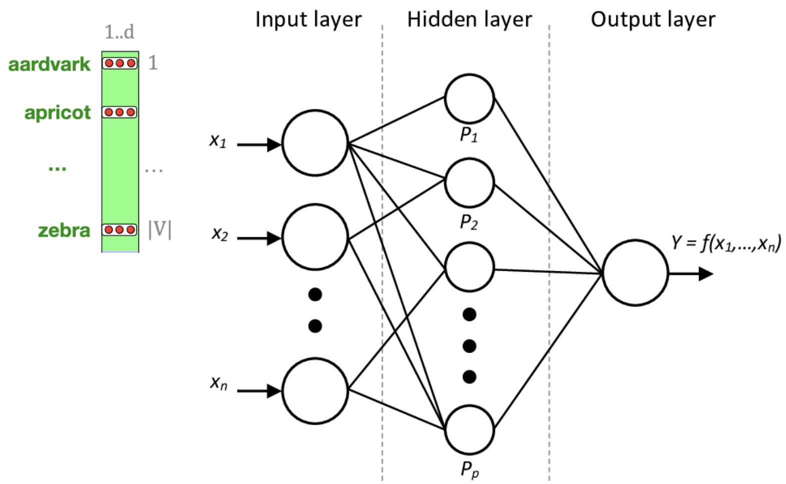
Word2Vec Sentiment Embeddings

What we want to see:

Sentiment Analysis with Word2Vec

- Word Embeddings
  - Word vectors trained on 25,000 movie reviews
  - Concatenate word vectors for each word in the review
- Multi-Layer Perceptron
  - First layer is embedding layer
  - Hidden layer fully connected (128 dimensions)
  - Output layer fully connected (1 neuron) for sentiment $(+/-)$
  - Train through backpropagation
  - 6,400,257 parameters (weights) to learn

Multi-Layer Perceptron (MLP)



Further Research Problems

- Aspect-Based Sentiment Analysis
  - Cameras: zoom, battery, resolution, ease of use, etc.
  - Hotels: location, room, cleanliness, host, value, etc.
  - Useful for more precise recommendations, or critiquing
- Stance Detection
  - Abortion, gun control, animal rights – for or against
  - Useful for political advertising

Summary

- Suitable for social media companies
- Much money to be made from targeted advertising
- Role of "influencers" important
- Research shows using tags improves recommendation
  - ... despite folksonomies being very noisy
- Susceptible to attacks by bots, extremists
  - And echo chambers leading to polarization
- Next lecture social network recommendation