

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 11:
Embeddings

Kaggle Competition

1	Tanner Moore Models				0.68560	16	1	—	Tanner Moore Models				0.67248
⋮	a TA got creative but no time				0.64637		⋮		a TA got creative but no time				0.63789
2	Heavy Head				0.64307	38	2	—	Heavy Head				0.63595
3	sample_submission-2.csv				0.64281	31	3	—	sample_submission-2.csv				0.63462
4	The Company				0.64234	35	4	—	The Company				0.63444
5	the coli crew				0.64121	18	⋮		DE\$traw, €dot, \$¥F Want \$...				0.63351
⋮	DE\$traw, €dot, \$¥F Want \$...				0.64117		5	▲ 2	PurkinjeCells				0.63339
6	We Want Jim and James				0.64114	26	6	▲ 2	deshew				0.63338
7	PurkinjeCells				0.64095	33	7	▼ 1	We Want Jim and James				0.63310
8	deshew				0.64093	11	8	▼ 3	the coli crew				0.63256
9	discounted freeze-dried eggs				0.63922	46	9	▲ 1	elggak				0.63146
10	elggak				0.63920	29	10	▲ 1	AdaGradStudents				0.63099
11	AdaGradStudents				0.63864	28	11	▲ 4	RNGesus				0.63096
12	Intel Atom Inside™				0.63859	26	12	▲ 2	0 Loss or Bust				0.63079
⋮	Jim and James Want \$ \$				0.63838		13	▼ 4	discounted freeze-dried eggs				0.63074
13	Random Variable				0.63837	23	14	▼ 2	Intel Atom Inside™				0.63072
14	0 Loss or Bust				0.63837	41	15	▲ 1	lemme learn ur decision bndry ;)				0.63018
15	RNGesus				0.63783	56	⋮		Jim and James Want \$ \$				0.62999

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
- Deep 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

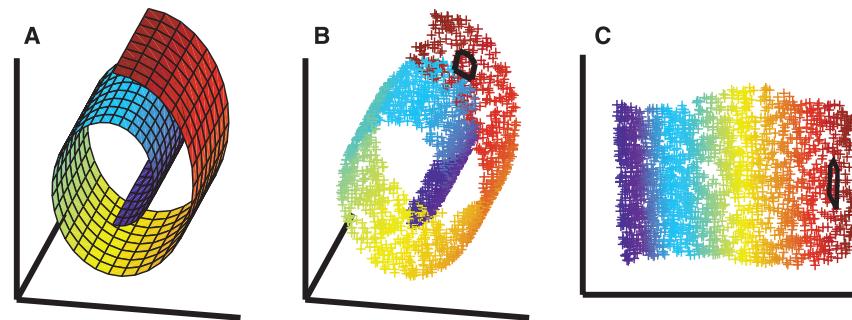
$$d(u, u') = u^T u'$$

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”

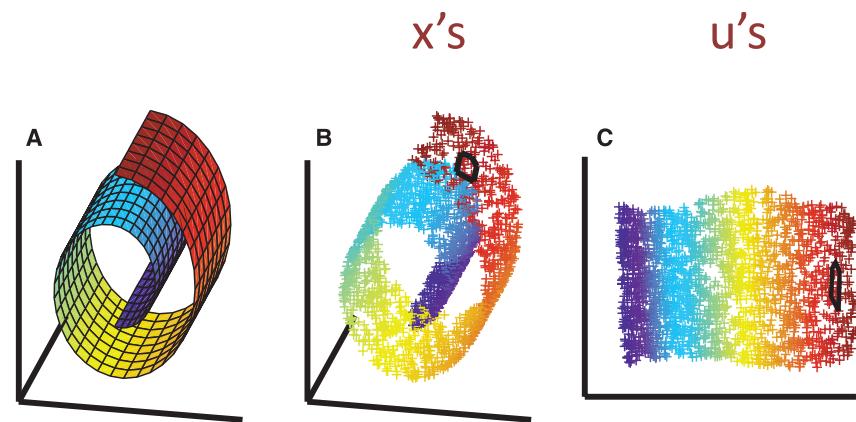
Any neighborhood
looks like a linear plane



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

Approach

- Define relationship of each x to its neighbors
- Find a lower dimensional u that preserves relationship



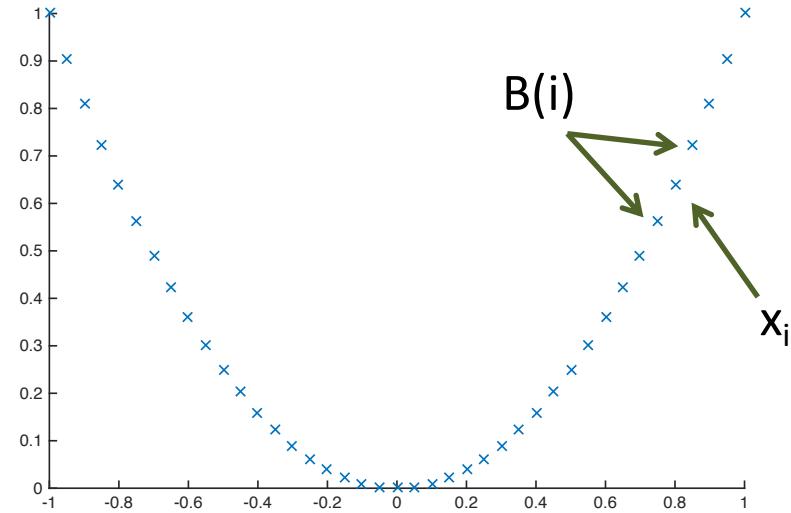
Locally Linear Embedding

- 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/>



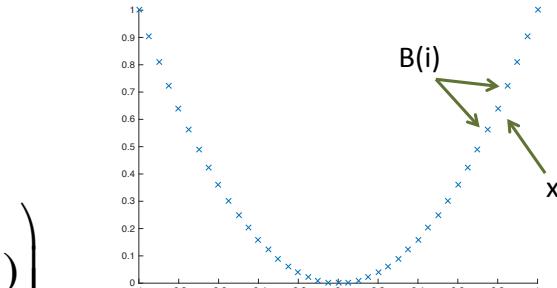
Locally Linear Embedding

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

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

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

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



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

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

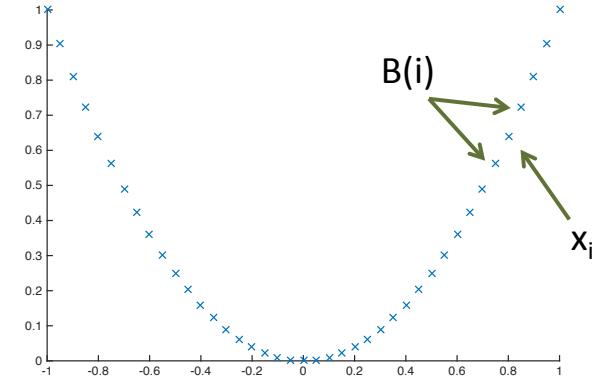
Locally Linear Embedding

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

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

$$C^i_{jk} = (x_i - x_j)^T (x_i - x_k)$$

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

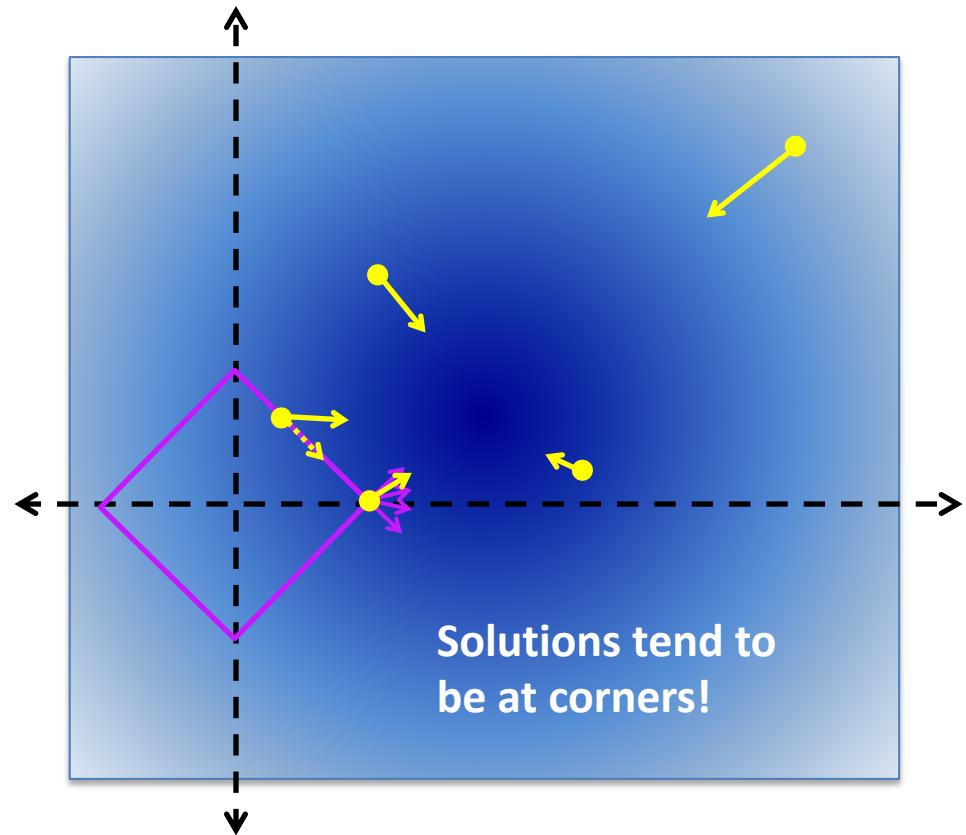


Lagrange Multipliers

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

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

$$\nabla_{w_j} |w| \left\{ \begin{array}{ll} -1 & \text{if } w_j < 0 \\ +1 & \text{if } w_j > 0 \\ [-1,+1] & \text{if } w_j = 0 \end{array} \right.$$



$$\exists \lambda \geq 0: (L(y, w) \in -\lambda \nabla_w |w|) \wedge (|w| \leq 1)$$

Solving Locally Linear Approximation

Lagrangian:

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

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

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

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

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

Locally Linear Approximation

- Invariant to:

- Rotation

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

- Scaling

$$5x_i \approx \sum_{j \in B(i)} 5W_{ij} x_j$$

- Translation

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

Story So Far: Locally Linear Embeddings

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

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

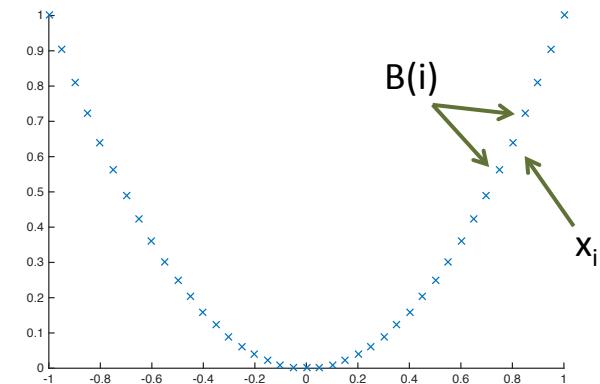
Solution via Lagrange Multipliers:

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

$$C^i_{jk} = (x_i - x_j)^T (x_i - x_k)$$

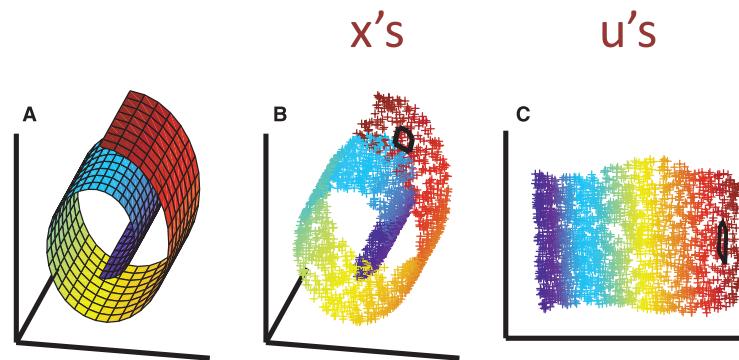
- **Locally Linear Approximation**

<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”



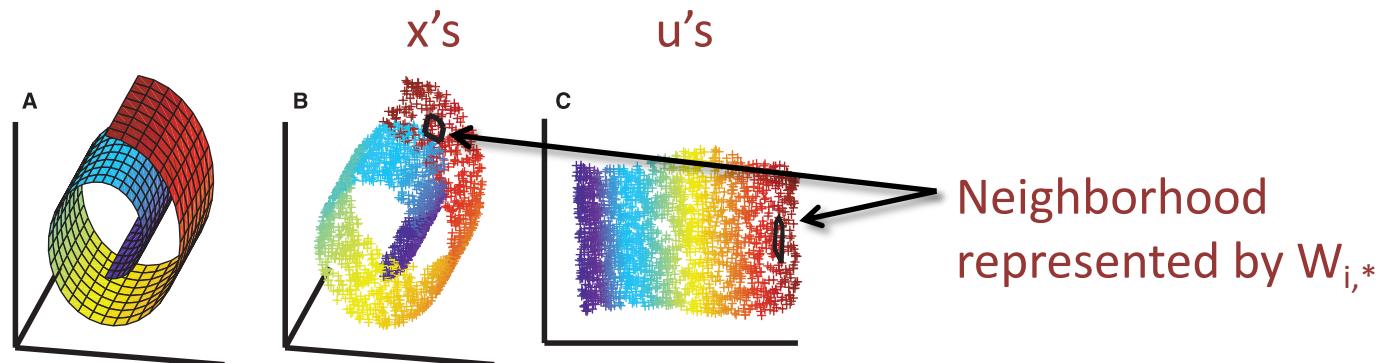
<https://www.cs.nyu.edu/~roweis/lle/>

Dimensionality Reduction (Learning the Embedding)

Given local approximation W , learn lower dimensional representation:

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

- 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:

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

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

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



Symmetric positive semidefinite

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

Given local approximation W , learn lower dimensional representation:

$$\operatorname{argmin}_U \sum_{ij} M_{ij} (u_i^T u_j) \equiv \operatorname{trace}(U M U^T) \quad U U^T = I_K$$

- Suppose $K=1$

$$\operatorname{argmin}_u \sum_{ij} M_{ij} (u_i^T u_j) \equiv \operatorname{trace}(u M u^T) \quad u u^T = 1$$

$$= \operatorname{argmax}_u \operatorname{trace}(u M^+ u^T)$$


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 ($\lambda_1 \geq \lambda_2 \geq \dots$)

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

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

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

Given local approximation W , learn lower dimensional representation:

$$\operatorname{argmin}_U \sum_{ij} M_{ij} (u_i^T u_j) = \text{trace}(U M U^T) \quad UU^T = I_K$$

- **K=1:**

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

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

- **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:

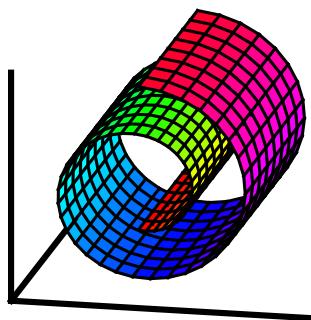
$$\operatorname{argmin}_W \sum_i \left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 \quad \sum_{j \in B(i)} W_{ij} = 1$$

- Compute low dimensional embedding

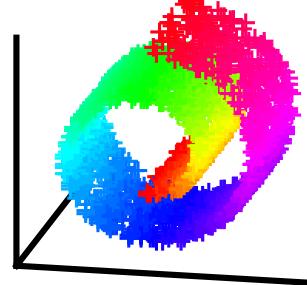
$$\operatorname{argmin}_U \sum_i \left\| u_i - \sum_{j \in B(i)} W_{ij} u_j \right\|^2 \quad \begin{aligned} UU^T &= I_K \\ \sum_i u_i &= \vec{0} \end{aligned}$$

Results for Different Neighborhoods

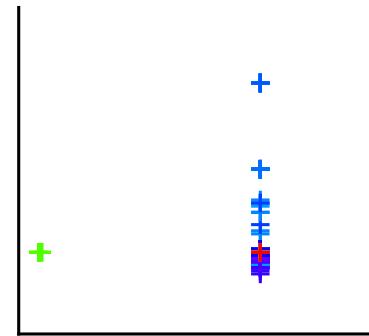
(K=2)



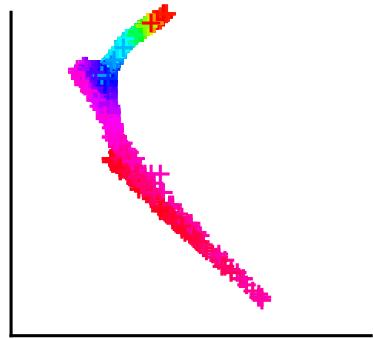
True Distribution



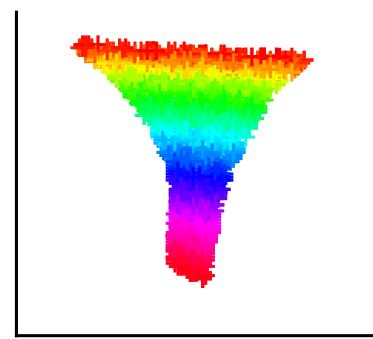
2000 Samples



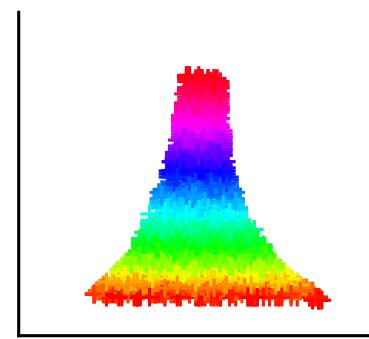
B=3



B=6



B=9

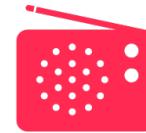


B=12

<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:

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

Songs, Words

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

Playlists, Documents

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

Sequence Definition

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

$$P(p_i^j | p_i^{j-1})$$

- 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

$P(s s')$	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)							

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

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

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

Sums over all songs

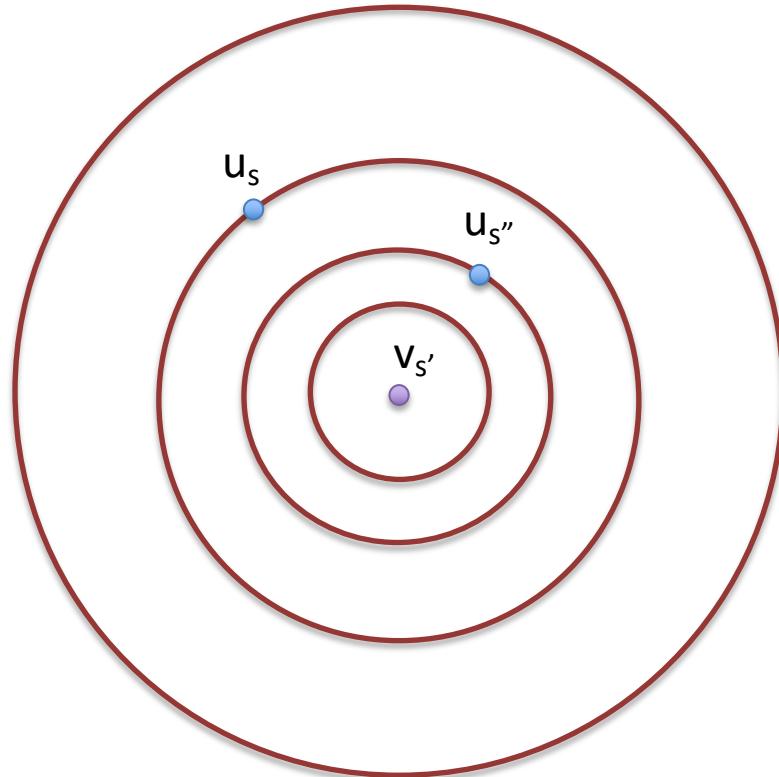
- “Log-Radial” function
 - (my own terminology)

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

Log-Radial Functions

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

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



Each ring defines an equivalence class of transition probabilities

Goals of Sequence Modeling

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

- Probabilistic transitions as “reconstruction”
- Low dimensional embedding as representation

Learning Problem

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

Songs

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

Playlists

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

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

- Learning Goal:

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

Sequences Tokens in each Sequence

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

Minimize Neg Log Likelihood

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

- Solve using gradient descent
 - Random initialization
- Normalization constant hard to compute:
 - Approximation heuristics
 - See paper



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

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

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

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

Simpler Version

- Dual point model:

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

- Single point model:

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

- Transitions are symmetric

- (almost)

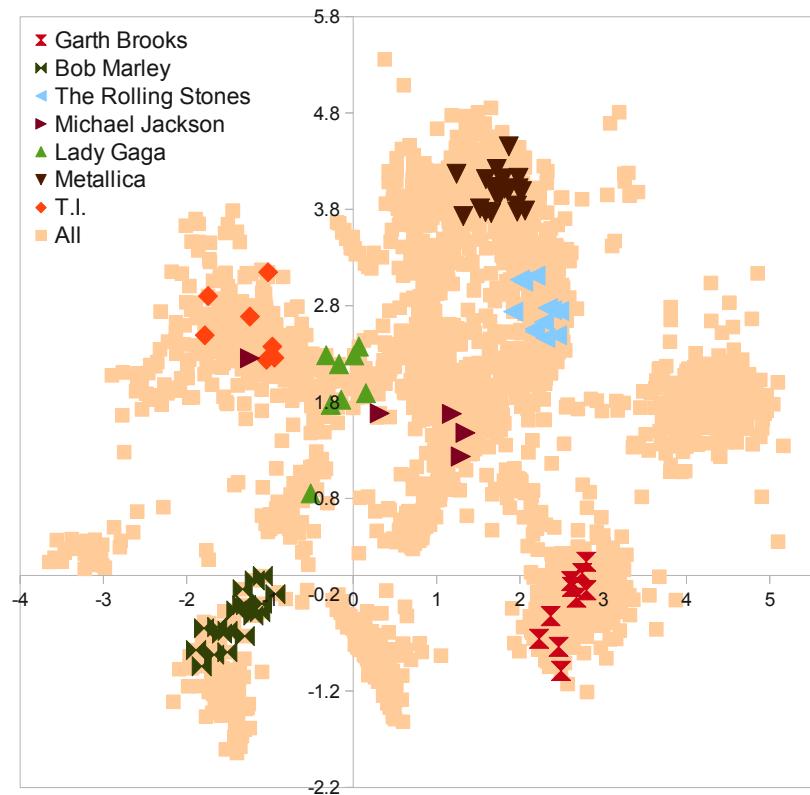
- Exact same form of training problem

Visualization in 2D

Simpler version:
Single Point Model

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

Single point model is easier to visualize



Sampling New Playlists

- Given partial playlist:

$$p = \langle p^1, \dots, p^j \rangle$$

- Generate next song for playlist p^{j+1}

- Sample according to:

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

Dual Point Model

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

Single Point Model

What About New Songs?

- Suppose we've trained U :

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

- 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!**

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

Songs

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

Playlists

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

Playlist Definition

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

Tags for Each Song

Learning Objective:

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

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

Song embedding \approx average of tag embeddings:

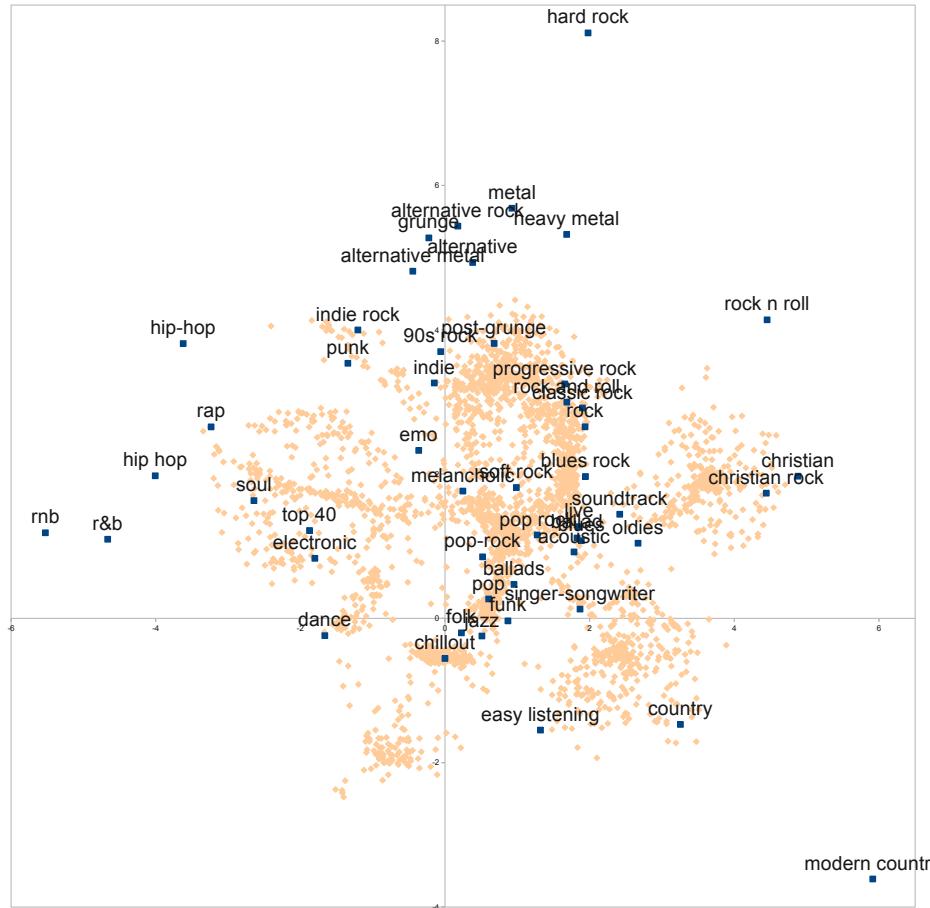
$$P(U|A, T) = \prod_s P(u_s|A, T_s) \propto \prod_s \exp \left\{ -\lambda \left\| u_s - \frac{1}{T_s} \sum_{t \in T_s} a_t \right\| \right\}$$

embedding of tag

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 $u_{s'} = \text{average of } A_t \text{ for tags } t \text{ in } T_{s'}$
 - Implication from objective:

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

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

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

- Can also capture semantics via inner product

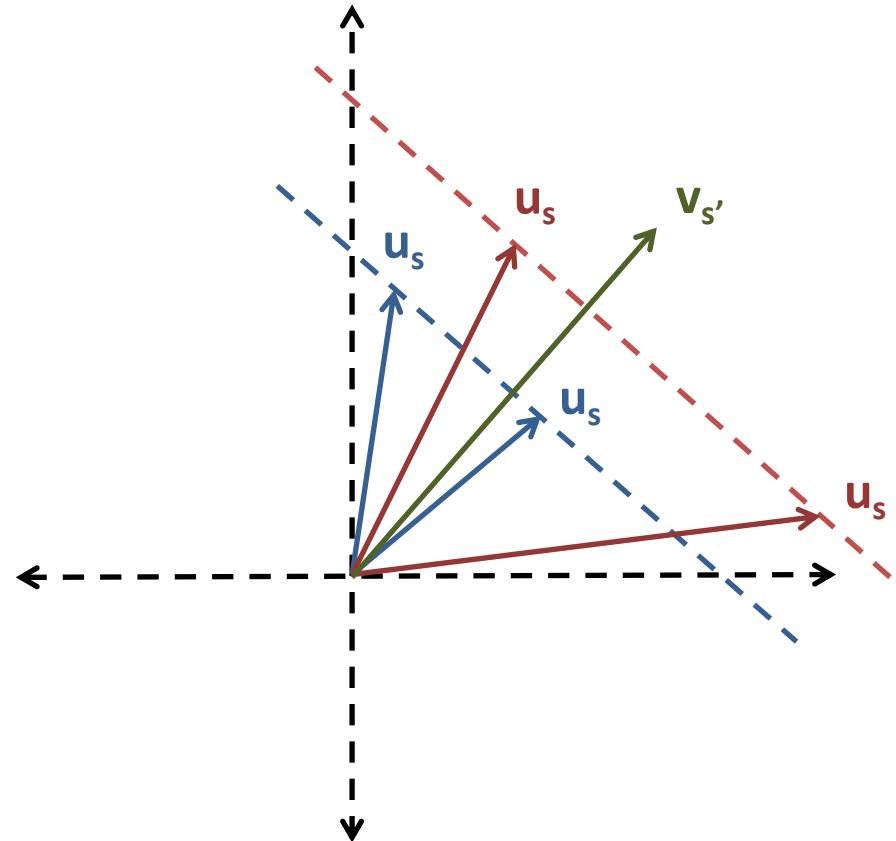
$$P(s | s') = \frac{\exp\left\{u_s^T v_{s'}\right\}}{\sum_{s''} \exp\left\{u_{s''}^T v_{s'}\right\}}$$

Basically a latent-factor model!

Log-Linear Embeddings

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

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



Each projection level onto the green line defines an equivalence class

Learning Problem (Version 1)

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

Words

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

Sentences

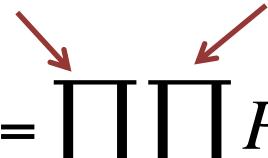
$$p_i = \langle p_i^1, \dots, p_i^{N_i} \rangle$$

Sentence Definition
(Each p_j^i is a word)

- Learning Goal:

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

Sequences Tokens in each Sequence



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

Skip-Gram Model (word2vec)

- Predict probability of any neighboring word

$$\operatorname{argmax}_{U,V} \prod_i \prod_j \prod_{k \in [-C,C] \setminus 0} P(p_i^{j+k} | p_i^j)$$

Sequences Tokens in each Sequence
↓ ↓
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

$$\operatorname{argmax}_{U,V} \prod_i \prod_j \prod_{k \in [-C,C] \setminus 0} P(p_i^{j+k} | p_i^j)$$

- 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?

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

- $K = |S|$ 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

- $\mathbf{v}_{\text{Czech}} + \mathbf{v}_{\text{currency}} \approx \mathbf{v}_{\text{koruna}}$

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

Example 2

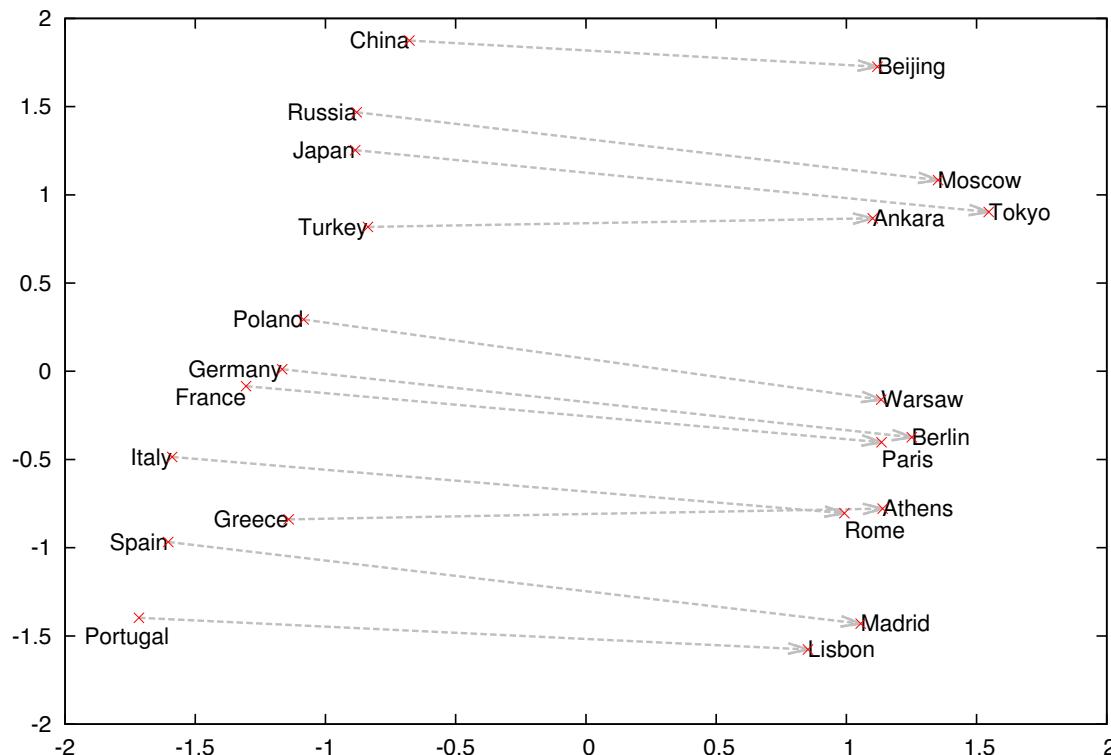
- E.g., $v_{\text{France}} - v_{\text{Paris}} + v_{\text{Italy}} \approx v_{\text{Rome}}$

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:



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

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

↑
Skip Length

Sequences Tokens in each Sequence

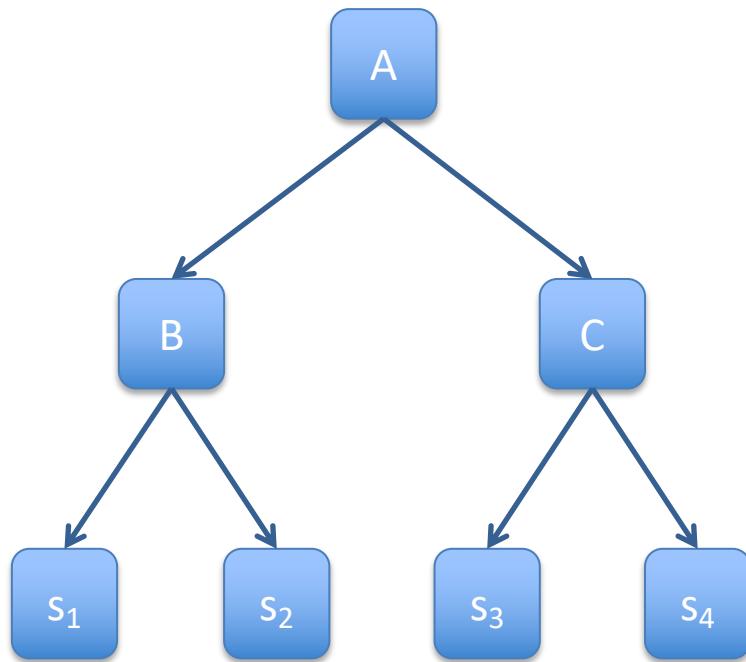
$$P(s|s') = \frac{\exp\left\{u_s^T v_{s'}\right\}}{\sum_{s''} \exp\left\{u_{s''}^T v_{s'}\right\}} = \frac{\exp\left\{u_s^T v_{s'}\right\}}{Z(s')}$$

Denominator
expensive!

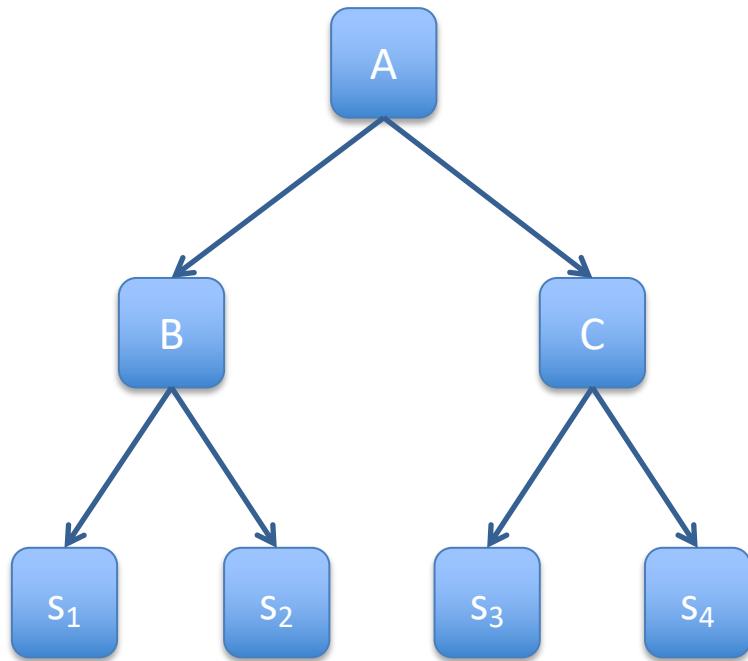
<https://code.google.com/archive/p/word2vec/>

Hierarchical Approach (Probabilistic Decision Tree)

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



Hierarchical Approach (Probabilistic Decision Tree)



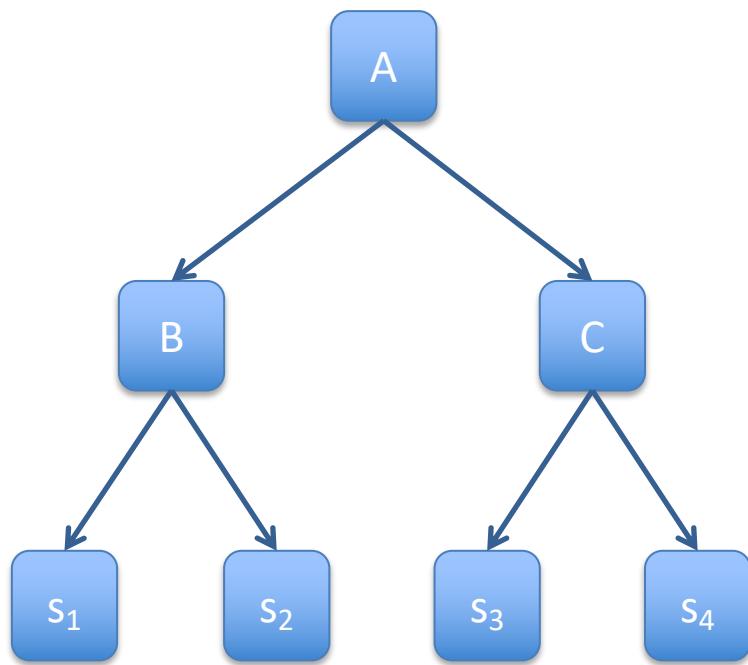
$$P(s_1 | s') = P(B | A, s')P(s_1 | B, s')$$

$$P(s_2 | s') = P(B | A, s')P(s_2 | B, s')$$

$$P(s_3 | s') = P(C | A, s')P(s_3 | C, s')$$

$$P(s_4 | s') = P(C | A, s')P(s_4 | C, s')$$

Hierarchical Approach (Probabilistic Decision Tree)

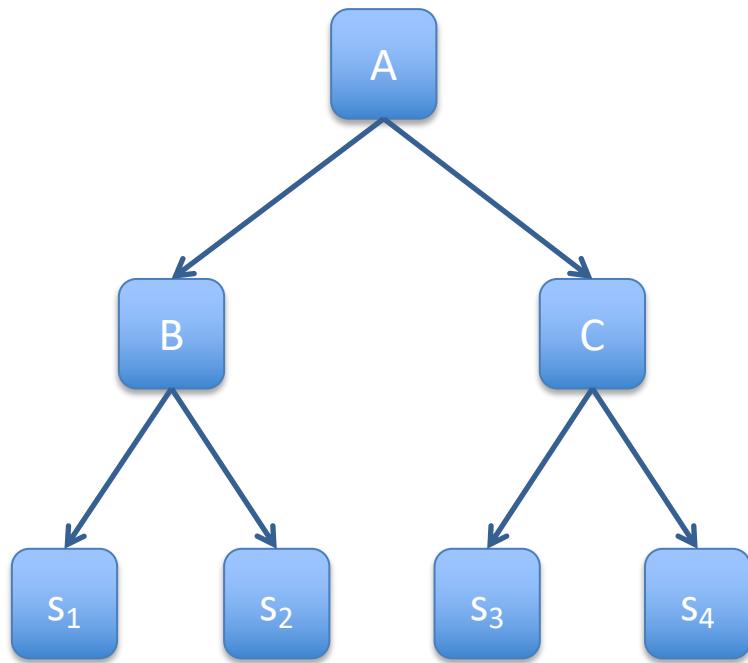


$$P(B | A, s) = \frac{1}{1 + \exp\{-u_{BC}^T v_s\}} = \frac{1}{1 + \exp\{u_{CB}^T v_s\}}$$

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

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

Hierarchical Approach (Probabilistic Decision Tree)



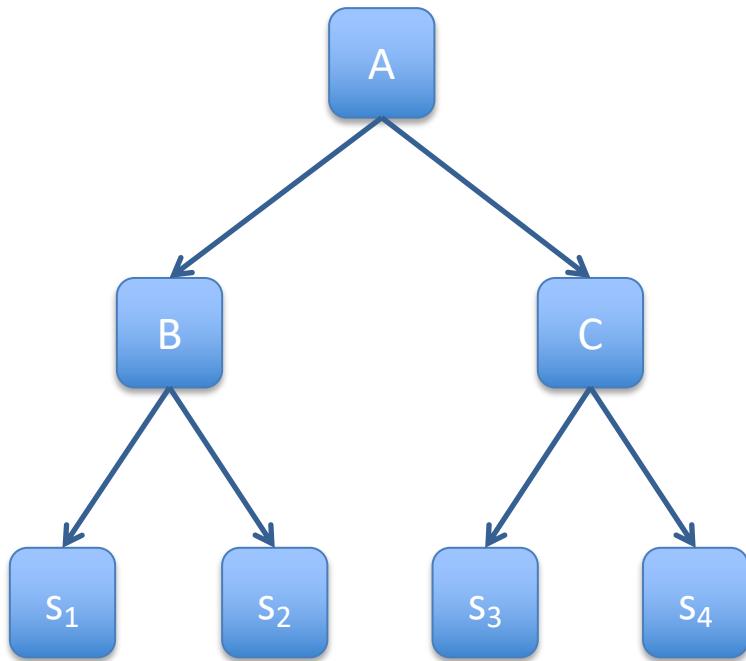
$$P(s_1 | B, s) = \frac{1}{1 + \exp\{-u_{12}^T v_s\}} = \frac{1}{1 + \exp\{u_{21}^T v_s\}}$$

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

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

Hierarchical Approach (Probabilistic Decision Tree)

- Compact formula:



Internal node at level m
on path to leaf node s

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

Levels in tree

Training Hierarchical Approach

- Train via gradient descent (same as before!)

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

↑
Skip Length

Sequences Tokens in each Sequence

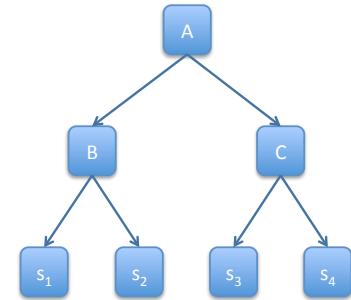
$$P(s | s') = \prod_m P(n_{m,s} | n_{m-1,s}, s)$$

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

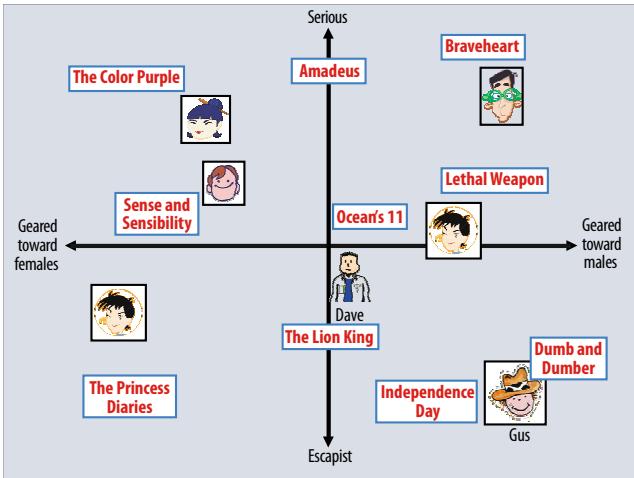
<https://code.google.com/archive/p/word2vec/>

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.



Visualization Semantics

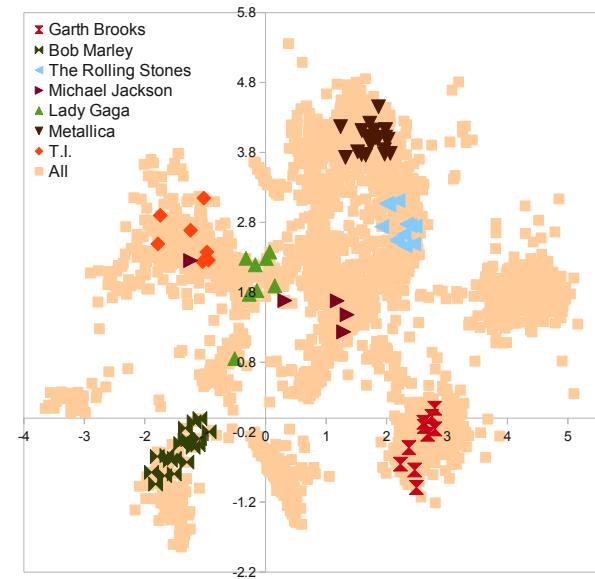


Inner-Product Embeddings

- 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



Learning Visual Style (Deep Embedding)



Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences

Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, Serge Belongie, ICCV 2015

Visually Compatible



Visually Incompatible



<http://vision.cornell.edu/se3/projects/clothing-style/>

Training Data

- Ground set of items
 - ~1M items
 - Image of item x
 - Category of item c
 - Coat, belt, pants, socks, etc.
- Pairwise relationships
 - “frequently bought together”
 - Interpret as visually compatible



Training Goal

(ignoring regularization)

$$\operatorname{argmin}_{\Theta} \sum_{(i,j) \in D} L^+ (\Phi(x_i), \Phi(x_j)) + \sum_{(i,j) \in \tilde{D}} L^- (\Phi(x_i), \Phi(x_j))$$

Penalizes too far

Penalizes too close

Embedding of image

Embedding of image

All Model Parameters

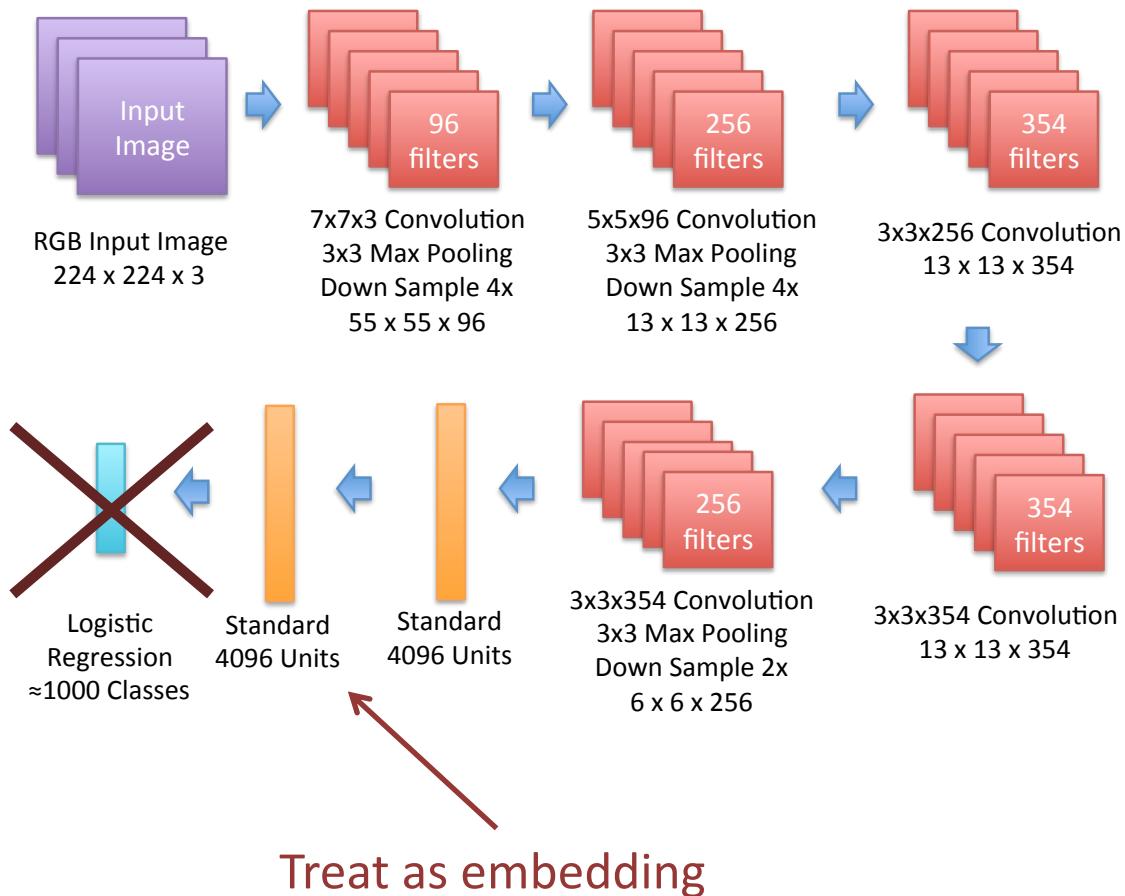
Compatible Pairs

Incompatible Pairs

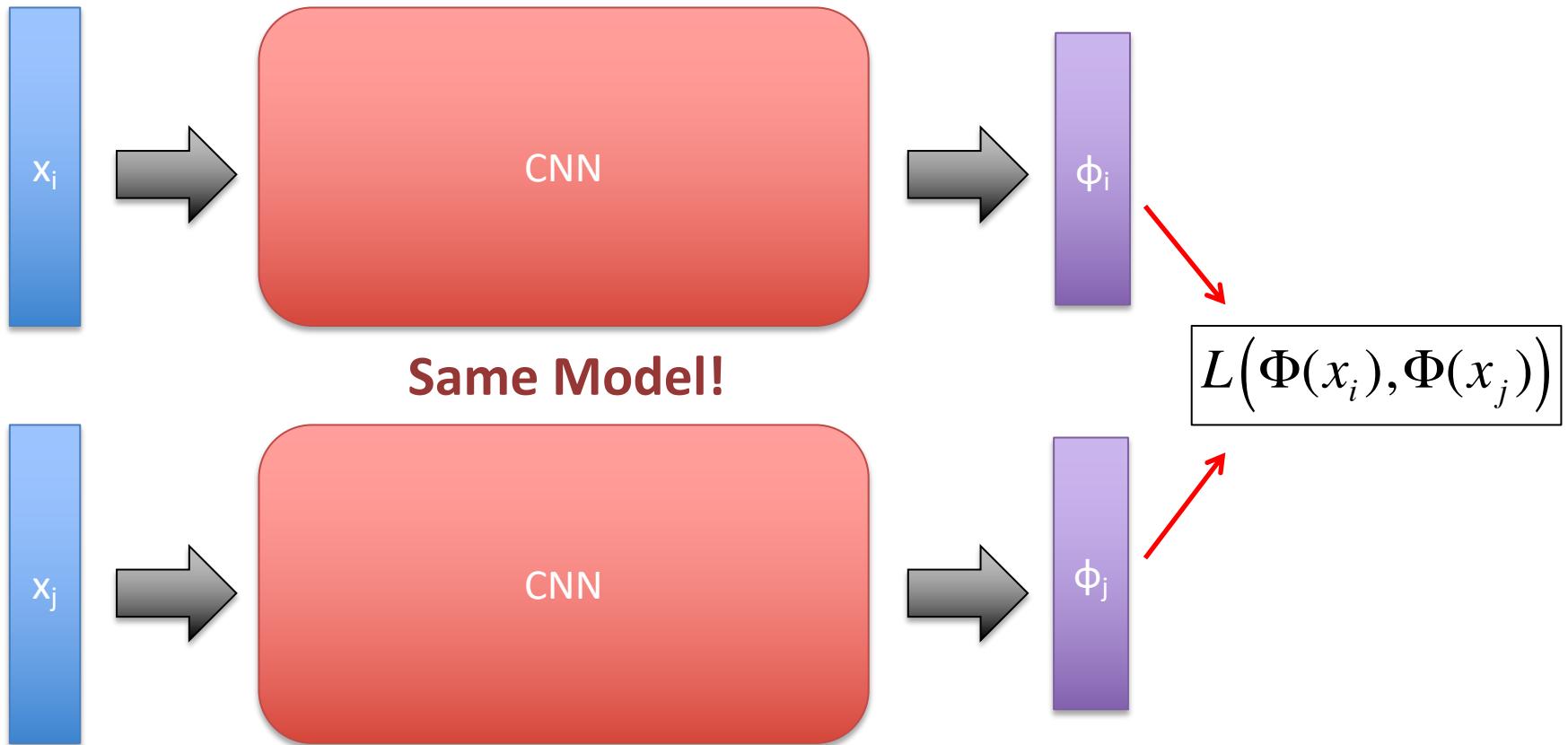
Only pairs in different categories.

The diagram illustrates the components of the training goal equation. At the top center is the equation: $\operatorname{argmin}_{\Theta} \sum_{(i,j) \in D} L^+ (\Phi(x_i), \Phi(x_j)) + \sum_{(i,j) \in \tilde{D}} L^- (\Phi(x_i), \Phi(x_j))$. Above the first sum, the text "Penalizes too far" has a red arrow pointing down to the term $L^+ (\Phi(x_i), \Phi(x_j))$. Above the second sum, the text "Penalizes too close" has a red arrow pointing down to the term $L^- (\Phi(x_i), \Phi(x_j))$. Below the first sum, the text "Compatible Pairs" has a red arrow pointing up to the term $L^+ (\Phi(x_i), \Phi(x_j))$. Below the second sum, the text "Incompatible Pairs" has a red arrow pointing up to the term $L^- (\Phi(x_i), \Phi(x_j))$. To the left of the equation, "All Model Parameters" has a red arrow pointing up to the Θ symbol. Below the equation, "Only pairs in different categories." has a blue arrow pointing up to the condition $(i,j) \in D$ and $(i,j) \in \tilde{D}$.

Recall: Convolutional Neural Networks

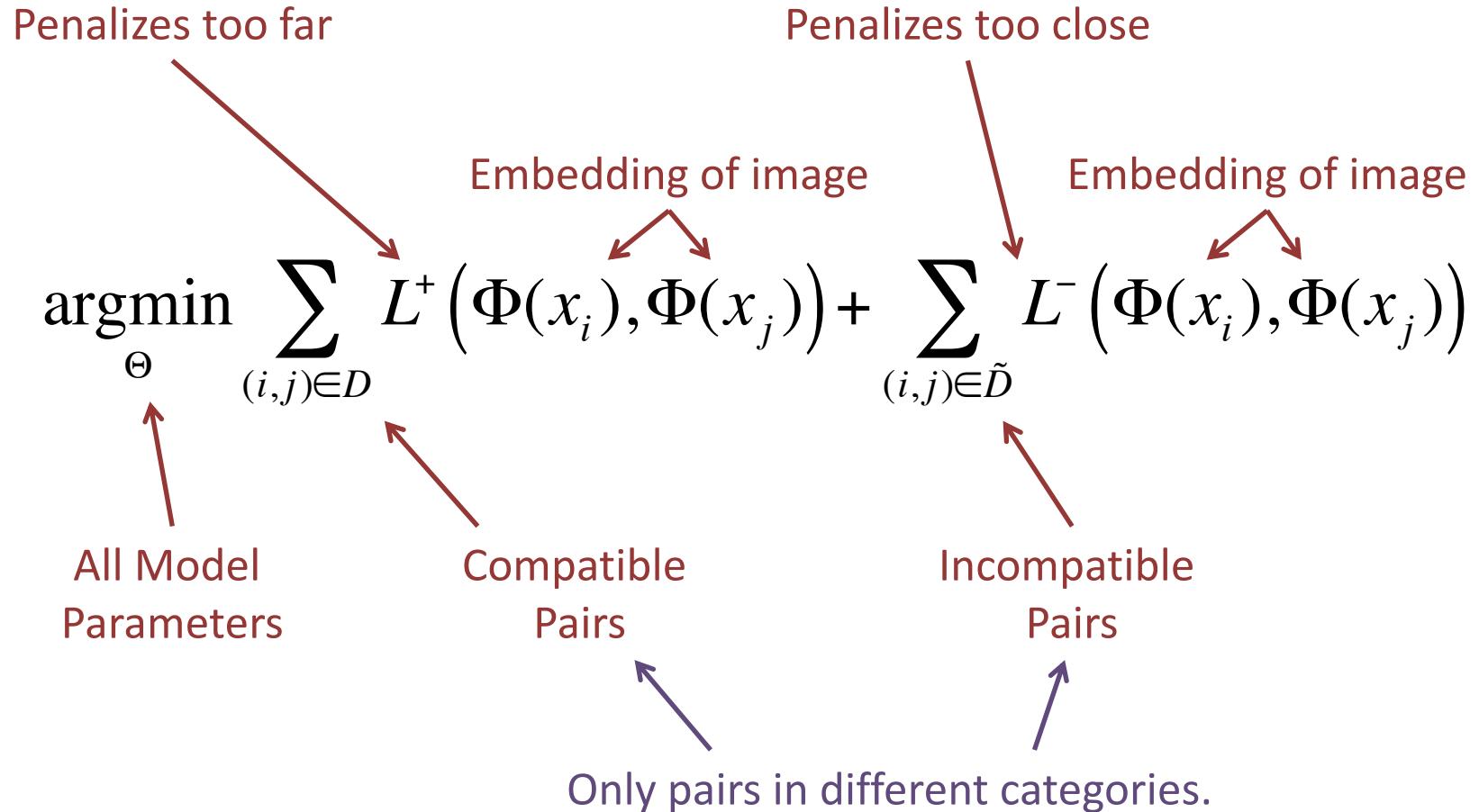


Siamese Convolutional Neural Networks



More details: <http://www.cs.cornell.edu/~kb/publications/SIG15ProductNet.pdf>

Recap: Training Goal



Model Embedding via Siamese Convolutional Neural Network!

Lecture 11: Embeddings

70

Training Details

- Want embedding dimension smaller
 - E.g., 128 rather than 4096
- Need to subsample negative pairs
 - Most items are not frequently bought together
 - Negative component can overwhelm objective



<http://www.cs.cornell.edu/~andreas/iccv15.pdf>

Lecture 11: Embeddings

72

Suggesting Outfits

Upper
Garment



Lower
Garment

Footware

Recap: Embeddings

- **Given:** Training Data
 - Care about some property of training data
 - Markov Chain
 - Skip-Gram
 - Affinity to be purchased together
- **Goal:** learn low dim representation
 - “Embedding”
 - Geometry of embedding captures property of interest
 - Either by distance or by inner-product
- Very powerful tool for representation learning
 - Used for downstream supervised learning tasks

Next Lectures

- Tuesday: Probabilistic Models & Naïve Bayes
- Thursday: Cancelled
 - Check out Emerging Neuroscience Symposium!
 - <https://neuroscience.caltech.edu/events/87895>
- Thursday Recitation: Probability