Advanced Machine Learning
Lecture 16

Word Embeddings

16.01.2017

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/
leibe@vision.rwth-aachen.de
Announcements

- Seminar registration period started
  - We will offer a lab course in the summer semester “Deep Robot Learning”
  - Topic: Deep reinforcement learning for robot control
    - Either UAV or grasping robot
  - If you’re interested, you can register at http://www.graphics.rwth-aachen.de/apse
  - Registration period: 13.01.2016 - 29.01.2016

  *Quick poll: Who would be interested in that?*
This Lecture: Advanced Machine Learning

- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes

- Approximate Inference
  - Sampling Approaches
  - MCMC

- Deep Learning
  - Linear Discriminants
  - Neural Networks
  - Backpropagation & Optimization
  - CNNs, RNNs, ResNets, etc.
Topics of This Lecture

• Recap
  ➢ ResNets
  ➢ Applications of CNNs

• Word Embeddings
  ➢ Neuroprobabilistic Language Models
  ➢ word2vec
  ➢ GloVe
  ➢ Hierarchical Softmax

• Outlook: Recurrent Neural Networks
Recap: Residual Networks

- Core component
  - Skip connections bypassing each layer
  - Better propagation of gradients to the deeper layers
  - This makes it possible to train (much) deeper networks.

\[ H(x) = F(x) + x \]
Recap: R-CNN for Object Detection

- Regions of Interest (RoI) from a proposal method (~2k)
- Forward each region through ConvNet
- Classify regions with SVMs
- Warped image regions
- Bbox reg
- SVMs
- ConvNet

Slide credit: Ross Girshick
Recap: Faster R-CNN

- One network, four losses
  - Remove dependence on external region proposal algorithm.
  - Instead, infer region proposals from same CNN.
  - Feature sharing
  - Joint training

⇒ Object detection in a single pass becomes possible.
Recap: Fully Convolutional Networks

- **CNN**
- **FCN**
- **Intuition**
  - Think of FCNs as performing a sliding-window classification, producing a heatmap of output scores for each class

Image source: Long, Shelhamer, Darrell
Recap: Semantic Image Segmentation

- Encoder-Decoder Architecture
  - Problem: FCN output has low resolution
  - Solution: perform upsampling to get back to desired resolution
  - Use skip connections to preserve higher-resolution information

Image source: Newell et al.
Topics of This Lecture

• Recap
  - ResNets
  - Applications of CNNs

• Word Embeddings
  - Neuroprobabilistic Language Models
  - word2vec
  - GloVe
  - Hierarchical Softmax

• Outlook: Recurrent Neural Networks
Neural Networks for Sequence Data

• Up to now
  - Simple structure: Input vector → Processing → Output

• In the following, we will look at sequence data
  - Interesting new challenges
  - Varying input/output length, need to memorize state, long-term dependencies, ...

• Currently a hot topic
  - Early successes of NNs for text / language processing.
  - Very good results for part-of-speech tagging, automatic translation, sentiment analysis, etc.
  - Recently very interesting developments for video understanding, image+text modeling (e.g., creating image descriptions), and even single-image understanding (attention processes).
Motivating Example

• Predicting the next word in a sequence
  • Important problem for speech recognition, text autocorrection, etc.

• Possible solution: The trigram (n-gram) method
  • Take huge amount of text and count the frequencies of all triplets (n-tuples) of words.
  • Use those frequencies to predict the relative probabilities of words given the two previous words

\[
\frac{p(w_3 = c | w_2 = b, w_1 = a)}{p(w_3 = d | w_2 = b, w_1 = a)} = \frac{\text{count}(abc)}{\text{count}(abd)}
\]

• State-of-the-art until not long ago…
Problems with N-grams

- **Problem: Scalability**
  - We cannot easily scale this to large $N$.
  - The number of possible combinations increases exponentially.
  - So does the required amount of data.

- **Problem: Partial Observability**
  - With larger $N$, many counts would be zero.
  - The probability is not zero, just because the count is zero!
  - Need to back off to (N-1)-grams when the count for N-grams is too small.
  - Necessary to use elaborate techniques, such as Kneser-Ney smoothing, to compensate for uneven sampling frequencies.
Let’s Try Neural Networks for this Task

- Important issues
  - How should we encode the words to use them as input?
  - What internal NN structure do we need?
  - How can we perform classification (softmax) with so many possible outputs?
Neural Probabilistic Language Model

**Core idea**
- Learn a shared distributed encoding (word embedding) for the words in the vocabulary.

Word Embedding

- **Idea**
  - Encode each word as a vector in a $d$-dimensional feature space.
  - Typically, $V \sim 1M$, $d \in (50, 300)$

- **Learning goal**
  - Determine weight matrix $W_{V \times d}$ that performs the embedding.
  - Shared between all input words

- **Input**
  - Vocabulary index $x$ in 1-of-K encoding.
  - For each input $x$, only one row of $W_{V \times d}$ is needed.
  - $W_{V \times d}$ is effectively a look-up table.

Image source: Xin Rong, 2015
Word Embedding: Full Network

- Train on large corpus of data, learn $W_{V \times d}$.
  $\Rightarrow$ Shown to outperform n-grams by [Bengio et al., 2003].

Many parameters: $W_{2d \times d}$ gets huge!
Visualization of the Resulting Embedding

(part of a 2.5D map of the most common 2500 words)

Image source: Geoff Hinton
Visualization of the Resulting Embedding
Visualization of the Resulting Embedding
Popular Word Embeddings

• Open issue
  - What is the best setup for learning such an embedding from large amounts of data (billions of words)?

• Several recent improvements
  - word2vec [Mikolov 2013]
  - GloVe [Pennington 2014]
  => Pretrained embeddings available for everyone to download.
word2vec

- **Goal**
  - Make it possible to learn high-quality word embeddings from huge data sets (billions of words in training set).

- **Approach**
  - Define two alternative learning tasks for learning the embedding:
    - “Continuous Bag of Words” (CBOW)
    - “Skip-gram”
  - Designed to require fewer parameters.

Image source: Mikolov et al., 2015
word2vec: CBOW Model

- Continuous BOW Model
  - Remove the non-linearity from the hidden layer
  - Share the projection layer for all words (their vectors are averaged)

⇒ Bag-of-Words model
(order of the words does not matter anymore)
word2vec: Skip-Gram Model

- Continuous Skip-Gram Model
  - Similar structure to CBOW
  - Instead of predicting the current word, predict words within a certain range of the current word.
  - Give less weight to the more distant words

- Implementation
  - Randomly choose a number $R \in [1, C]$.
  - Use $R$ words from history and $R$ words from the future of the current word as correct labels.
  - $R + R$ word classifications for each input.

Image source: Xin Rong, 2015
Interesting property

• Embedding often preserves linear regularities between words
  - Analogy questions can be answered through simple algebraic operations with the vector representation of words.

• Example
  - What is the word that is similar to small in the same sense as bigger is to big?
  - For this, we can simply compute
  \[ X = \text{vec}("bigger") - \text{vec}("big") + \text{vec}("small") \]
  - Then search the vector space for the word closes to \( X \) using the cosine distance.
  \[ \Rightarrow \] Result (when words are well trained): \( \text{vec}("smaller") \).

• Other example
  - E.g., \( \text{vec}("King") - \text{vec}("Man") + \text{vec}("Woman") \approx \text{vec}("Queen") \)
# Evaluation on Analogy Questions

<table>
<thead>
<tr>
<th>Type of relationship</th>
<th>Word Pair 1</th>
<th>Word Pair 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common capital city</td>
<td>Athens, Greece</td>
<td>Oslo, Norway</td>
</tr>
<tr>
<td>All capital cities</td>
<td>Astana, Kazakhstan</td>
<td>Harare, Zimbabwe</td>
</tr>
<tr>
<td>Currency</td>
<td>Angola, Illinois</td>
<td>Iran, Miguel</td>
</tr>
<tr>
<td>City-in-state</td>
<td>Chicago, sister</td>
<td>Stockton, Cali</td>
</tr>
<tr>
<td>Man-Woman</td>
<td>brother</td>
<td>grandson, California</td>
</tr>
<tr>
<td>Adjective to adverb</td>
<td>apparent, apparently</td>
<td>rapid, quickly</td>
</tr>
<tr>
<td>Opposite</td>
<td>possibly, impossibly</td>
<td>ethical, unethical</td>
</tr>
<tr>
<td>Comparative</td>
<td>great, greater</td>
<td>tough, tougher</td>
</tr>
<tr>
<td>Superlative</td>
<td>easy, easiest</td>
<td>lucky, luckiest</td>
</tr>
<tr>
<td>Present Participle</td>
<td>think, thinking</td>
<td>read, reading</td>
</tr>
<tr>
<td>Nationality adjective</td>
<td>Switzerland, Swiss</td>
<td>Cambodia, Cambodian</td>
</tr>
<tr>
<td>Past tense</td>
<td>walking, walked</td>
<td>swimming, swam</td>
</tr>
<tr>
<td>Plural nouns</td>
<td>mouse, mice</td>
<td>dollar, dollars</td>
</tr>
<tr>
<td>Plural verbs</td>
<td>work, works</td>
<td>speak, speaks</td>
</tr>
</tbody>
</table>

**semantic**

**syntactic**

---

Image source: Mikolov et al., 2015
Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Vector Dimensionality</th>
<th>Training words</th>
<th>Accuracy [%]</th>
<th>Training time [days x CPU cores]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Semantic</td>
<td>Syntactic</td>
</tr>
<tr>
<td>NNLM</td>
<td>100</td>
<td>6B</td>
<td>34.2</td>
<td>64.5</td>
</tr>
<tr>
<td>CBOW</td>
<td>1000</td>
<td>6B</td>
<td>57.3</td>
<td>68.9</td>
</tr>
<tr>
<td>Skip-gram</td>
<td>1000</td>
<td>6B</td>
<td>66.1</td>
<td>65.1</td>
</tr>
</tbody>
</table>

- word2vec embedding is able to correctly answer many of those analogy questions.
- CBOW structure better for syntactic tasks
- Skip-gram structure better for semantic tasks

Image source: Mikolov et al., 2015
Problems with 100k-1M outputs

- Weight matrix gets huge!

- Example: CBOW model
  - One-hot encoding for inputs
    ⇒ Input-hidden connections are just vector lookups.
  - This is not the case for the hidden-output connections!
  - State $h$ is not one-hot, and vocabulary size is 1M.
    ⇒ $W'_{N \times V}$ has $300 \times 1M$ entries
    ⇒ All of those need to be updated by backprop.
Problems with 100k-1M outputs

- Softmax gets expensive!
  - Need to compute normalization over 100k-1M outputs

Image source: Xin Rong, 2015
Solution: Hierarchical Softmax

- **Idea**
  - Organize words in binary search tree, words are at leaves.
  - Factorize probability of word $w_0$ as a product of node probabilities along the path.
  - Learn a linear decision function $y = v_{n(w,j)} \cdot h$ at each node to decide whether to proceed with left or right child node.
  - Decision based on output vector of hidden units directly.
Topics of This Lecture

• Recap: CNN Architectures

• Applications of CNNs

• Word Embeddings
  ➢ Neuroprobabilistic Language Models
  ➢ word2vec
  ➢ GloVe
  ➢ Hierarchical Softmax

• Embeddings in Vision
  ➢ Siamese networks
  ➢ Triplet loss networks

• Outlook: Recurrent Neural Networks
Siamese Networks

- Similar idea to word embeddings
  - Learn an embedding network that preserves (semantic) similarity between inputs
  - E.g., used for patch matching

---

B. Leibe
### Discriminative Face Embeddings

- **Learning an embedding using a Triplet Loss Network**
  - Present the network with triplets of examples
    - Negative
    - Anchor
    - Positive

- Apply triplet loss to learn an embedding $f(\cdot)$ that groups the positive example closer to the anchor than the negative one.

\[
\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2
\]

⇒ Used with great success in Google’s FaceNet face recognition
Topics of This Lecture

• Recap: CNN Architectures
• Applications of CNNs
• Word Embeddings
  ➢ Neuroprobabilistic Language Models
  ➢ word2vec
  ➢ GloVe
  ➢ Hierarchical Softmax
• Embeddings in Vision
  ➢ Siamese networks
  ➢ Triplet loss networks
• Outlook: Recurrent Neural Networks
Outlook: Recurrent Neural Networks

- **Up to now**
  - Simple neural network structure: 1-to-1 mapping of inputs to outputs

- **Next lecture: Recurrent Neural Networks**
  - Generalize this to arbitrary mappings

Image source: Andrej Karpathy
References and Further Reading

• **Neural Probabilistic Language Model**

• **word2vec**

• **GloVe**

• **Hierarchical Softmax**
References: Other Embeddings

• Face Embeddings