CENG 783

Deep Learning

Week – 10
Recurrent Neural Networks (contd.)

Sinan Kalkan
Today

- Finalize RNNs
  - Word-level text modelling
  - Image captioning
  - Neural machine translation
  - Echo state networks
  - Time-delay neural networks

- Neural Turing Machines

- NOTES:
  - Final Exam date: 31 May, 17:00.
  - HW3 to be announced next week.
  - Project demos and papers due: 6 June.
Word Embedding (word2vec)

Fig: http://www.languagejones.com/blog-1/2015/11/1/word-embedding
Why do we embed words?

• 1-of-n encoding is not suitable to learn from
  • It is sparse
  • Similar words have different representations
  • Compare the pixel-based representation of images: Similar images/objects have similar pixels

• Embedding words in a map allows
  • Encoding them with fixed-length vectors
  • “Similar” words having similar representations
  • Allows complex reasoning between words:
    • king - man + woman = queen

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>NEAREST TOKEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris - France + Italy</td>
<td>Rome</td>
</tr>
<tr>
<td>bigger - big + cold</td>
<td>colder</td>
</tr>
<tr>
<td>sushi - Japan + Germany</td>
<td>bratwurst</td>
</tr>
<tr>
<td>Cu - copper + gold</td>
<td>Au</td>
</tr>
<tr>
<td>Windows - Microsoft + Google</td>
<td>Android</td>
</tr>
<tr>
<td>Montréal Canadiens - Montreal + Toronto</td>
<td>Toronto Maple Leafs</td>
</tr>
</tbody>
</table>

Table 1: Mikolov et al. [3] showcase simple additive properties of their word embeddings.

More examples

http://deeplearning4j.org/word2vec
More examples

Male-Female

Verb tense

Country-Capital
More examples

• Geopolitics: *Iraq* - *Violence* = *Jordan*
• Distinction: *Human* - *Animal* = *Ethics*
• *President* - *Power* = *Prime Minister*
• *Library* - *Books* = *Hall*

http://deeplearning4j.org/word2vec
Using word embeddings

• E.g., for language modeling

• Given “I am eating …”, a language model can predict what can come next.
  • This both requires syntax and semantics (context)

• Before deep learning, n-gram (2-gram, 3-gram) models were state of the art.

Fig: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/
Using word embeddings

Fig: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/
Using word embeddings

Fig: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/
word2vec

- “Similarity” to Sweden (cosine distance between their vector representations)

<table>
<thead>
<tr>
<th>Word</th>
<th>Cosine distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>norway</td>
<td>0.760124</td>
</tr>
<tr>
<td>denmark</td>
<td>0.715460</td>
</tr>
<tr>
<td>finland</td>
<td>0.620022</td>
</tr>
<tr>
<td>switzerland</td>
<td>0.588132</td>
</tr>
<tr>
<td>belgium</td>
<td>0.585835</td>
</tr>
<tr>
<td>netherlands</td>
<td>0.574631</td>
</tr>
<tr>
<td>iceland</td>
<td>0.562368</td>
</tr>
<tr>
<td>estonia</td>
<td>0.547621</td>
</tr>
<tr>
<td>slovenia</td>
<td>0.531408</td>
</tr>
</tbody>
</table>

http://deeplearning4j.org/word2vec
Two different ways to train

1. Using context to predict a target word (~ continuous bag-of-words)

2. Using word to predict a target context (skip-gram)
   • Produces more accurate results on large datasets
   • If the vector for a word cannot predict the context, the mapping to the vector space is adjusted
   • Since similar words should predict the same or similar contexts, their vector representations should end up being similar

http://deeplearning4j.org/word2vec
Two different ways to train

1. Using context to predict a target word (~ continuous bag-of-words)

https://www.tensorflow.org/versions/r0.7/tutorials/word2vec/index.html
Two different ways to train

2. Using word to predict a target context (skip-gram)
   • Produces more accurate results on large datasets
   • Given a sentence:
     the quick brown fox jumped over the lazy dog
   • For each word, take context to be
     (N-words to the left, N-words to the right)
   • If N = 1 (context, word):
     ([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...
Some details

- CBOW is called continuous BOW since the context is regarded as a BOW and it is continuous.

- In both approaches, the networks are composed of linear units

- The output units are usually normalized with the softmax

- According to Mikolov:
  - “Skip-gram: works well with small amount of the training data, represents well even rare words or phrases.

  - CBOW: several times faster to train than the skip-gram, slightly better accuracy for the frequent words”
An extra issue: Sampling

- Greedy sampling: Take the most likely word at each step vs

- Beam search (or alternatives): Consider $k$ most likely words at each step, and expand search.

Figure: http://mttalks.ufal.ms.mff.cuni.cz/index.php

Figure: https://geekyisawesome.blogspot.com.tr/2016/10/using-beam-search-to-generate-most.html
Example: Image Captioning

Fig: https://github.com/karpathy/neuraltalk2
Demo video

https://vimeo.com/146492001
Overview

Pre-trained word embedding is also used.

Pre-trained CNN (e.g., on imagenet)

Image: Karpathy
Training

before:
\[ h_0 = \max(0, Wxh \times x_0) \]

now:
\[ h_0 = \max(0, Wxh \times x_0 + Wih \times v) \]
test image

sample!
test image

sample! <END> token => finish.
Example: Neural Machine Translation
Neural Machine Translation

- Model
Neural Machine Translation

• Model

Each box is an LSTM or GRU cell.

Sutskever et al. 2014

Haitham Elmarakeby
Neural Machine Translation

\[ f = (\text{La, croissance, économique, s'est, ralentie, ces, dernières, années, .}) \]

- Model: encoder

\[ e = (\text{Economic, growth, has, slowed, down, in, recent, years, .}) \]

Cho: From Sequence Modeling to Translation

Haitham Elmarakeby
Neural Machine Translation

\[ e = (\text{Economic, growth, has, slowed, down, in, recent, years, .}) \]
Neural Machine Translation

- Model-decoder

\[ f = (\text{La, croissance, éconmique, s'est, ralentie, ces, dernières, années, .}) \]

\[ e = (\text{Economic, growth, has, slowed, down, in, recent, years, .}) \]
Decoder in more detail

Given

(i) the “summary” \((c)\) of the input sequence,
(ii) the previous output / word \((y(t - 1))\)
(iii) the previous state \((h(t - 1))\)

the hidden state of the decoder is:
\[
h(t) = f(h(t - 1), y(t - 1), c)
\]

Then, we can find the most likely next word:
\[
P(y(t) | y(t - 1), y(t - 2), ..., c) = g(h(t), y(t - 1), c)
\]

\(f, g\): activation functions of our choice. For \(g\), we need a function that maps to probabilities, e.g., softmax.
Encoder-decoder

- Jointly trained to maximize

\[
\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_{\theta}(y_n | x_n),
\]
Check the following tutorial

- http://smerity.com/articles/2016/google_nmt_arch.html
More on attention

More on attention

Attention mechanism: A two-layer neural network.
Input: $z_i$ and $h_j$
Output: $e_j$, a scalar for the importance of word $j$.
The scores of words are normalized: $a_j = \text{softmax}(e_j)$
More on attention

What does Attention in Neural Machine Translation Pay Attention to?

Hamidreza Ghader and Christof Monz
Informatics Institute, University of Amsterdam, The Netherlands
h.ghader, c.monz@uva.nl

2017
NMT can be done at char-level too

This can be done with CNNs
Echo State Networks
Reservoir Computing
Motivation

• “Schiller and Steil (2005) also showed that in traditional training methods for RNNs, where all weights (not only the output weights) are adapted, the dominant changes are in the output weights. In cognitive neuroscience, a related mechanism has been investigated by Peter F. Dominey in the context of modelling sequence processing in mammalian brains, especially speech recognition in humans (e.g., Dominey 1995, Dominey, Hoen and Inui 2006). Dominey was the first to explicitly state the principle of reading out target information from a randomly connected RNN. The basic idea also informed a model of temporal input discrimination in biological neural networks (Buonomano and Merzenich 1995).”

http://www.scholarpedia.org/article/Echo_state_network
Echo State Networks (ESN)

- Reservoir of a set of neurons
  - Randomly initialized and fixed
  - Run input sequence through the network and keep the activations of the reservoir neurons
  - Calculate the “readout” weights using linear regression.

- Has the benefits of recurrent connections/networks

- No problem of vanishing gradient

Li et al., 2015.
The reservoir

- Provides non-linear expansion
  - This provides a “kernel” trick.
- Acts as a memory
- Parameters:
  - $W_{in}$, $W$ and $\alpha$ (leaking rate).
- Global parameters:
  - Number of neurons: The more the better.
  - Sparsity: Connect a neuron to a fixed but small number of neurons.
  - Distribution of the non-zero elements: Uniform or Gaussian distribution. $W_{in}$ is denser than $W$.
  - Spectral radius of $W$: Maximum absolute eigenvalue of $W$, or the width of the distribution of its non-zero elements.
    - Should be less than 1. Otherwise, chaotic, periodic or multiple fixed-point behavior may be observed.
    - For problems with large memory requirements, it should be bigger than 1.
  - Scale of the input weights.

Fig. 1: An echo state network.
A Practical Guide to Applying Echo State Networks

Mantas Lukoševičius

\[ \tilde{x}(n) = \tanh \left( W^\text{in}[1; u(n)] + Wx(n - 1) \right), \]  
(2)
\[ x(n) = (1 - \alpha)x(n - 1) + \alpha \tilde{x}(n), \]  
(3)

where \( x(n) \in \mathbb{R}^{N_x} \) is a vector of reservoir neuron activations and \( \tilde{x}(n) \in \mathbb{R}^{N_x} \) is its update, all at time step \( n \), \( \tanh(\cdot) \) is applied element-wise, \([; ; ]\) stands for a vertical vector (or matrix) concatenation, \( W^\text{in} \in \mathbb{R}^{N_x \times (1 + N_u)} \) and \( W \in \mathbb{R}^{N_x \times N_x} \) are the input and recurrent weight matrices respectively, and \( \alpha \in (0, 1] \) is the leaking rate. Other sigmoid wrappers can be used besides the \( \tanh \), which however is the most common choice. The model is also sometimes used without the leaky integration, which is a special case of \( \alpha = 1 \) and thus \( \tilde{x}(n) \equiv x(n) \).

\[ y(n) = W^\text{out}[1; u(n); x(n)], \]

again stands for a vertical vector (or matrix) concatenation. An additional nonlinearity can be applied to \( y(n) \) in (4), as well as feedback connections \( W^\text{fb} \) from \( y(n - 1) \) to \( \tilde{x}(n) \) in (2). A graphical representation of an echo state network is shown in Fig. 1.

Fig. 1: An echo state network.
Training ESN

\( Y_{\text{target}} = W_{\text{out}} X \)

Probably the most universal and stable solution to (8) in this context is ridge regression, also known as regression with Tikhonov regularization:

\[
W_{\text{out}} = Y_{\text{target}} X^T \left( X X^T + \beta I \right)^{-1},
\]

where \( \beta \) is a regularization coefficient explained in Section 4.2, and I is the identity matrix.

Overfitting (regularization):

\[
W_{\text{out}} = \arg \min_{W_{\text{out}}} \frac{1}{N_y} \sum_{i=1}^{N_y} \left( \sum_{n=1}^{T} \left( y_i(n) - y_{i \text{target}}(n) \right)^2 + \beta \| w_{i \text{out}} \|^2 \right),
\]
Beyond echo state networks

• **Good aspects of ESNs**
  Echo state networks can be trained very fast because they just fit a linear model.

• They demonstrate that it’s very important to initialize weights sensibly.

• They can do impressive modeling of one-dimensional time-series.
  – but they cannot compete seriously for high-dimensional data.

• **Bad aspects of ESNs**
  They need many more hidden units for a given task than an RNN that learns the hidden weights.
Similar models

• Liquid State Machines (Maas et al., 2002)
  • A spiking version of Echo-state networks

• Extreme Learning Machines
  • Feed-forward network with a hidden layer.
  • Input-to-hidden weights are randomly initialized and never updated
Time Delay Neural Networks
Skipped points
Skipping

- Stability
- Continuous-time recurrent networks
- Attractor networks

Stability of Discrete Time Recurrent Neural Networks and Nonlinear optimization problems

Dr. Nikita Barabanov, and Jayant Singh

Abstract We consider the method of Reduction of Dissipativity Domain to prove global Lyapunov stability of Discrete Time Recurrent Neural Networks. The standard and advanced criteria for Absolute Stability of these essentially nonlinear systems produce rather weak results. The method mentioned above is proved to be more powerful. It involves a multi-step procedure with maximization of special nonconvex functions over polytopes on every step. We derive conditions which guarantee an existence of at most one point of local maximum for such functions over every hyperplane. This nontrivial result is valid for wide range of neuron transfer functions.
An Empirical Exploration of Recurrent Network Architectures

Rafal Jozefowicz  
Google Inc.

Wojciech Zaremba  
New York University, Facebook

Ilya Sutskever  
Google Inc.

RAFA LJ@GOOGLE.COM

WOJ.ZAREMBA@GMAIL.COM

ILYASU@GOOGLE.COM
Under review as a conference paper at ICLR 2016

VISUALIZING AND UNDERSTANDING RECURRENT NETWORKS

Andrei Karpathy* Justin Johnson* LI Fei-Fei
Department of Computer Science, Stanford University
{karpathy, jcjohns, feifeili}@cs.stanford.edu