Caltech

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 14:

Embeddings

Past Two Lectures

- Dimensionality Reduction
- Clustering

- Latent Factor Models
  – Learn low-dimensional representation of data

This Lecture

- Embeddings
  - Generalization of Latent-Factor Models
- Warm-up: Locally-Linear Embeddings
- Probabilistic Sequence Embeddings
  - Playlist embeddings
  - Word embeddings

Embedding

- Learn a representation U
  - Each column u corresponds to data point
- Semantics encoded via d(u,u')
  - Distance between points

$$d(u,u') = \|u - u'\|^2$$

  - Similarity between points

$$d(u,u') = u^T u'$$

Generalizes
Latent-Factor Models

Locally Linear Embedding

- Given: $S = \{x_i\}_{i=1}^{N}$

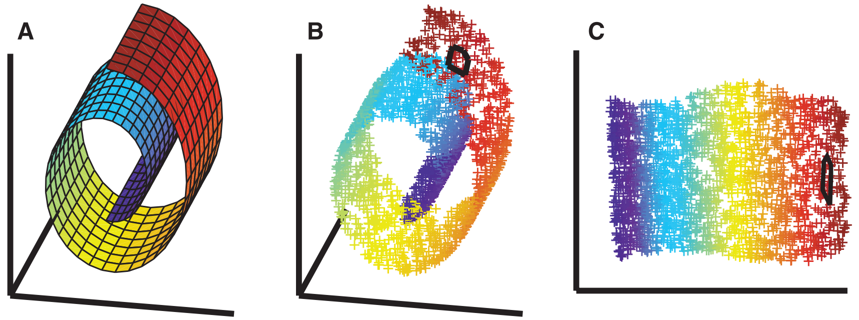
Unsupervised Learning

- Learn U such that local linearity is preserved
  – Lower dimensional than x
  – "Manifold Learning"

x's

u's

Any neighborhood
looks like a linear plane

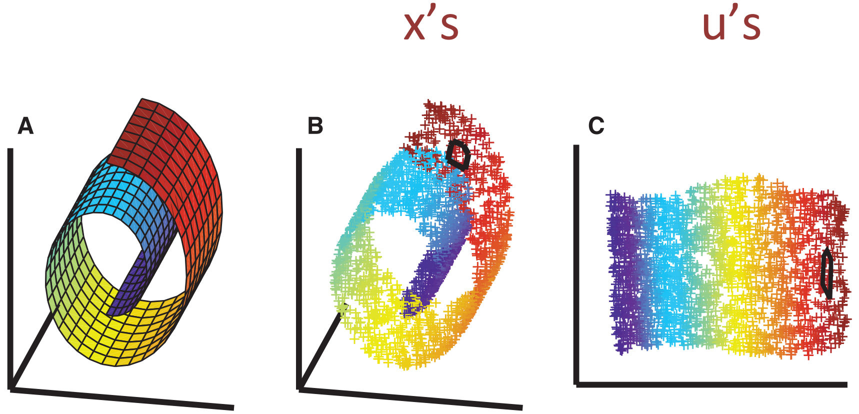


https://www.cs.nyu.edu/~roweis/lle/

Approach

- Define relationship of each x to its neighbors

- Find a lower dimensional u that preserves relationship



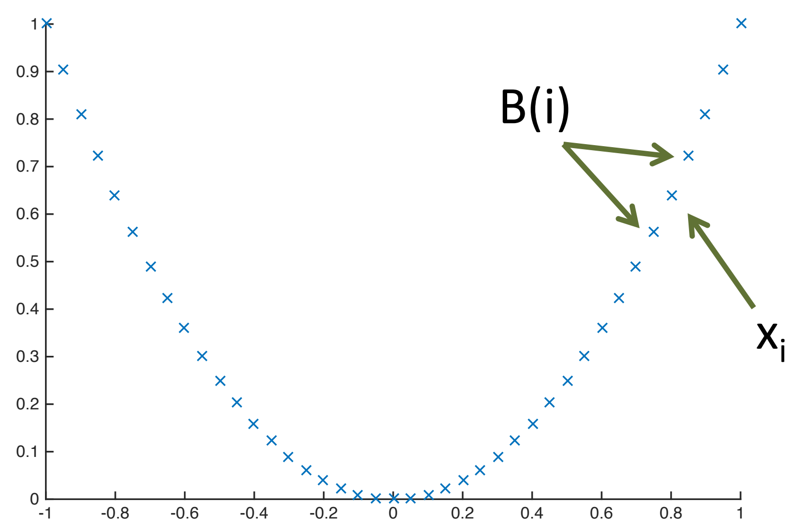
Locally Linear Embedding

$$S = \{x_i\}_{i=1}^{N}$$

- Create B(i)
  - B nearest neighbors of x_i
  - **Assumption:** B(i) is approximately linear
  - x_i can be written as a convex combination of x_j in B(i)

$$x_i \approx \sum_{j \in B(i)} W_{ij} x_j$$

$$\sum_{j \in B(i)} W_{ij} = 1$$



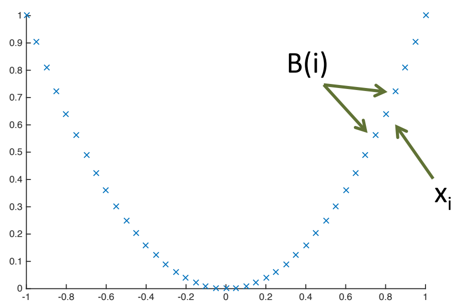
https://www.cs.nyu.edu/~roweis/lle/

Locally Linear Embedding

Given Neighbors B(i), solve local linear approximation W:

$$\underset{W}{\operatorname{argmin}} \sum_i \left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 \qquad \sum_{j \in B(i)} W_{ij} = 1$$

$$\begin{aligned}
\left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 &= \left\| \sum_{j \in B(i)} W_{ij} (x_i - x_j) \right\|^2 \\
&= \left(\sum_{j \in B(i)} W_{ij} (x_i - x_j) \right)^T \left(\sum_{j \in B(i)} W_{ij} (x_i - x_j) \right) \\
&= \sum_{j \in B(i)} \sum_{k \in B(i)} W_{ij} W_{ik} C^i_{jk} \\
&= W_{i,*}^T C^i W_{i,*}
\end{aligned}$$



https://www.cs.nyu.edu/~roweis/lle/

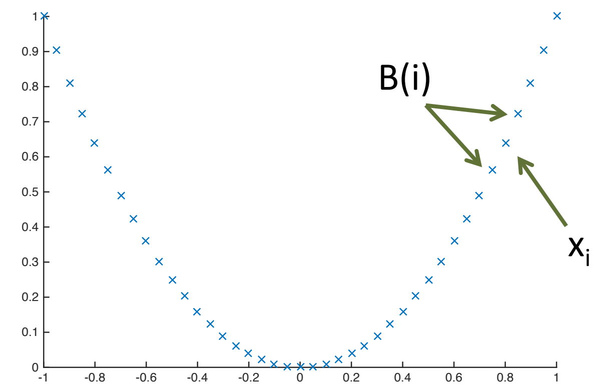
Locally Linear Embedding

Given Neighbors B(i), solve local linear approximation W:

$$\underset{W}{\operatorname{argmin}} \sum_i \left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 = \underset{W}{\operatorname{argmin}} \sum_i W_{i,*}^T C^i W_{i,*} \qquad \sum_{j \in B(i)} W_{ij} = 1$$

$$C_{jk}^i = (x_i - x_j)^T (x_i - x_k)$$

- Every x_i is approximated as a convex combination of neighbors
  - **How to solve?**

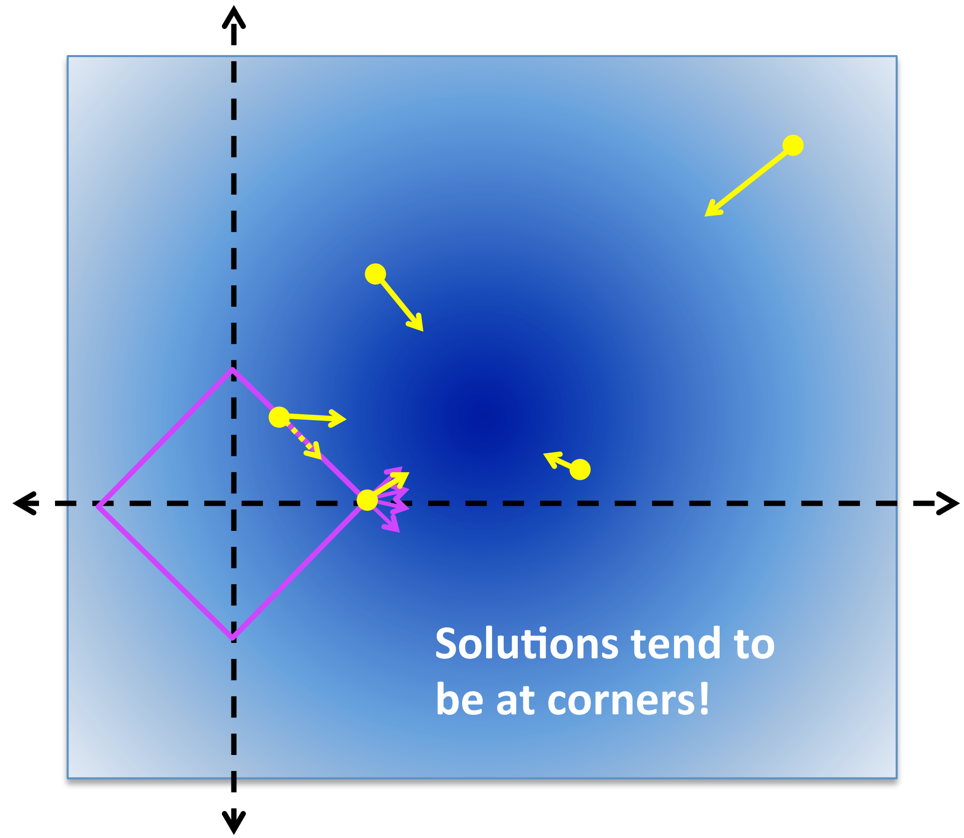


Lagrange Multipliers

$$\underset{w}{\operatorname{argmin}} L(w) \equiv w^T C w$$

$$\text{s.t. } |w| = 1$$

$$\nabla_{w_j} |w| \begin{cases} -1 & \text{if} \quad w_j < 0 \\ +1 & \text{if} \quad w_j > 0 \\ [-1, +1] & \text{if} \quad w_j = 0 \end{cases}$$



$$\exists \lambda \geq 0 : \left(\partial_w L(y, w) \in \lambda \nabla_w |w| \right) \wedge \left(|w| = 1 \right)$$

http://en.wikipedia.org/wiki/Lagrange_multiplier

Solving Locally Linear Approximation

Lagrangian:

$$L(W,\lambda) = \sum_i \left(W_{i,*}^T C^i W_{i,*} - \lambda_i \left(\vec{1}^T W_{i,*} - 1 \right) \right) \qquad \sum_j W_{ij} = \vec{1}^T W_{i,*}$$

$$\partial_{W_{i,*}} L(W,\lambda) = 2C^i W_{i,*} - \lambda_i \vec{1}$$

$$W_{i,*} = \frac{\lambda_i}{2} \left(C^i \right)^{-1} \vec{1} \propto \left(C^i \right)^{-1} \vec{1}$$

$$W_{ij} \propto \sum_{k \in B(i)} \left(C^i \right)_{jk}^{-1}$$

$$W_{ij} = \frac{\sum_{k \in B(i)} \left(C^i \right)_{jk}^{-1}}{\sum_{l \in B(i)} \sum_{m \in B(i)} \left(C^i \right)_{lm}^{-1}}$$

Locally Linear Approximation

$$x_i \approx \sum_{j \in B(i)} W_{ij} x_j$$

$$\sum_{j \in B(i)} W_{ij} = 1$$

- Invariant to:
  - Rotation $$A x_i \approx \sum_{j \in B(i)} A W_{ij} x_j$$
  - Scaling $$5 x_i \approx \sum_{j \in B(i)} 5 W_{ij} x_j$$
  - Translation $$x_i + x' \approx \sum_{j \in B(i)} W_{ij} \left(x_j + x' \right)$$

Story So Far: Locally Linear Embeddings

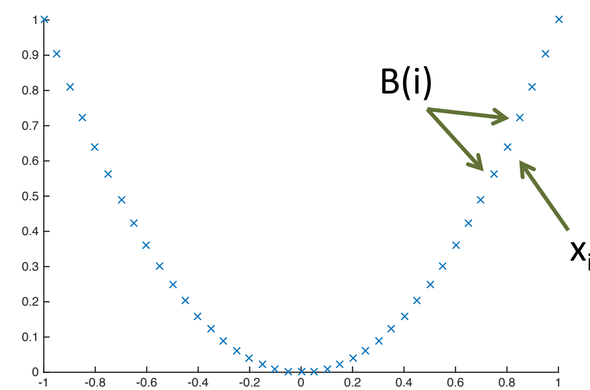
Given Neighbors B(i), solve local linear approximation W:

$$\underset{W}{\operatorname{argmin}} \sum_i \left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 = \underset{W}{\operatorname{argmin}} \sum_i W_{i,*}^T C^i W_{i,*} \qquad \sum_{j \in B(i)} W_{ij} = 1$$

Solution via Lagrange Multipliers:

$$C_{jk}^i = (x_i - x_j)^T (x_i - x_k)$$

$$W_{ij} = \frac{\sum_{k \in B(i)} \left(C^i\right)_{jk}^{-1}}{\sum_{l \in B(i)} \sum_{m \in B(i)} \left(C^i\right)_{lm}^{-1}}$$



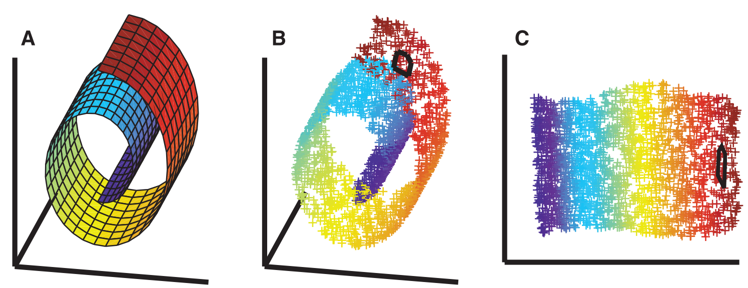
- **Locally Linear Approximation**

https://www.cs.nyu.edu/~roweis/lle/

Recall: Locally Linear Embedding

- Given: $S = \{x_i\}_{i=1}^{N}$
- Learn U such that local linearity is preserved
  - Lower dimensional than x
  - "Manifold Learning"

x's u's



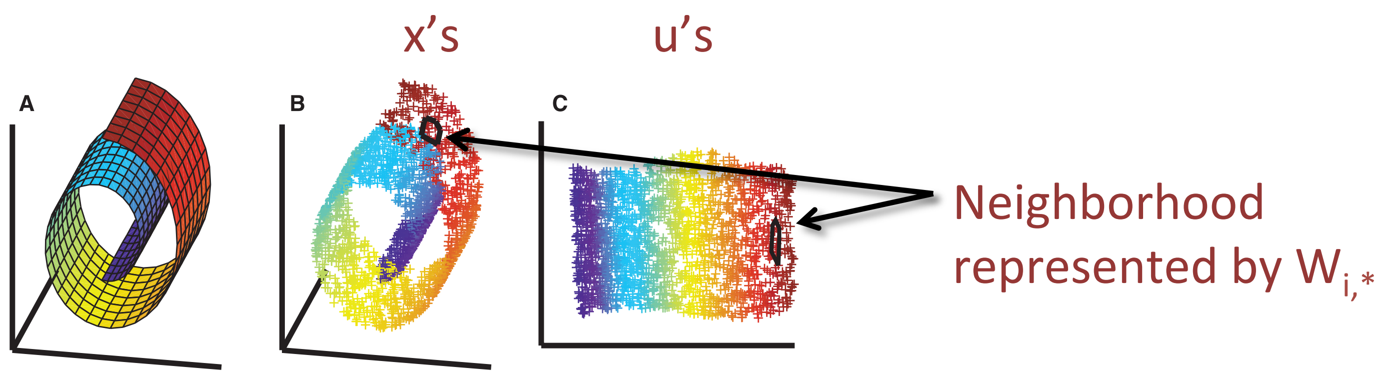
https://www.cs.nyu.edu/~roweis/lle/

Dimensionality Reduction
(Learning the Embedding)

Given local approximation W, learn lower dimensional representation:

$$\underset{U}{\operatorname{argmin}} \sum_i \left\| u_i - \sum_{j \in B(i)} W_{ij} u_j \right\|^2$$

- Find low dimensional U
  - Preserves approximate local linearity



https://www.cs.nyu.edu/~roweis/lle/

Given local approximation W, learn lower dimensional representation:

$$\operatorname*{argmin}_U \sum_i \left\| u_i - \sum_{j \in B(i)} W_{ij} u_j \right\|^2$$

$$UU^T = I_K$$

$$\sum_i u_i = \vec{0}$$

- Rewrite as:

$$\operatorname*{argmin}_U \sum_{ij} M_{ij} \left(u_i^T u_j \right) \equiv \operatorname{trace}\left(UMU^T \right)$$

$$M_{ij} = 1_{[i=j]} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}$$

$$M = (I_N - W)^T (I_N - W)$$

Symmetric positive semidefinite

https://www.cs.nyu.edu/~roweis/lle/

Given local approximation W, learn lower dimensional representation:

$$\underset{U}{\operatorname{argmin}} \sum_{ij} M_{ij}\left(u_i^T u_j\right) \equiv \operatorname{trace}\left(UMU^T\right) \qquad UU^T = I_K$$

$$\sum_i u_i = \vec{0}$$

- Suppose K=1

$$\underset{u}{\operatorname{argmin}} \sum_{ij} M_{ij}\left(u_i^T u_j\right) \equiv \operatorname{trace}\left(uMu^T\right) \qquad uu^T = 1$$

$$= \underset{u}{\operatorname{argmax}} \operatorname{trace}\left(uM^+u^T\right)$$

pseudoinverse

- By min-max theorem
  - u = principal eigenvector of M^+

http://en.wikipedia.org/wiki/Min-max_theorem

Recap: Principal Component Analysis

$$M = V\Lambda V^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & 0 & \\ & & & 0 \end{bmatrix}$$

- Each column of V is an Eigenvector
- Each λ is an Eigenvalue ($\lambda_1 \geq \lambda_2 \geq \ldots$)

$$M^+ = V\Lambda^+ V^T$$

$$\Lambda^+ = \begin{bmatrix} 1/\lambda_1 & & & \\ & 1/\lambda_2 & & \\ & & 0 & \\ & & & 0 \end{bmatrix}$$

$$MM^+ = V\Lambda\Lambda^+ V^T = V_{1:2}V_{1:2}^T = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & & 0 \end{bmatrix}$$

Given local approximation W, learn lower dimensional representation:

$$\underset{U}{\operatorname{argmin}} \sum_{ij} M_{ij} \left(u_i^T u_j \right) \equiv \operatorname{trace}\left(U M U^T \right) \qquad U U^T = I_K$$

$$\sum_i u_i = \vec{0}$$

- **K=1:**
  - u = principal eigenvector of M^+
  - u = smallest non-trivial eigenvector of M
    - Corresponds to smallest non-zero eigenvalue

- **General K**
  - U = top K principal eigenvectors of M^+
  - U = bottom K non-trivial eigenvectors of M
    - Corresponds to bottom K non-zero eigenvalues

https://www.cs.nyu.edu/~roweis/lle/

http://en.wikipedia.org/wiki/Min-max_theorem

Recap: Locally Linear Embedding

- Generate nearest neighbors of each x_i, B(i)

- Compute Local Linear Approximation:

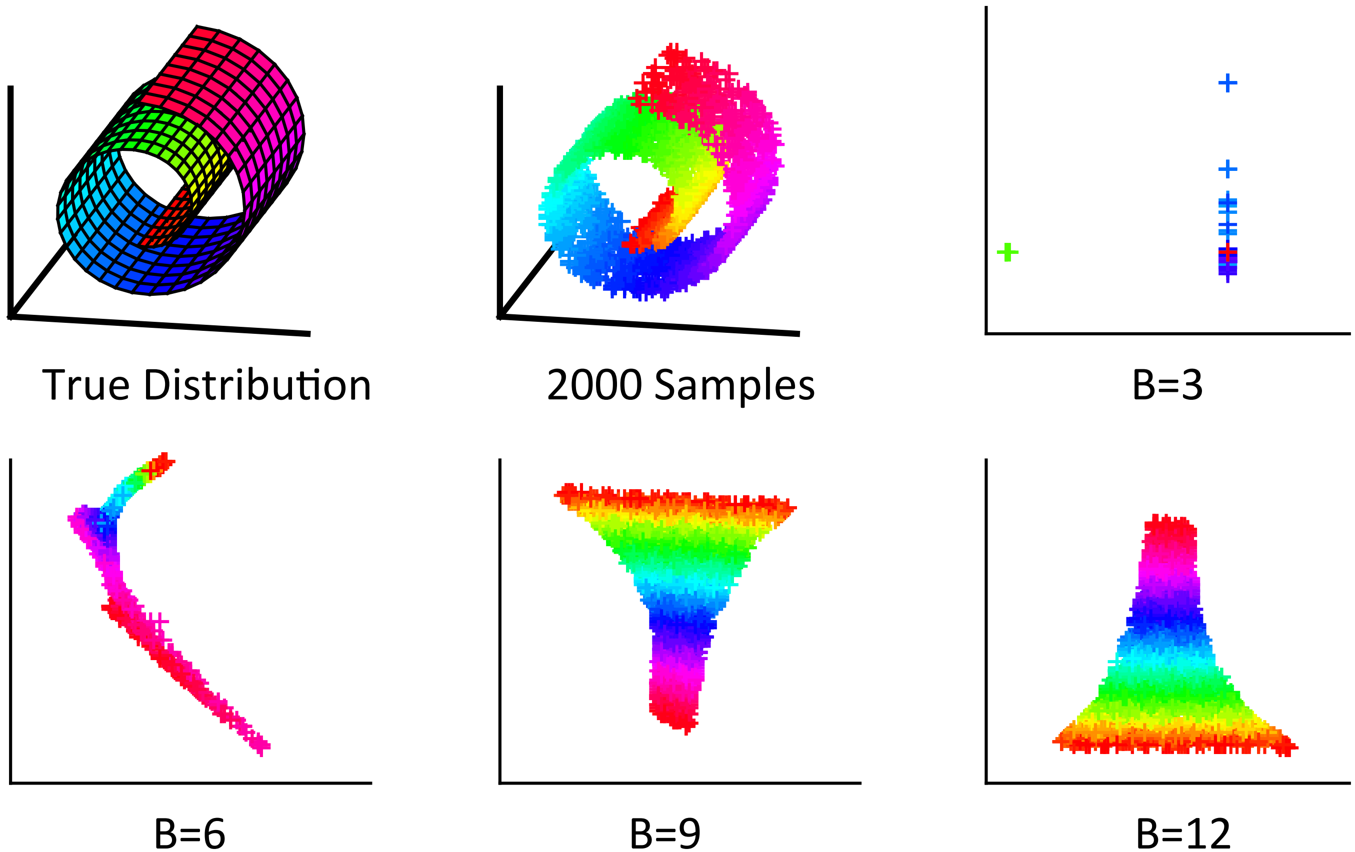
$$\underset{W}{\operatorname{argmin}} \sum_i \left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 \qquad \sum_{j \in B(i)} W_{ij} = 1$$

- Compute low dimensional embedding

$$\underset{U}{\operatorname{argmin}} \sum_i \left\| u_i - \sum_{j \in B(i)} W_{ij} u_j \right\|^2 \qquad \begin{aligned} UU^T &= I_K \\ \sum_i u_i &= \vec{0} \end{aligned}$$

Results for Different Neighborhoods

(K=2)



https://www.cs.nyu.edu/~roweis/lle/gallery.html

Probabilistic Sequence Embeddings

Example 1: Playlist Embedding

- Users generate song playlists
  - Treat as training data

- **Can we learn a probabilistic model of playlists?**

Example 2: Word Embedding

- People write natural text all the time
  - Treat as training data

- **Can we learn a probabilistic model of word sequences?**

Probabilistic Sequence Modeling

- Training set:

$$S = \left\{ s_1, \ldots s_{|S|} \right\} \qquad D = \left\{ p_i \right\}_{i=1}^{N} \qquad p_i = \left\langle p_i^1, \ldots, p_i^{N_i} \right\rangle$$

Songs, Words — Playlists, Documents — Sequence Definition

- **Goal:** Learn a Markov model of sequences:

$$P(p_i^j \mid p_i^{j-1})$$

- What is the form of P?

First Try: Probability Tables

$P(s\|s')$	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_{start}
s_1	0.01	0.03	0.01	0.11	0.04	0.04	0.01	0.05
s_2	0.03	0.01	0.04	0.03	0.02	0.01	0.02	0.02
s_3	0.01	0.01	0.01	0.07	0.02	0.02	0.05	0.09
s_4	0.02	0.11	0.07	0.01	0.07	0.04	0.01	0.01
s_5	0.04	0.01	0.02	0.17	0.01	0.01	0.10	0.02
s_6	0.01	0.02	0.03	0.01	0.01	0.01	0.01	0.08
s_7	0.07	0.02	0.01	0.01	0.03	0.09	0.03	0.01

...

⋮

First Try: Probability Tables

$P(s \mid s')$	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_{start}
s_1	0.01	0.03	0.01	0.11	0.04	0.04	0.01	0.05
s_2	0.03	0.01	0.04	0.03	0.02	0.01	0.02	0.02
s_3	0.01	0.01	0.01	0.07	0.02	0.02	0.05	0.09
s_4	0.02	0.11	0.07	0.01	0.07	0.04	0.01	0.01
s_5								
s_6								
s_7								

…

#Parameters = $O(|S|^2)$!!!

(worse for higher-order sequence models)

Second Try: Hidden Markov Models

$$P\left(p_i, z\right) = P(End \mid z^{N_i}) \prod_{j=1}^{N_i} P(z^j \mid z^{j-1}) \prod_{j=1}^{N_j} P(p_i^j \mid z^j)$$

$P(z^j \mid z^{j-1})$

- #Parameters = O(K^2)

$P(p_i^j \mid z^j)$

- #Parameters = O(|S|K)

- Total = O(K^2) + O(|S|K)

Problem with Hidden Markov Models

$$P\left(p_i, z\right) = P(End \mid z^{N_i}) \prod_{j=1}^{N_i} P(z^j \mid z^{j-1}) \prod_{j=1}^{N_j} P(p_i^j \mid z^j)$$

- Need to reliably estimate P(s|z)

$$S = \left\{s_1, \ldots s_{|S|}\right\} \qquad D = \left\{p_i\right\}_{i=1}^{N} \qquad p_i = \left\langle p_i^1, \ldots, p_i^{N_i} \right\rangle$$

- **Hard to do!**

Outline for Sequence Modeling

- Playlist Embedding
  - Distance-based embedding
  - http://www.cs.cornell.edu/people/tj/playlists/index.html

- Word Embedding (word2vec) **Homework Question!**
  - Inner-product embedding
  - https://code.google.com/archive/p/word2vec/

- Compare the two approaches

Markov Embedding (Distance)

u_s: entry point of song s
v_s: exit point of song s

$$P(s \mid s') \propto \exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}$$

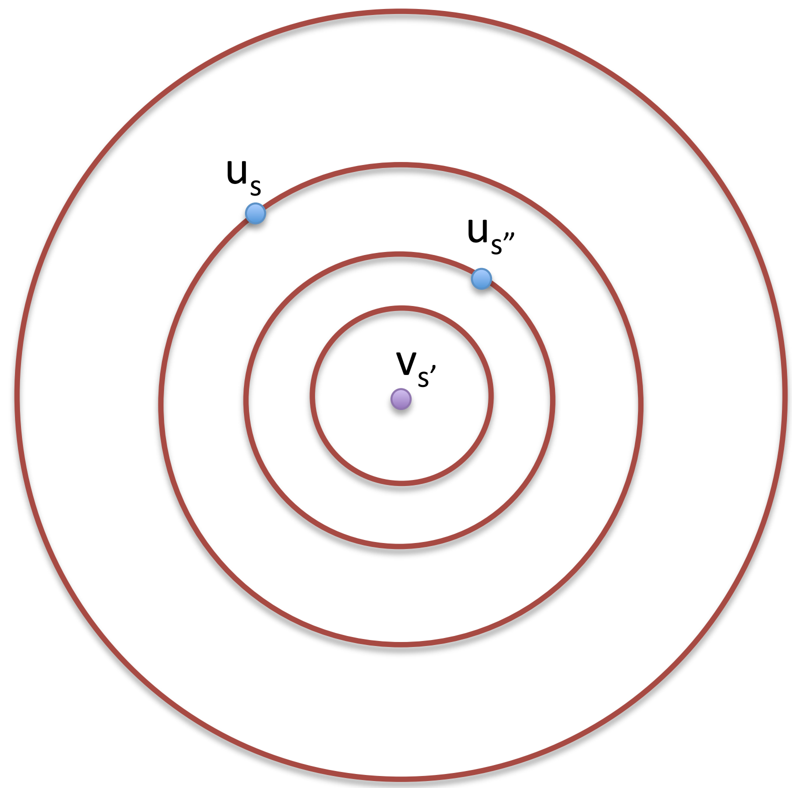
$$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}}{\sum_{s''} \exp\left\{-\left\|u_{s''} - v_{s'}\right\|^2\right\}}$$

- “Log-Radial” function
  - (my own terminology)

http://www.cs.cornell.edu/People/tj/publications/chen_etal_12a.pdf

Log-Radial Functions

$$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}}{\sum_{s''} \exp\left\{-\left\|u_{s''} - v_{s'}\right\|^2\right\}}$$



2K parameters per song
2|S|K parameters total

Each ring defines an equivalence class of transition probabilities

Learning Problem

$$S = \{s_1, \ldots s_{|S|}\}$$

Songs

$$D = \{p_i\}_{i=1}^{N}$$

Playlists

$$p_i = \langle p_i^1, \ldots, p_i^{N_i} \rangle$$

Playlist Definition
(each p^j corresponds to a song)

- Learning Goal:

Sequences

Tokens in each Sequence

$$\underset{U,V}{\operatorname{argmax}} \prod_i P(p_i) = \prod_i \prod_j P(p_i^j \mid p_i^{j-1})$$

$$P(s \mid s') = \frac{\exp\left\{-\|u_s - v_{s'}\|^2\right\}}{\sum_{s''} \exp\left\{-\|u_{s''} - v_{s'}\|^2\right\}} = \frac{\exp\left\{-\|u_s - v_{s'}\|^2\right\}}{Z(s')}$$

http://www.cs.cornell.edu/People/tj/publications/chen_etal_12a.pdf

Minimize Neg Log Likelihood

$$\underset{U,V}{\operatorname{argmax}} \prod_i \prod_j P(p_i^j \mid p_i^{j-1}) = \underset{U,V}{\operatorname{argmin}} \sum_i \sum_j -\log P(p_i^j \mid p_i^{j-1})$$

- Solve using gradient descent
  - Random initialization

- Normalization constant hard to compute:
  - Approximation heuristics
    - See paper

$$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}}{Z(s')}$$

http://www.cs.cornell.edu/People/tj/publications/chen_etal_12a.pdf

Story so Far: Playlist Embedding

- Training set of playlists
  - Sequences of songs
- Want to build probability tables P(s|s')
  - But a lot of missing values, hard to generalize directly
  - Assume low-dimensional embedding of songs

$$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}}{\sum_{s''} \exp\left\{-\left\|u_{s''} - v_{s'}\right\|^2\right\}} = \frac{\exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}}{Z(s')}$$

Simpler Version

- Dual point model: $$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}}{Z(s')}$$

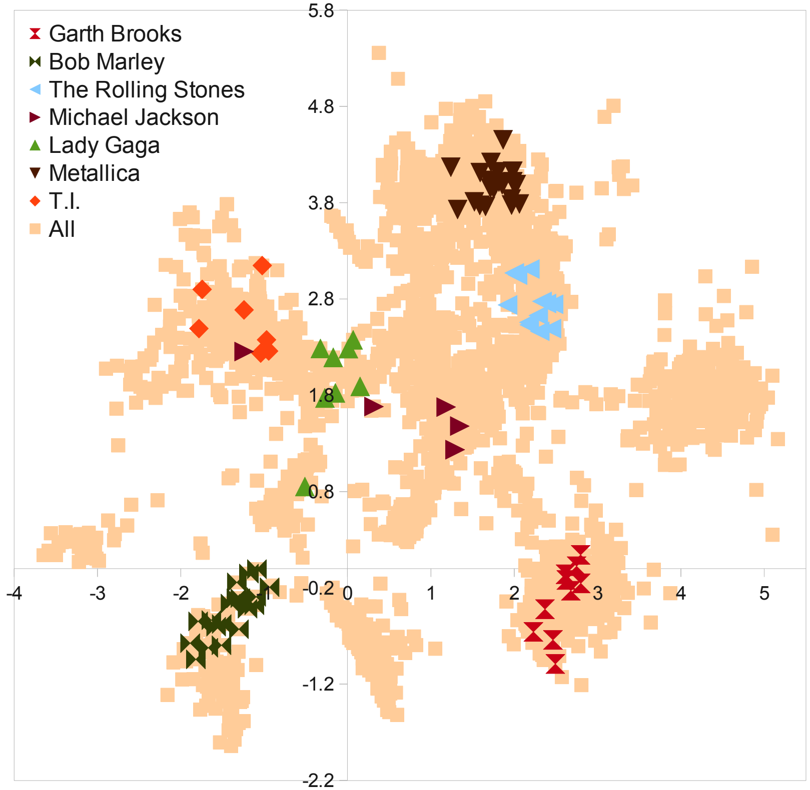
- Single point model: $$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - u_{s'}\right\|^2\right\}}{Z(s')}$$
  – Transitions are symmetric
    - (almost)
  – Exact same form of training problem

Visualization in 2D

Simpler version:
Single Point Model

$$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - u_{s'}\right\|^2\right\}}{Z(s')}$$

Single point model is easier to visualize



http://www.cs.cornell.edu/People/tj/publications/chen_etal_12a.pdf

Sampling New Playlists

- Given partial playlist:

$$p = \left\langle p^1, \ldots p^j \right\rangle$$

- Generate next song for playlist p^{j+1}
  - Sample according to:

$$P(s \mid p^j) = \frac{\exp\left\{ -\left\| u_s - v_{p^j} \right\|^2 \right\}}{Z(p^j)}$$

Dual Point Model

$$P(s \mid p^j) = \frac{\exp\left\{ -\left\| u_s - u_{p^j} \right\|^2 \right\}}{Z(p^j)}$$

Single Point Model

http://www.cs.cornell.edu/People/tj/publications/chen_etal_12a.pdf

Demo

http://jimi.ithaca.edu/~dturnbull/research/lme/lmeDemo.html

What About New Songs?

- Suppose we've trained U:

$$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - u_{s'}\right\|^2\right\}}{Z(s')}$$

- What if we add a new song s'?
  - No playlists created by users yet...
  - Only options: $u_{s'} = 0$ or $u_{s'} = \text{random}$
    - Both are terrible!
    - "Cold-start" problem

Song & Tag Embedding

- Songs are usually added with tags
  - E.g., indie rock, country
  - Treat as features or attributes of songs
- How to leverage tags to generate a reasonable embedding of new songs?
  - **Learn an embedding of tags as well!**

http://www.cs.cornell.edu/People/tj/publications/moore_etal_12a.pdf

$$S = \{s_1, \ldots s_{|S|}\}$$

Songs

$$D = \{p_i\}_{i=1}^{N}$$

Playlists

$$p_i = \langle p_i^1, \ldots, p_i^{N_i} \rangle$$

Playlist Definition

$$T = \{T_1, \ldots T_{|S|}\}$$

Tags for Each Song

Learning Objective:

$$\underset{U,A}{\operatorname{argmax}} P(D \mid U) P(U \mid A, T)$$

Same term as before: $P(D \mid U) = \prod_i P(p_i \mid U) = \prod_i \prod_j P(p_i^j \mid p_i^{j-1}, U)$

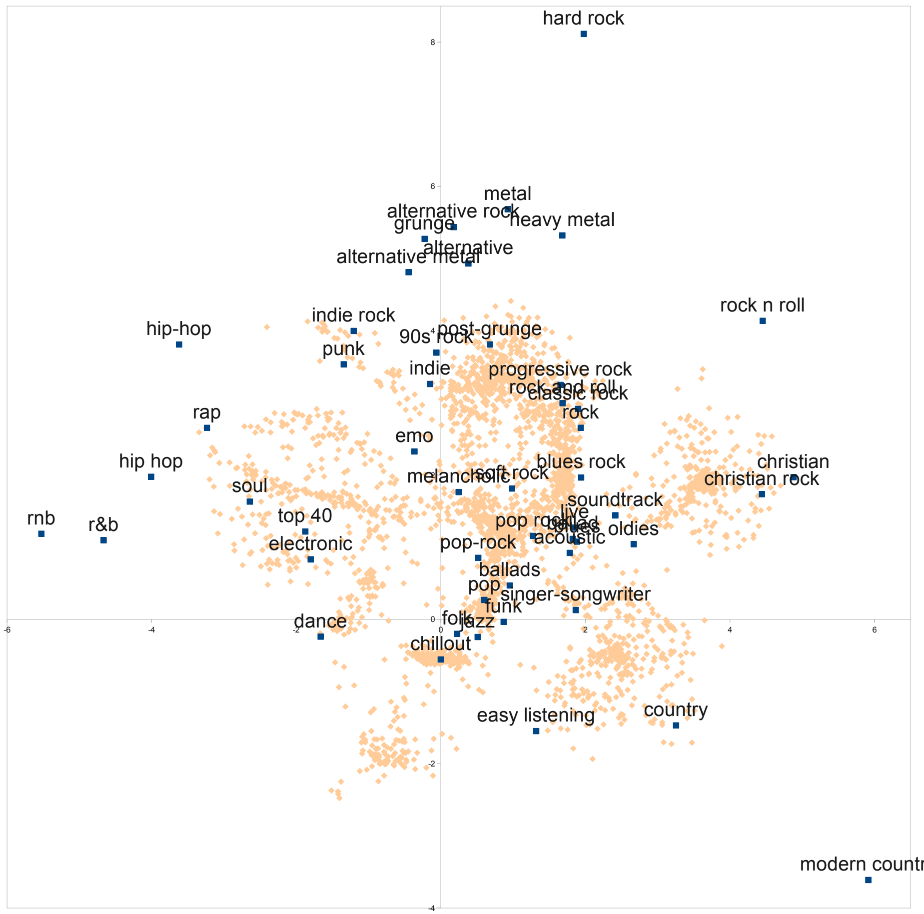
Song embedding ≈ average of tag embeddings:

$$P(U \mid A, T) = \prod_s P(u_s \mid A, T_S) \propto \prod_s \exp\left\{ -\lambda \left\| u_s - \frac{1}{|T_s|} \sum_{t \in T_s} A_t \right\|^2 \right\}$$

Solve using gradient descent:

http://www.cs.cornell.edu/People/tj/publications/moore_etal_12a.pdf

Visualization in 2D



http://www.cs.cornell.edu/People/tj/publications/moore_etal_12a.pdf

Revisited: What About New Songs?

- No user has s' added to playlist
  - So no evidence from playlist training data:

s' does not appear in $D = \{p_i\}_{i=1}^{N}$

- Assume new song has been tagged $T_{s'}$
  - The $u_{s'}$ = average of A_t for tags t in $T_{s'}$
  - Implication from objective:

$$\underset{U,A}{\operatorname{argmax}} P(D \mid U) P(U \mid A, T)$$

Switching Gears: Word Embeddings

- Given a large corpus
  - Wikipedia
  - Google News

- Learn a word embedding to model sequences of words (e.g., sentences)

https://code.google.com/archive/p/word2vec/

Switching Gears: Inner Product Embeddings

- Previous: capture semantics via distance

$$P(s \mid s') = \frac{\exp\left\{-\left\|u_s - v_{s'}\right\|^2\right\}}{\sum_{s''} \exp\left\{-\left\|u_{s''} - v_{s'}\right\|^2\right\}}$$

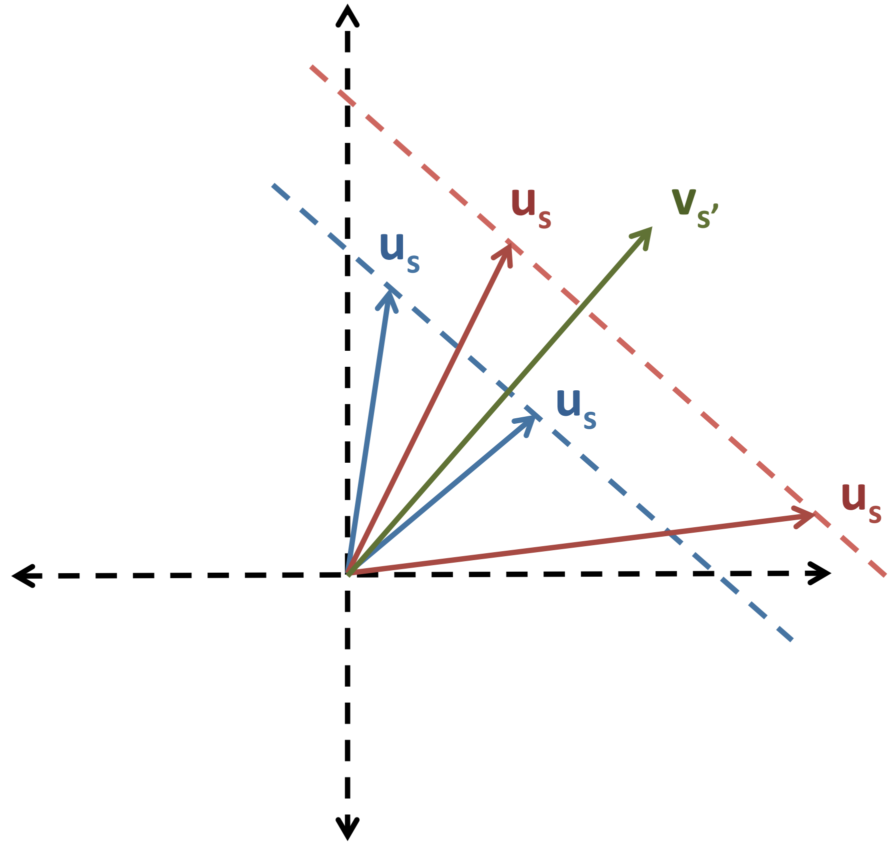
- Can also capture semantics via inner product

$$P(s \mid s') = \frac{\exp\left\{u_s^T v_{s'}\right\}}{\sum_{s''} \exp\left\{u_{s''}^T v_{s'}\right\}}$$

Basically a latent-factor model!

Log-Linear Embeddings

$$P(s \mid s') = \frac{\exp\left\{\boldsymbol{u}_s^T \boldsymbol{v}_{s'}\right\}}{\sum_{s''} \exp\left\{\boldsymbol{u}_{s''}^T \boldsymbol{v}_{s'}\right\}}$$



2K parameters per song
2|S|K parameters total

Each projection level onto the green line defines an equivalence class

Learning Problem (Version 1)

$$S = \{s_1, \ldots s_{|S|}\} \qquad D = \{p_i\}_{i=1}^{N} \qquad p_i = \langle p_i^1, \ldots, p_i^{N_i} \rangle$$

Words

Sentences

Sentence Definition
(Each p^j is a word)

- Learning Goal:

Sequences

Tokens in each Sequence

$$\underset{U,V}{\operatorname{argmax}} \prod_i P(p_i) = \prod_i \prod_j P(p_i^j \mid p_i^{j-1})$$

$$P(s \mid s') = \frac{\exp\{u_s^T v_{s'}\}}{\sum_{s''} \exp\{u_{s''}^T v_{s'}\}} = \frac{\exp\{u_s^T v_{s'}\}}{Z(s')}$$

Skip-Gram Model (word2vec)

- Predict probability of any neighboring word

Sequences   Tokens in each Sequence

$$\underset{U,V}{\operatorname{argmax}} \prod_{i} \prod_{j} \prod_{k \in [-C, C] \setminus 0} P(p_i^{j+k} \mid p_i^j)$$

Skip Length

$$P(s \mid s') = \frac{\exp\left\{u_s^T v_{s'}\right\}}{\sum_{s''} \exp\left\{u_{s''}^T v_{s'}\right\}} = \frac{\exp\left\{u_s^T v_{s'}\right\}}{Z(s')}$$

https://code.google.com/archive/p/word2vec/

Skip-Gram Model (word2vec)

- Predict probability of any neighboring word

Sequences   Tokens in each Sequence

$$\underset{U,V}{\operatorname{argmax}} \prod_{i} \prod_{j} \prod_{k \in [-C,C] \backslash 0} P(p_i^{j+k} \mid p_i^j)$$

Skip Length

What are benefits of Skip-Gram model?

https://code.google.com/archive/p/word2vec/

Intuition of Skip-Gram Model

- "The dog jumped over the fence."
- "My dog ate my homework."
- "I walked my dog up to the fence."

Example sentences

$$\underset{U,V}{\operatorname{argmax}} \prod_{i} \prod_{j} \prod_{k \in [-C,C] \setminus 0} P(p_i^{j+k} \mid p_i^j)$$

- Distribution of neighboring words more peaked
- Distribution of further words more diffuse
- Capture everything in a single model

Dimensionality Reduction

- What dimensionality should we choose U,V?
  - E.g., what should K be?

$$P(s \mid s') = \frac{\exp\left\{u_s^T v_{s'}\right\}}{\sum_{s''} \exp\left\{u_{s''}^T v_{s'}\right\}}$$

- K = $|S|^2$ implies we can memorize every word pair interaction
- Smaller K assumes words lie in lower-dimensional space
  - Easier to generalize across words
- Larger K can overfit

Example 1

- $v_{\text{Czech}} + v_{\text{currency}} \approx v_{\text{koruna}}$

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

http://arxiv.org/pdf/1310.4546.pdf

Example 2

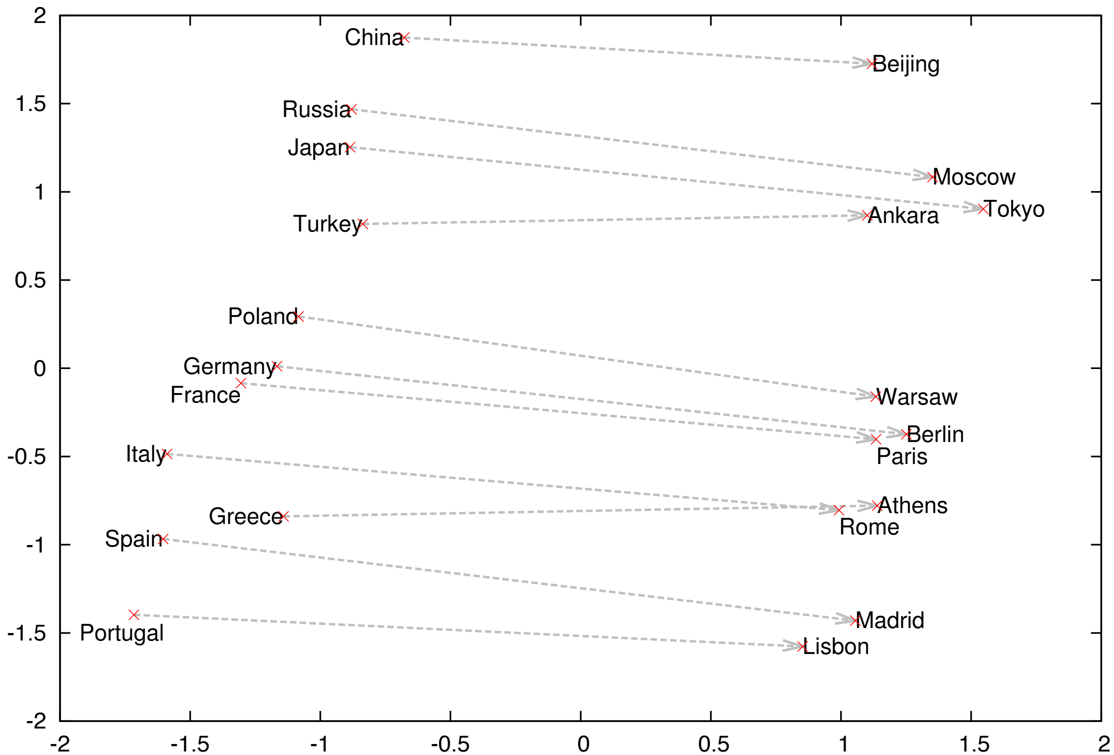
- E.g., $v_{France} - v_{Paris} + v_{Italy} \approx v_{Rome}$

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

http://arxiv.org/pdf/1301.3781.pdf

Example 3

- 2D PCA projection of countries and cities:



http://arxiv.org/pdf/1310.4546.pdf

Aside: Embeddings as Features

- Use the learned u (or v) as features

- E.g., linear models for classification:

$$h(x) = \text{sign}\left(w^T \phi(x)\right)$$

Can be word identities or word2vec representation!

Training word2vec

- Train via gradient descent

Sequences

Tokens in each Sequence

$$\underset{U,V}{\operatorname{argmin}} \sum_{i} \sum_{j} \sum_{k\in[-C,C]\setminus 0} -\log P(p_i^{j+k} \mid p_i^{j})$$

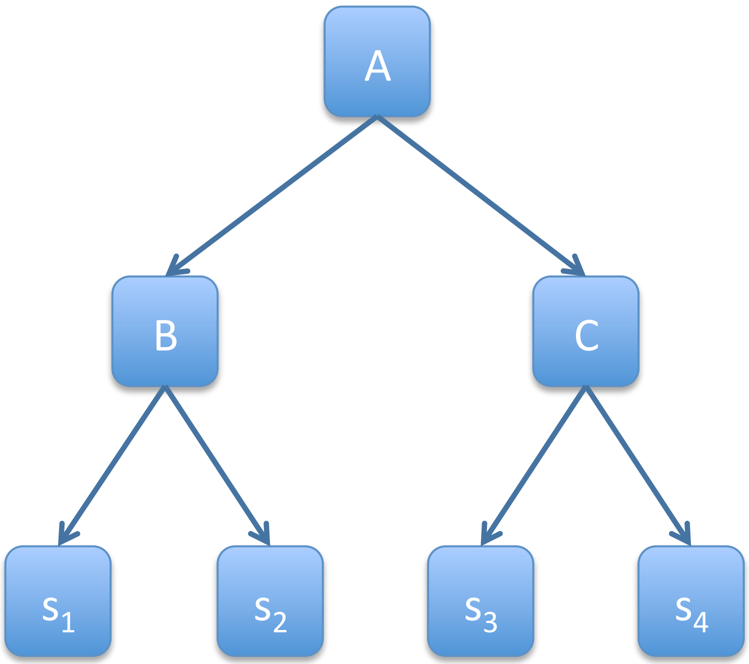
Skip Length

$$P(s \mid s') = \frac{\exp\left\{u_s^T v_{s'}\right\}}{\sum_{s''} \exp\left\{u_{s''}^T v_{s'}\right\}} = \frac{\exp\left\{u_s^T v_{s'}\right\}}{Z(s')}$$

Denominator expensive!

https://code.google.com/archive/p/word2vec/

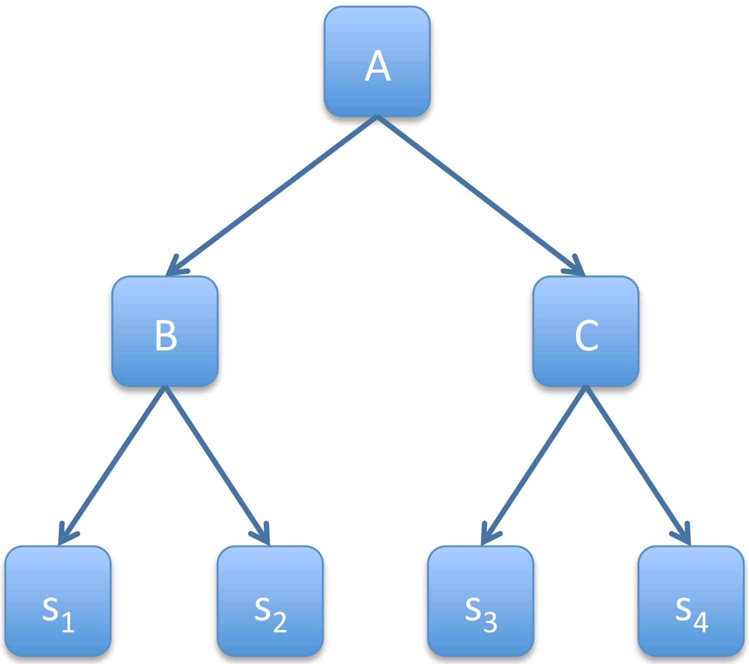
Hierarchical Approach
(Probabilistic Decision Tree)



- Decision tree of paths
- Leaf node = word
- Choose each branch independently

http://arxiv.org/pdf/1310.4546.pdf

Hierarchical Approach
(Probabilistic Decision Tree)



$$P(s_1 \mid s') = P(B \mid A, s')P(s_1 \mid B, s')$$

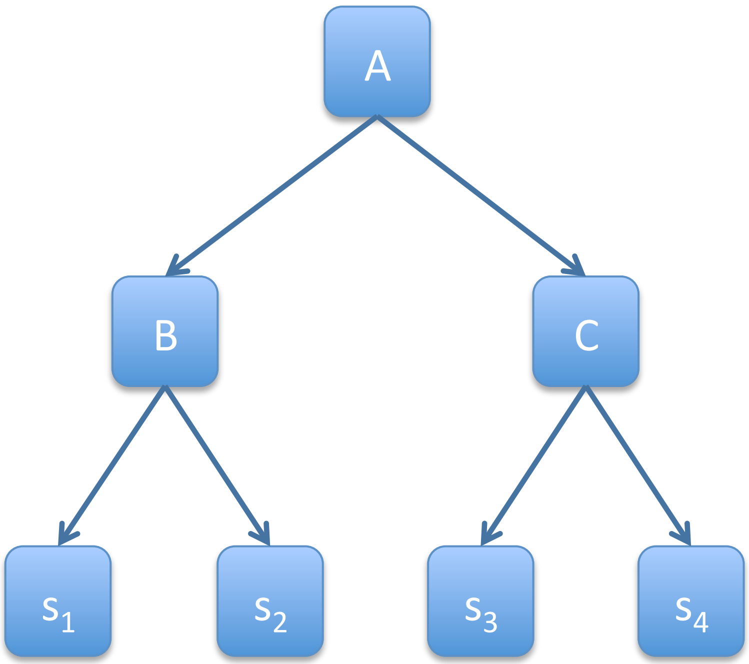
$$P(s_2 \mid s') = P(B \mid A, s')P(s_2 \mid B, s')$$

$$P(s_3 \mid s') = P(C \mid A, s')P(s_3 \mid C, s')$$

$$P(s_4 \mid s') = P(C \mid A, s')P(s_4 \mid C, s')$$

http://arxiv.org/pdf/1310.4546.pdf

Hierarchical Approach
(Probabilistic Decision Tree)



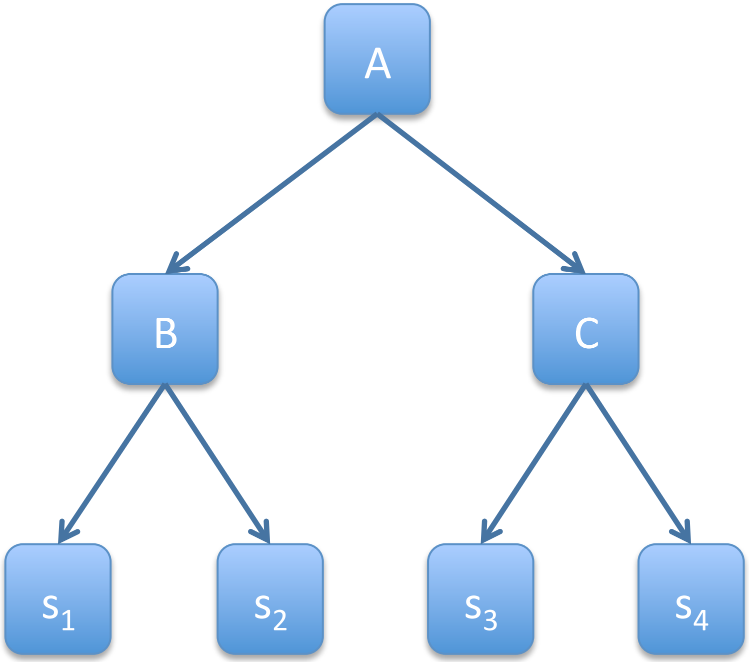
$$P(B \mid A, s) = \frac{1}{1+\exp\left\{-u_{BC}^T v_s\right\}} = \frac{1}{1+\exp\left\{u_{CB}^T v_s\right\}}$$

$$P(C \mid A, s) = \frac{1}{1+\exp\left\{-u_{CB}^T v_s\right\}} = \frac{1}{1+\exp\left\{u_{BC}^T v_s\right\}}$$

$$u_{BC} = -u_{CB}$$

http://arxiv.org/pdf/1310.4546.pdf

Hierarchical Approach
(Probabilistic Decision Tree)



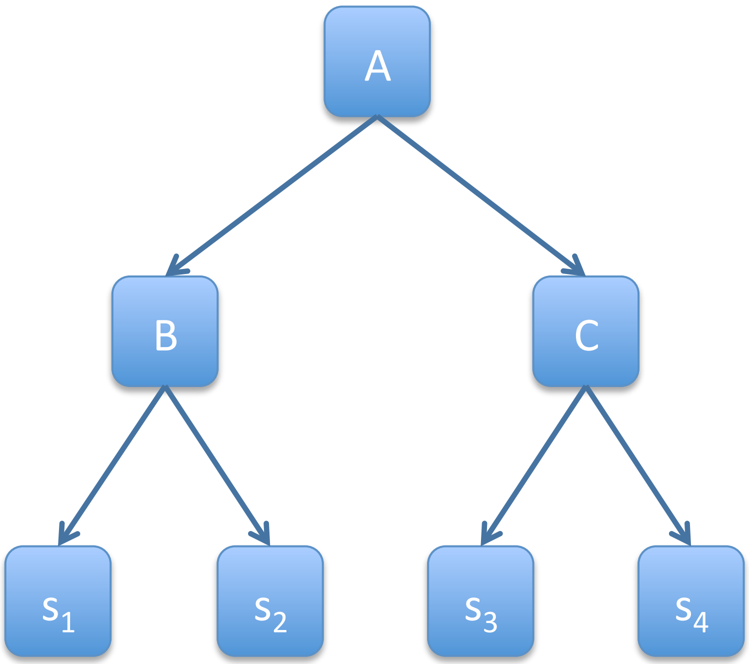
$$P(s_1 \mid B, s) = \frac{1}{1+\exp\left\{-u_{12}^T v_s\right\}} = \frac{1}{1+\exp\left\{u_{21}^T v_s\right\}}$$

$$P(s_2 \mid B, s) = \frac{1}{1+\exp\left\{-u_{21}^T v_s\right\}} = \frac{1}{1+\exp\left\{u_{12}^T v_s\right\}}$$

$$u_{12} = -u_{21}$$

http://arxiv.org/pdf/1310.4546.pdf

Hierarchical Approach
(Probabilistic Decision Tree)



- Compact formula:

Internal node at level m on path to leaf node s

$$P(s \mid s') = \prod_m P(n_{m,s} \mid n_{m-1,s}, s)$$

Levels in tree

Training Hierarchical Approach

- Train via gradient descent (same as before!)

Sequences

Tokens in each Sequence

$$\underset{U,V}{\operatorname{argmin}} \sum_i \sum_j \sum_{k \in [-C,C] \setminus 0} -\log P(p_i^{j+k} \mid p_i^j)$$

Skip Length

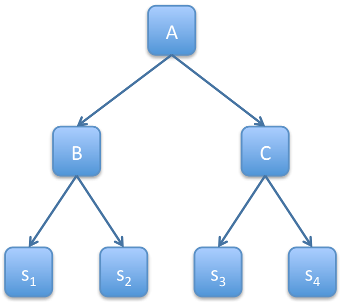
$$P(s \mid s') = \prod_m P(n_{m,s} \mid n_{m-1,s}, s)$$

Complexity = O($\log_2$(|S|))!

https://code.google.com/archive/p/word2vec/

Summary: Hierarchical Approach

- Each word has s corresponds to:
  - One v_s
  - $Log_2(|S|)$ u's!



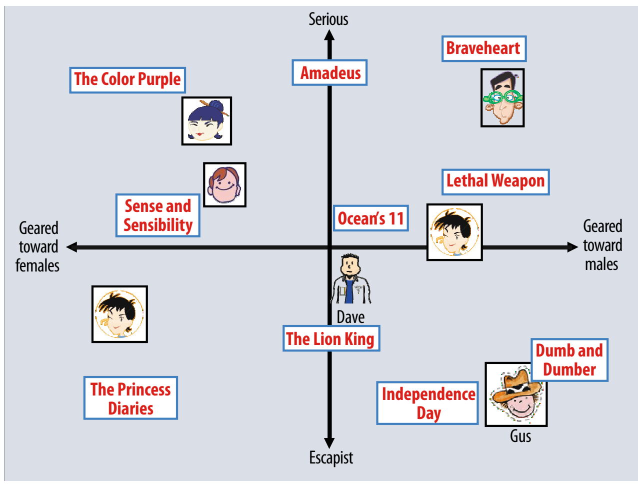
- Target factors u's are shared across words
  - Total number of U is still $O(|S|)$
- Previous use cases unchanged
  - They all used v_s
  - Vector subtraction, use as features for CRF, etc.

Recap: Embeddings

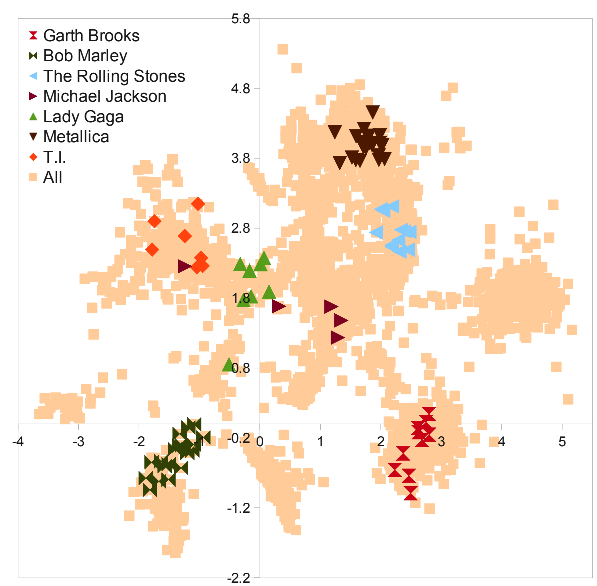
- **Given:** Training Data
  - Care about some property of training data
    - Markov Chain
    - Skip-Gram

- **Goal:** learn low dim representation
  - “Embedding”
  - Geometry of embedding captures property of interest
    - Either by distance or by inner-product

Visualization Semantics



Inner-Product Embeddings
Similarity measured via dot product
Rotational semantics
Can interpret axes
Can only visualize 2 axes at a time



Distance-Based Embedding
Similarity measured via distance
Clustering/locality semantics
Cannot interpret axes
Can visualize many clusters simultaneously

Next Week

- Recent Applications Lectures
- Latent Factor Models
- Deep Generative Models