Announcements

• Assignment 2 fixes (tf-idf vs. BOW)
• NLP talks in the department!
  • **Oct 13th, 12pm Panel/Discussion** Strong vs. Weak Compositionality in Humans and Machines!
Topics

• More Followup on Word Embeddings from SVD
• Logistic Regression and Gradient Descent
• Multilayer Perceptrons
• Word Embeddings from NNs
Topics

- More Followup on Word Embeddings from SVD
- Logistic Regression and Gradient Descent
- Multilayer Perceptrons
- Word Embeddings from NNs
SVD Revisited

The below figure shows the following: (Part of) a term-document matrix $M$; A $V$ matrix that results when running LSA on $M$; An embedding (i.e., row of the $U$ matrix) associated with a document $d$. Which of the below represents the most likely content of document $d$? (Note that document $d$ is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of $M$. You can assume $d$ is a different document that also occurred in $M$ but is not shown in the below figure.)

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
<th>apple</th>
<th>kiwi</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$$M = \begin{bmatrix} 10 & 1 & 0 \\ 8 & 0 & 8 \\ 1 & 6 & 1 \\ -1 & 7 & 11 \end{bmatrix}$$

$$V = \begin{bmatrix} 1 & 2 & 12 \end{bmatrix}$$
The below figure shows the following: (Part of) a term-document matrix $M$; A $V$ matrix that results when running LSA on $M$; An embedding (i.e., row of the $U$ matrix) associated with a document $d$. Which of the below represents the most likely content of document $d$? (Note that document $d$ is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of $M$. You can assume $d$ is a different document that also occurred in $M$ but is not shown in the below figure.)

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
<th>apple</th>
<th>kiwi</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$$M = \begin{pmatrix}
10 & 1 & 0 \\
8 & 0 & 8 \\
1 & 6 & 1 \\
-1 & 7 & 11 \\
\end{pmatrix}$$

$$V = \begin{pmatrix}
1 & 2 & 12 \\
\end{pmatrix}$$
SVD Revisited

The below figure shows the following: (Part of) a term-document matrix \( M \); A \( V \) matrix that results when running LSA on \( M \); An embedding (i.e., row of the \( U \) matrix) associated with a document \( d \). Which of the below represents the most likely content of document \( d \)? (Note that document \( d \) is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of \( M \). You can assume \( d \) is a different document that also occurred in \( M \) but is not shown in the below figure.)

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
<th>apple</th>
<th>kiwi</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ M \]

\[
\begin{pmatrix}
10 & 1 & 0 \\
8 & 0 & 8 \\
1 & 6 & 1 \\
-1 & 7 & 11
\end{pmatrix}
\]

\[ V \]

\[ d \]

\[
\begin{pmatrix}
1 & 2 & 12
\end{pmatrix}
\]
SVD Revisited

The below figure shows the following: (Part of) a term-document matrix \( M \); A \( V \) matrix that results when running LSA on \( M \); An embedding (i.e., row of the \( U \) matrix) associated with a document \( d \). Which of the below represents the most likely content of document \( d \)? (Note that document \( d \) is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of \( M \). You can assume \( d \) is a different document that also occurred in \( M \) but is not shown in the below figure.)

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
<th>apple</th>
<th>kiwi</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{pmatrix}
10 & 1 & 0 \\
8 & 0 & 8 \\
1 & 6 & 1 \\
-1 & 7 & 11
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 2 & 12 \\
\end{pmatrix}
\]
SVD Revisited

The below figure shows the following: (Part of) a term-document matrix $M$; A $V$ matrix that results when running LSA on $M$; An embedding (i.e., row of the $U$ matrix) associated with a document $d$. Which of the below represents the most likely content of document $d$? (Note that document $d$ is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of $M$. You can assume $d$ is a different document that also occurred in $M$ but is not shown in the below figure.)
SVD Revisited

The below figure shows the following: (Part of) a term-document matrix $M$; A $V$ matrix that results when running LSA on $M$; An embedding (i.e., row of the $U$ matrix) associated with a document $d$. Which of the below represents the most likely content of document $d$? (Note that document $d$ is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of $M$. You can assume $d$ is a different document that also occurred in $M$ but is not shown in the below figure.)

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
<th>apple</th>
<th>kiwi</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$$
\begin{array}{|c|c|c|c|}
\hline
\text{red} & \text{green} & \text{apple} & \text{kiwi} \\
\hline
1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\hline
\end{array}
\quad
\begin{array}{|c|c|c|}
10 & 1 & 0 \\
8 & 0 & 8 \\
1 & 6 & 1 \\
-1 & 7 & 11 \\
\end{array}
\quad
\begin{array}{|c|c|c|}
\text{d} & 1 & 2 & 12 \\
\end{array}
$$
SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.)

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>green</th>
<th>apple</th>
<th>kiwi</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ M = \begin{pmatrix}
  1 & 1 & 1 & 1 \\
  1 & 0 & 1 & 0 \\
  1 & 1 & 0 & 1 \\
  1 & 0 & 1 & 1 \\
  0 & 0 & 1 & 1 \\
\end{pmatrix} \]

\[ V = \begin{pmatrix}
  10 & 1 & 0 \\
  8 & 0 & 8 \\
  1 & 6 & 1 \\
  -1 & 7 & 11 \\
\end{pmatrix} \]

\[ d = \begin{pmatrix}
  1 \\
  2 \\
  12 \\
\end{pmatrix} \]
SVD Revisited

Colab Notebook
Topics

- More Followup on Word Embeddings from SVD
- **Logistic Regression and Gradient Descent**
- Multilayer Perceptrons
- Word Embeddings from NNs
Logistic Regression Classifiers
Making Predictions
Logistic Regression Classifiers

Making Predictions

\[ y = \mathbf{w} \cdot \mathbf{x} \]
Logistic Regression Classifiers

Making Predictions

\[ y = \frac{1}{1 + e^{-(\overrightarrow{w} \cdot \overrightarrow{x})}} \]
Logistic Regression Classifiers
Training with Gradient Descent

minimize loss(data, w)
Logistic Regression Classifiers
Training with Gradient Descent

\[-Y \log \hat{Y} + (1 - Y) \log (1 - \hat{Y})\]
Goal (lowest achievable value for loss given data). You don't know what value of parameters will give you this.
Logistic Regression Classifiers
Training with Gradient Descent

\[-Y \log \hat{Y} + (1 - Y) \log (1 - \hat{Y})\]
Logistic Regression Classifiers
Training with Gradient Descent

\[-Y \log \hat{Y} + (1 - Y) \log (1 - \hat{Y})\]
Logistic Regression Classifiers
Training with Gradient Descent

\[-Y \log \hat{Y} + (1 - Y) \log (1 - \hat{Y})\]
Logistic Regression Classifiers
Training with Gradient Descent

\[-Y \log \hat{Y} + (1 - Y) \log (1 - \hat{Y})\]

Take small step to reduce loss, update parameters accordingly.
Logistic Regression Classifiers
Training with Gradient Descent

\[-Y \log \hat{Y} + (1 - Y) \log (1 - \hat{Y})\]

Repeat until you converge, i.e., loss can’t be decreased, or you time out (like in kmeans).
Logistic Regression Classifiers

Training with Gradient Descent

\[-Y \log \hat{Y} + (1 - Y) \log (1 - \hat{Y})\]

Take small step to reduce loss, update parameters accordingly.
Topics

• More Followup on Word Embeddings from SVD
• Logistic Regression and Gradient Descent
• **Multilayer Perceptrons**
• Word Embeddings from NNs
Language Modeling Task

Running Example

Task: Predict the next word in a sentence.

The cat sat on the ___
Basic Perceptron
Same as Logistic Regression

Task: Predict the next word
Input: the
Expected: cat

input features
weights
activations
activation function
output

activation function
sigmoid

$D \times O$

$N \times D$
$D \times O$
$N \times O$

$0.5 \quad 0.3 \quad 0.2 \quad 0.4 \quad 0.8 \quad 0.1 \quad 0.8$

*warning: numbers made up, matrix multiplications don't necessarily work out
Basic Perceptron

Same as Logistic Regression

$$y = \overrightarrow{w} \cdot \overrightarrow{x}$$

Task: Predict the next word
Input: the
Expected: cat

*warning: numbers made up, matrix multiplications don't necessarily work out*
**Basic Perceptron**

**Forward Pass**

**Task:** Predict the next word

**Input:** the

**Expected:** cat

---

**raw inputs**

**embedding lookup**

**input features**

**weights**

**activations**

**activation function**

output

---

N x V  V x D  N x D  D x O  N x O

---

*warning: numbers made up, matrix multiplications don't necessarily work out*
Basic Perceptron
Forward Pass

Task: Predict the next word
Input: the
Expected: cat

raw inputs -> embedding lookup -> input features -> weights -> activations -> activation function -> output

**Basic Perceptron**

Forward Pass

- **Input:** the
- **Expected Output:** cat

**Diagram:**
-raw inputs [the]
-embedding lookup [cat, mat, on, sat, the, is]
-input features [V x D]
-weights [N x D]
-activations [D x O]
-output [N x O]

**Numbers:**
- N x V
- V x D
- N x D
- D x O
- N x O

**Activation Function:**
-softmax

**Output:** 0.5

**Warning:** numbers made up, matrix multiplications don't necessarily work out
**Basic Perceptron**

**Forward Pass**

**Task:** Predict the next word  
**Input:** the  
**Expected:** cat

![Diagram of a basic perceptron forward pass](image)

- **Raw inputs:** the
- **Embedding lookup**
- **Input features**
- **Weights**
- **Activations**
- **Activation function**
- **Output**

<table>
<thead>
<tr>
<th>Input</th>
<th>cat</th>
<th>mat</th>
<th>on</th>
<th>sat</th>
<th>the</th>
<th>is</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Example Matrix Multiplications

- **$N \times V$**

- **$V \times D$**

- **$N \times D$**

- **$D \times O$**

- **$N \times O$**

*Warning: numbers made up, matrix multiplications don't necessarily work out.*
**Task:** Predict the next word

**Input:** the

**Expected:** cat

---

*Warning: numbers made up, matrix multiplications don't necessarily work out*
Basic Perceptron

Forward Pass

Task: Predict the next word
Input: the
Expected: cat

**Warning:** numbers made up, matrix multiplications don't necessarily work out
Basic Perceptron

Forward Pass

Task: Predict the next word
Input: the
Expected: cat

raw inputs
embedding lookup
input features
weights
activations
activation function
output

*warning: numbers made up, matrix multiplications don't necessarily work out
Basic Perceptron
Forward Pass

Task: Predict the next word
Input: the
Expected: cat

raw inputs | embedding lookup | input features | weights
--- | --- | --- | ---
the | | | |

N x V | V x D | N x D | D x O

activations | activation function
--- | ---
1 | 1
1 | 0
0 | 0
0 | 0
0 | 0

softmax

N x O | N x O

*warning: numbers made up, matrix multiplications don't necessarily work out
**Basic Perceptron**

**Forward Pass**

Task: Predict the next word
Input: the
Expected: cat

- **raw inputs**
- **embedding lookup**
- **input features**
- **weights**
- **activations**
- **activation function**
- **output**

**“feature”**
**“neuron”**
**“node”**

```
<table>
<thead>
<tr>
<th>N x V</th>
<th>V x D</th>
<th>N x D</th>
<th>D x O</th>
<th>N x O</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1 0 0 0 0</td>
<td>0 0 1 1 0 1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ma</td>
<td>0 1 0 0 0</td>
<td>0 0 1 0 0 1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>on</td>
<td>0 0 1 0 0 0</td>
<td>0 0 1 0 0 1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>sat</td>
<td>0 0 0 1 0 0</td>
<td>0 0 1 0 0 1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>0 0 0 0 1 0</td>
<td>0 0 1 1 0 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>0 0 0 0 0 1</td>
<td>0 0 1 1 0 0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

*warning: numbers made up, matrix multiplications don't necessarily work out*
Basic Perceptron
Forward Pass

Task: Predict the next word
Input: the
Expected: cat

raw inputs     embedding lookup

input features  weights  activations

output

“feature”
“neuron”
“node”

*warning: numbers made up, matrix multiplications don't necessarily work out
Basic Perceptron

Forward Pass

Task: Predict the next word
Input: the
Expected: cat

raw inputs
embedding lookup

“weight”
“parameter”
“connection”

N x V
V x D

input features
weights
activations

activation function
output

N x D
D x O
N x O

*warning: numbers made up, matrix multiplications don’t necessarily work out
Task: Predict the next word
Input: the
Expected: cat

*warning: numbers made up, matrix multiplications don't necessarily work out*
Multilayer Perceptron

Task: Predict the next word
Input: the
Expected: cat

Forward Pass

raw inputs
input features
weights
hidden state
weights
activations
activation function
output

embedding lookup

N x V
N x D
D x H
N x H
H x O
N x O

*warning: numbers made up, matrix multiplications don't necessarily work out*
Multilayer Perceptron
Forward Pass

Task: Predict the next word
Input: the
Expected: cat

Can capture abstractions over input features, e.g., “nouns” vs. “verbs”...

*warning: numbers made up, matrix multiplications don’t necessarily work out
Multilayer Perceptron
Forward Pass

Task: Predict the next word
Input: the
Expected: cat

Abstractions will capture groups that are “functionally equivalent” w.r.t. the training task.

Warning: numbers made up, matrix multiplications don’t necessarily work out
Multilayer Perceptron
Training with Backpropagation

Parameters are randomly initialized.

Task: Predict the next word
Input: the
Expected: cat

Training with Backpropagation

Parameters are randomly initialized.
*warning: numbers made up, matrix multiplications don't necessarily work out
## Multilayer Perceptron

### Training with Backpropagation

**Input:** the  
**Expected:** cat

**Task:** Predict the next word

---

### Example

<table>
<thead>
<tr>
<th>Raw Inputs</th>
<th>Embedding Lookup</th>
<th>Input Features</th>
<th>Weights</th>
<th>Hidden State</th>
<th>Weights</th>
<th>Activations</th>
<th>Activation Function</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>N x V</td>
<td>N x D</td>
<td>D x H</td>
<td>N x H</td>
<td>H x O</td>
<td>N x O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **cat:**  0  
- **mat:**  0  
- **on:**  0  
- **sat:**  0  
- **the:**  1  
- **is:**  0

- **Weights:**  
  - N x V  
  - N x D  
  - D x H  
  - N x H  
  - H x O  

- **Activations:**  
  - N x O  

---

### I.e., predictions are random

*Warning: numbers made up, matrix multiplications don't necessarily work out.*
Multilayer Perceptron
Training with Backpropagation

Compare predictions to ground truth output...

raw inputs
input features
weights
hidden state
weights
activations
activation function
output

embedding lookup
the

N x V
N x D
D x H
N x H
H x O
N x O

*warning: numbers made up, matrix multiplications don't necessarily work out
Multilayer Perceptron
Training with Backpropagation

Adjust each weight (using gradient descent and chain rule)

raw inputs

input features

weights

hidden state

weights

activations

activation function

output

embedding lookup

the

cat 0
mat 0
on 0
sat 0
the 1
is 0

N x V
N x D
D x H
N x H
H x O
N x O

N x V
N x D
D x H
N x H
H x O
N x O

*warning: numbers made up, matrix multiplications don't necessarily work out
Multilayer Perceptron
Training with Backpropagation

Adjust each weight
(using gradient descent and chain rule)

raw inputs
input features
weights
hidden state
weights
activations
activation function
output

embedding lookup

N x V
N x D
D x H
N x H
H x O
N x O

Adjust each weight
(using gradient descent and chain rule)

*warning: numbers made up, matrix multiplications don't necessarily work out
Multilayer Perceptron
Training with Backpropagation

Adjust each weight
(using gradient descent and chain rule)

raw inputs
input features
weights
hidden state
weights
activations
activation function
output

embedding lookup

the

N x V
N x D
D x H
N x H
H x O
N x O

Adjust each weight (using gradient descent and chain rule)

ε

*warning: numbers made up, matrix multiplications don’t necessarily work out
Topics

• More Followup on Word Embeddings from SVD
• Logistic Regression and Gradient Descent
• Multilayer Perceptrons
• Word Embeddings from NNs
Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat

Can capture abstractions over input features, e.g., "nouns" vs. "verbs"...

*warning: numbers made up, matrix multiplications don't necessarily work out
Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat

Embedding layer can also be low-dimensional and trained with backprop!

*warning: numbers made up, matrix multiplications don't necessarily work out
Word Embeddings from Neural Networks

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>mat</th>
<th>on</th>
<th>sat</th>
<th>the</th>
<th>is</th>
</tr>
</thead>
<tbody>
<tr>
<td>N x V</td>
<td>0.3</td>
<td>0.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>V x D</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>N x D</td>
<td>0.1</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>D x H</td>
<td>0.2</td>
<td>0.1</td>
<td>0.8</td>
<td>0.7</td>
<td>0.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Task: Predict the next word
Input: the
Expected: cat

Embedding layer can also be low-dimensional and trained with backprop!

*warning: numbers made up, matrix multiplications don't necessarily work out
**Word Embeddings from Neural Networks**

Task: Predict the next word
Input: the
Expected: cat

No different than other hidden states. But often gets special attention as the “word representation”

*warning: numbers made up, matrix multiplications don’t necessarily work out*
Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat

No different than other hidden states. But often gets special attention as the “word representation”

*warning: numbers made up, matrix multiplications don't necessarily work out
Word Embeddings from Neural Networks

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov  
Google Inc., Mountain View, CA  
tmikolov@google.com

Kai Chen  
Google Inc., Mountain View, CA  
kaichen@google.com

Greg Corrado  
Google Inc., Mountain View, CA  
gcorrado@google.com

Jeffrey Dean  
Google Inc., Mountain View, CA  
jeff@google.com

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov  
Google Inc.  
Mountain View  
mikolov@google.com

Ilya Sutskever  
Google Inc.  
Mountain View  
ilyasu@google.com

Kai Chen  
Google Inc.  
Mountain View  
kai@google.com

Greg Corrado  
Google Inc.  
Mountain View  
gcorrado@google.com

Jeffrey Dean  
Google Inc.  
Mountain View  
jeff@google.com
## Word Embeddings from Neural Networks

### Distribution of words in w’s context

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>kitten</th>
<th>cute</th>
<th>adorable</th>
<th>gradients</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>kitten</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cute</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>adorable</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gradients</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Word Embeddings from Neural Networks

A '1' in the position corresponding to the word "anis"

Word w being represented

10,000 positions

$\Sigma$

Input Vector

300 neurons

$\Sigma$

Hidden Layer
Linear Neurons

$\Sigma$

Output Layer
Softmax Classifier

Probability that the word at a randomly chosen, nearby position is "abandon"

... "ability"

... "able"

... "zone"
Word Embeddings from Neural Networks

A '1' in the position corresponding to the word "amis"

Output Layer Softmax Classifier

Probability that the word at a randomly chosen, nearby position is "abandon"

... "ability"

... "able"

... "zone"
Word Embeddings from Neural Networks
Continuous Bag of Words (CBOW)

Given context, predict the word
Given the word, predict the context
Pretrained Word Embeddings

Evaluations of word2vec embeddings
Pretrained Word Embeddings
Evaluations of word2vec embeddings
## Pretrained Word Embeddings

### Evaluations of word2vec embeddings

Table 8: Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).

<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>
Pretrained Word Embeddings

NNs vs. SVD
Pretrained Word Embeddings

NNs vs. SVD

- Same basic idea! Dimensionality reduction leads to good abstractions
Pretrained Word Embeddings

NNs vs. SVD

- Same basic idea! Dimensionality reduction leads to good abstractions
- In fact, the two methods are provably equivalent in the simplest case
Pretrained Word Embeddings
NNs vs. SVD

• Same basic idea! Dimensionality reduction leads to good abstractions
  • In fact, the two methods are provably equivalent in the simplest case
• But embeddings from NNs can become more powerful (and harder to interpret) as:

Neural Word Embedding as Implicit Matrix Factorization
Pretrained Word Embeddings

NNs vs. SVD

• Same basic idea! Dimensionality reduction leads to good abstractions
  • In fact, the two methods are provably equivalent in the simplest case
• But embeddings from NNs can become more powerful (and harder to interpret) as:
  • We add more layers
Pretrained Word Embeddings

NNs vs. SVD

• Same basic idea! Dimensionality reduction leads to good abstractions
  • In fact, the two methods are provably equivalent in the simplest case
• But embeddings from NNs can become more powerful (and harder to interpret) as:
  • We add more layers
  • We add more non-linearity
Pretrained Word Embeddings

NNs vs. SVD

• Same basic idea! Dimensionality reduction leads to good abstractions
  • In fact, the two methods are provably equivalent in the simplest case
• But embeddings from NNs can become more powerful (and harder to interpret) as:
  • We add more layers
  • We add more non-linearity
  • We invent more complex training objectives
Pretrained Word Embeddings

**NNs vs. SVD**

- Same basic idea! Dimensionality reduction leads to good abstractions
  - In fact, the two methods are provably equivalent in the simplest case
- But embeddings from NNs can become more powerful (and harder to interpret) as:
  - We add more layers
  - We add more non-linearity
  - We invent more complex training objectives
  - More next lecture(s)!