Lecture 11: Embeddings
## Kaggle Competition (Part 1)

<table>
<thead>
<tr>
<th>#</th>
<th>Team Name</th>
<th>Team Members</th>
<th>Score</th>
<th>Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jim Wants Job Offers</td>
<td></td>
<td>0.81531</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>Xiaoqiao Chen</td>
<td></td>
<td>0.81399</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Ohio</td>
<td></td>
<td>0.81174</td>
<td>46</td>
</tr>
<tr>
<td>4</td>
<td>IDontHaveATeamName</td>
<td></td>
<td>0.81076</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>HeMaSu</td>
<td></td>
<td>0.81005</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Sidchastic Guthient Dingscent</td>
<td></td>
<td>0.80831</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>C_grade_benchmark</td>
<td></td>
<td>0.80784</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>TaKiNg bErRiEs fRoM OpEn kl...</td>
<td></td>
<td>0.80757</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>2nd_attempt_by_TAs_2008.csv</td>
<td></td>
<td>0.80744</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Bitsplease</td>
<td></td>
<td>0.80731</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>NaN</td>
<td></td>
<td>0.80636</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>The Richard Priors</td>
<td></td>
<td>0.80620</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>Graeters Sucks</td>
<td></td>
<td>0.80580</td>
<td>35</td>
</tr>
<tr>
<td>13</td>
<td>Pathological</td>
<td></td>
<td>0.80578</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>Party Trio</td>
<td></td>
<td>0.80548</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>Working...</td>
<td></td>
<td>0.80496</td>
<td>18</td>
</tr>
<tr>
<td>16</td>
<td>Unif(0, 1)</td>
<td></td>
<td>0.80463</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Team Name</th>
<th>Team Members</th>
<th>Score</th>
<th>Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jim Wants Job Offers</td>
<td></td>
<td>0.80685</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Xiaoqiao Chen</td>
<td></td>
<td>0.80644</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>IDontHaveATeamName</td>
<td></td>
<td>0.80375</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>HeMaSu</td>
<td></td>
<td>0.80159</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sidchastic Guthient Dingscent</td>
<td></td>
<td>0.80144</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Bitsplease</td>
<td></td>
<td>0.80140</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2nd_attempt_by_TAs_2008.csv</td>
<td></td>
<td>0.80105</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Ohio</td>
<td></td>
<td>0.80086</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Graeters Sucks</td>
<td></td>
<td>0.79335</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Unif(0, 1)</td>
<td></td>
<td>0.79876</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>C_grade_benchmark</td>
<td></td>
<td>0.79870</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>NaN</td>
<td></td>
<td>0.79853</td>
<td></td>
</tr>
<tr>
<td></td>
<td>prediction_by_TA_2008.csv</td>
<td></td>
<td>0.79761</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Pathological</td>
<td></td>
<td>0.79752</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>NULL</td>
<td></td>
<td>0.79729</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>The Richard Priors</td>
<td></td>
<td>0.79728</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Ss1996</td>
<td></td>
<td>0.79692</td>
<td></td>
</tr>
</tbody>
</table>
# Kaggle Competition (Part 2)

<table>
<thead>
<tr>
<th>#</th>
<th>Team Name</th>
<th>Kernel</th>
<th>Team Members</th>
<th>Score</th>
<th>Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NaN</td>
<td></td>
<td></td>
<td>0.7915</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Graeters Sucks</td>
<td></td>
<td></td>
<td>0.7903</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2nd_attempt_by_TAs_2012.csv</td>
<td></td>
<td></td>
<td>0.7913</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Unif(0,1)</td>
<td></td>
<td></td>
<td>0.7888</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Working...</td>
<td></td>
<td></td>
<td>0.7887</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Sidchastic Guthient Dinscent</td>
<td></td>
<td></td>
<td>0.7886</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>NULL</td>
<td></td>
<td></td>
<td>0.7882</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>Xiaoqiao Chen</td>
<td></td>
<td></td>
<td>0.7880</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>Bitsplease</td>
<td></td>
<td></td>
<td>0.7792</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>NULL</td>
<td></td>
<td></td>
<td>0.7879</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>The Richard Priors</td>
<td></td>
<td></td>
<td>0.7878</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>Albert Zhai</td>
<td></td>
<td></td>
<td>0.7873</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>TaKiNg bErRiEs fRoM OpEn kl</td>
<td></td>
<td></td>
<td>0.7866</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>HeMaSu</td>
<td></td>
<td></td>
<td>0.7864</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>send_nodes</td>
<td></td>
<td></td>
<td>0.7864</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>hot_as_asphalt</td>
<td></td>
<td></td>
<td>0.7862</td>
<td>1</td>
</tr>
</tbody>
</table>
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: \( S = \{x_i\}_{i=1}^{N} \)

- 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 $W$:

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

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

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

$$= \left( \sum_{j \in B(i)} W_{ij} (x_i - x_j) \right)^T \left( \sum_{j \in B(i)} W_{ij} (x_i - x_j) \right)$$

$$= \sum_{j \in B(i)} \sum_{k \in B(i)} W_{ij} W_{ik} C_{jk}^i$$

$$= W_{i,*}^T C^i W_{i,*}$$

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

https://www.cs.nyu.edu/~roweis/lle/
Locally Linear Embedding

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

$$\arg\min_W \sum_i \left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 = \arg\min_W \sum_i W_i^T C_i W_i^* \quad \sum_{j \in B(i)} W_{ij} = 1$$

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

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

\[ \text{argmin } L(w) \equiv w^T C w \]

s.t. \( |w| = 1 \)

\( \nabla_w |w| = \begin{cases} 
-1 & \text{if } w_j < 0 \\
+1 & \text{if } w_j > 0 \\
[-1, +1] & \text{if } w_j = 0 
\end{cases} \)

\[ \exists \lambda \geq 0: (L(y, w) \in -\lambda \nabla_w |w|) \land (|w| \leq 1) \]

Solutions tend to be at corners!
Solving Locally Linear Approximation

Lagrangian:

\[ L(W, \lambda) = \sum_i \left( W_i^T C^i W_i, - \lambda_i \left( \bar{1}^T W_i, - 1 \right) \right) \]

\[ \sum W_{ij} = \bar{1}^T W_i, \]

\[ \partial_{W_i,} L(W, \lambda) = 2 C^i W_i, - \lambda_i \bar{1} \]

\[ W_{i,} = \frac{\lambda_i}{2} \left( C^i \right)^{-1} \bar{1} \alpha \left( C^i \right)^{-1} \bar{1} \]

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

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

- Invariant to:
  - Rotation
    \[ Ax_i \approx \sum_{j \in B(i)} AW_{ij}x_j \]
    \[ \sum_{j \in B(i)} W_{ij} = 1 \]
  - 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 $W$:

$$\arg\min_W \sum_i \left\| x_i - \sum_{j \in B(i)} W_{ij} x_j \right\|^2 = \arg\min_W \sum_i W_i^T C_i W_i^*$$

Solution via Lagrange Multipliers:

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

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

- 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:

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

- Find low dimensional $U$
  - Preserves approximate local linearity

https://www.cs.nyu.edu/~roweis/lle/
Given local approximation $W$, learn lower dimensional representation:

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

$$UU^T = I_K$$

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

• Rewrite as:

$$\arg\min_U \sum_{ij} M_{ij} (u_i^T u_j) \equiv \text{trace}(UMU^T)$$

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

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

Symmetric positive semidefinite

https://www.cs.nyu.edu/~roweis/lle/
Given local approximation W, learn lower dimensional representation:

\[
\arg\min_U \sum_{ij} M_{ij} (u_i^T u_j) \equiv \text{trace}(U M U^T) = 1
\]

- Suppose \( K=1 \)

\[
\arg\max_U \sum_{ij} M_{ij} (u_i^T u_j) \equiv \text{trace}(u M u^T) = \text{tr}(u M^+ u^T)
\]

\[
U U^T = I_K
\]

\[
\sum_i u_i = \vec{0}
\]

\[
u u^T = 1
\]

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

http://en.wikipedia.org/wiki/Min-max_theorem
Recap: Principal Component Analysis

\[ M = V \Lambda V^T \]

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

\[ M^+ = V \Lambda^+ V^T \]

\[ MM^+ = V \Lambda \Lambda^+ V^T = V_{1:2} V_{1:2}^T = \begin{bmatrix} 1 & \frac{1}{\lambda_1} \\ 0 & 0 \end{bmatrix} \]
Given local approximation \( W \), learn lower dimensional representation:

\[
\arg\min_U \sum_{ij} M_{ij} (u_i^T u_j) \equiv \text{trace} \left( U M U^T \right)
\]

\[
UU^T = I_K
\]

\[
\sum_i u_i = \bar{0}
\]

- **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:

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

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

• Compute low dimensional embedding

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

$$UU^T = I_K$$

$$\sum_i u_i = \bar{0}$$
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:**

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

Songs, Words \quad Playlists, Documents \quad 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') | s₁   | s₂   | s₃   | s₄   | s₅   | s₆   | s₇   | s_{start} |
|--------|------|------|------|------|------|------|------|-----------|
| s₁     | 0.01 | 0.03 | 0.01 | 0.11 | 0.04 | 0.04 | 0.01 | 0.05      |
| s₂     | 0.03 | 0.01 | 0.04 | 0.03 | 0.02 | 0.01 | 0.02 | 0.02      |
| s₃     | 0.01 | 0.01 | 0.01 | 0.07 | 0.02 | 0.02 | 0.05 | 0.09      |
| s₄     | 0.02 | 0.11 | 0.07 | 0.01 | 0.07 | 0.04 | 0.01 | 0.01      |
| s₅     | 0.04 | 0.01 | 0.02 | 0.17 | 0.01 | 0.01 | 0.10 | 0.02      |
| s₆     | 0.01 | 0.02 | 0.03 | 0.01 | 0.01 | 0.01 | 0.01 | 0.08      |
| s₇     | 0.07 | 0.02 | 0.01 | 0.01 | 0.03 | 0.09 | 0.03 | 0.01      |
First Try: Probability Tables

| P(s|s') | s₁  | s₂  | s₃  | s₄  | s₅  | s₆  | s₇  | s_start |
|--------|-----|-----|-----|-----|-----|-----|-----|---------|
| s₁     | 0.01| 0.03| 0.01| 0.11| 0.04| 0.04| 0.01| 0.05    |
| s₂     | 0.03| 0.01| 0.04| 0.03| 0.02| 0.01| 0.02| 0.02    |
| s₃     | 0.01| 0.01| 0.01| 0.07| 0.02| 0.02| 0.05| 0.09    |
| s₄     | 0.02| 0.11| 0.07| 0.01| 0.07| 0.04| 0.01| 0.01    |

#Parameters = O(|S|^2) !!!
(worse for higher-order sequence models)
Outline for Sequence Modeling

• Playlist Embedding
  – Distance-based embedding

• Word Embedding (word2vec)
  – Inner-product embedding
  – https://code.google.com/archive/p/word2vec/

• Compare the two approaches

Homework Question!
Markov Embedding (Distance)

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

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

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

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

Sums over all songs

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

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

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

\[ S = \{s_1, \ldots, s_{|S|}\} \]

Songs

\[ D = \{p_i\}_{i=1}^N \]

Playlists

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

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

**Learning Goal:**

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

Sequences

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

Tokens in each Sequence

Minimize Neg Log Likelihood

\[
\arg\max_{U,V} \prod_i \prod_j P(p_i^j \mid p_i^{j-1}) = \arg\min_{U,V} \sum_i \sum_j -\log P(p_i^j \mid p_i^{j-1})
\]

- Solve using gradient descent
  - Random initialization

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

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

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

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

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

Single point model is easier to visualize

Sampling New Playlists

• Given partial playlist:

\[ p = \langle p^1, \ldots, p^j \rangle \]

• Generate next song for playlist \( p^{j+1} \)
  – Sample according to:

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

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

Dual Point Model  Single Point Model

What About New Songs?

• Suppose we’ve trained $U$:

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

• 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!

Learning Objective:
\[
\arg\max_{U,A} P(D \mid U)P(U \mid A,T)
\]

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 \( \approx \) average of tag embeddings:
\[
P(U \mid A, T) = \prod_{s} P(u_s \mid A, T_s) \propto \prod_{s} \exp\left\{-\lambda \left\| u_s - \frac{1}{T_s} \sum_{t \in T_s} a_t \right\|\right\}
\]

Solve using gradient descent:
The simplest NLP model is the Unigram Model. In particular, we explore to what extent the unigram model on the baselines is even larger for songs to choose from, while the LME shows no signs of overfitting. In more detail where the conventional bigram model fails, the number of parameters in the bigram model scales quadratically with the number of songs, while it scales only linearly in the LME model. The following section analyzes the unigram model on the test set of the baselines on the top 40 songs. The probability of each song is sampled independently of the previous songs. The probability of the next song is estimated from the training set as \( P(s_{n+1} | s_n) \), where each song has equal probability \( P(s) = \frac{1}{|S|} \). As a reference, we also report the results for the uni-gram model. Among the conventional sequence models, the bigram model performs best on the baselines. The x-axis of Model Figure 3 shows the dimensionality (left) and (right). However, the transition probabilities \( p(s_{n+1} | s_n) \) that the number of parameters in the bigram model scales with \( s_{n+1} \) similar to our LME model. The tag-based model from Section 3.2 performs considerably to the results in Figure 2. For datasets with less parabolically cover the space, but forms clusters as expected. The location of the tags provides interesting insight into the semantics of these clusters. Note that semantically synonymous tags are typically close in embedding space (e.g. “rock” and “metal”). Note that some tags lie outside the anonymous tags are labeled; lighter points represent songs.

Visualization in 2D
Revisited: What About New Songs?

• No user has s’ added to playlist
  – So no evidence from playlist training data:

  \[ s' \text{ does not appear in } D = \left\{ p_i \right\}_{i=1}^N \]

• 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:

\[
\arg\max_{U,A} P(D \mid U)P(U \mid A,T)
\]
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
  \[ P(s \mid s') = \frac{\exp\left\{-\|u_s - v_{s'}\|^2\right\}}{\sum_{s''} \exp\left\{-\|u_{s''} - v_{s'}\|^2\right\}} \]

- Can also capture semantics via inner product
  \[ P(s \mid s') = \frac{\exp\left\{u_s^T v_{s'}\right\}}{\sum_{s''} \exp\left\{u_{s''}^T v_{s'}\right\}} \]

Basically a latent-factor model!
Log-Linear Embeddings

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

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

Each projection level onto the green line defines an equivalence class
Learning Problem (Version 1)

\[ S = \{ s_1, ..., s_{|S|} \} \]
Words

\[ D = \{ p_i \}_{i=1}^N \]
Sentences

\[ p_i = \langle p_i^1, ..., p_i^{N_i} \rangle \]
Sentence Definition
(Each \( p_i \) is a word)

\[ \arg\max_{U,V} \prod_i P(p_i) = \prod_i \prod_j P(p_i^j | p_i^{j-1}) \]

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

Tokens in each Sequence
Skip-Gram Model (word2vec)

• Predict probability of any neighboring word

\[
\arg\max_{U,V} \prod_i \prod_j \prod_{k \in [-C,C]\setminus 0} P(p_i^{j+k} \mid p_i^j)
\]

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

https://code.google.com/archive/p/word2vec/
Skip-Gram Model (word2vec)

- Predict probability of any neighboring word

\[
\text{argmax}_{U,V} \prod_i \prod_j \prod_{k \in [-C,C] \setminus 0} P(p_{i+k}^j \mid p_i^j)
\]

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.”

$\arg\max_{U,V} \prod_i \prod_j \prod_{k \in [-C,C] \setminus 0} P(p_{i+k}^j \mid p_i^j)$

• 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| 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

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

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


Lecture 11: Embeddings
Example 2

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

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

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:

\[ h(x) = \text{sign}(w^T \phi(x)) \]

Can be word identities or word2vec representation!
Training word2vec

- Train via gradient descent

\[
\text{argmin}_{U,V} \sum_{i} \sum_{j} \sum_{k \in [-C,C] \setminus 0} -\log P(p_{i}^{j+k} \mid p_{i}^{j})
\]

Sequences

Tokens in each Sequence

Skip Length

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

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)

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

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

\[u_{BC} = -u_{CB}\]
Hierarchical Approach
(Probabilistic Decision Tree)

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

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

\[ u_{12} = -u_{21} \]
Hierarchical Approach
(Probabilistic Decision Tree)

• Compact formula:

\[ P(s \mid s') = \prod_{m} P(n_{m,s} \mid n_{m-1,s}, s) \]

Levels in tree

Internal node at level \( m \) on path to leaf node \( s \)
Training Hierarchical Approach

- Train via gradient descent (same as before!)

\[
\arg\min_{U,V} \sum_i \sum_j \sum_{k \in [-C,C] \setminus 0} -\log P(p_{i}^{j+k} | p_{i}^{j})
\]

Sequences

Tokens in each Sequence

Skip Length

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

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.
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 Lectures

• Thursday: Cancelled

• Next Week:
  – Recent Applications
  – Probabilistic Models
  – Recitation on Probability