Neural Networks

Lecture 9: Applications
<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Probability and Information Theory</td>
</tr>
<tr>
<td>3</td>
<td>Basics of Machine Learning</td>
</tr>
<tr>
<td>4</td>
<td>Deep Feed-forward networks</td>
</tr>
<tr>
<td>5</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>6</td>
<td>Optimization and regularization</td>
</tr>
<tr>
<td>7</td>
<td>Convolutional Networks</td>
</tr>
<tr>
<td>8</td>
<td>Sequence Modeling: recurrent networks</td>
</tr>
<tr>
<td>9</td>
<td>Applications</td>
</tr>
<tr>
<td>10</td>
<td>Guest lecture: MindTitan</td>
</tr>
<tr>
<td>11</td>
<td>Software and Practical methods</td>
</tr>
<tr>
<td>12</td>
<td>Autoencoders &amp; GANs</td>
</tr>
<tr>
<td>13</td>
<td>Deep Reinforcement Learning</td>
</tr>
<tr>
<td>14</td>
<td>Deep Learning and the Brain</td>
</tr>
<tr>
<td>15</td>
<td>Conclusions and Future Perspective</td>
</tr>
<tr>
<td>16</td>
<td>Project presentations</td>
</tr>
</tbody>
</table>
Problems with MLPs for sequence tasks

**MLPs** only accept an input of fixed dimensionality and map it to an output of fixed dimensionality.

- Images

- Categories

- Text (Estonian)

- Text (English)
Convolutional Neural Networks

Parameter sharing

Convolution

Full connections
Recurrent Neural Networks (RNN)

Exploit and old idea in ML: share parameters across different parts of a model

- **Sharing** makes possible to extend to apply it to sequences of different lengths not seen during training

- **Without sharing** it would not be possible to share statistical strength and generalise to lengths of sequences not seen during training

RNNs share parameters: each output is a function of the previous output, with the same update rule applied
Recurrent Networks

When the task is to predict the future from the past, the network learns to use $h(t)$ as a summary of task relevant aspects of the sequence up to $t$.

The summary is lossy because it maps an arbitrary length sequence $(x(t), x(t-1), x(t-2), \ldots, x(2), x(1))$ to a fixed vector $h(t)$.

Most demanding situation for $h(t)$: approximately recover the input sequence.
Design patterns of RNNs

CHAPTER 10. SEQUENCE MODELING: RECURRENT AND RECURSIVE NETS

information flow forward in time (computing outputs and losses) and backward in time (computing gradients) by explicitly showing the path along which this information flows.

10.2 Recurrent Neural Networks

Armed with the graph unrolling and parameter sharing ideas of section 10.1, we can design a wide variety of recurrent neural networks.

Unfold

Figure 10.3: The computational graph to compute the training loss of a recurrent network that maps an input sequence of $x$ values to a corresponding sequence of output $o$ values.

Loss $L$ measures how far each $o$ is from the corresponding training target $y$. When using softmax outputs, we assume $o$ is the unnormalized log probabilities. The loss $L$ internally computes $\hat{y} = \text{softmax}(o)$ and compares this to the target $y$. The RNN has input to hidden connections parametrized by a weight matrix $U$, hidden-to-hidden recurrent connections parametrized by a weight matrix $W$, and hidden-to-output connections parametrized by a weight matrix $V$. Equation 10.8 defines forward propagation in this model.

Some examples of important design patterns for recurrent neural networks include the following:
Design patterns of RNNs

CHAPTER 10. SEQUENCE MODELING: RECURRENT AND RECURSIVE NETS

information flow forward in time (computing outputs and losses) and backward in time (computing gradients) by explicitly showing the path along which this information flows.

10.2 Recurrent Neural Networks

Armed with the graph unrolling and parameter sharing ideas of section 10.1, we can design a wide variety of recurrent neural networks.
Vanilla plain RNN: produce an output at each time stamp and have recurrent connections between hidden units (Turing complete)
Feedforward pass

RNN where input and output have the same length

Feed-forward propagation proceeds from left to right

Update equations:

\[
\begin{align*}
a^{(t)} & = b + Wh^{(t-1)} + Ux^{(t)}, \\
h^{(t)} & = \tanh(a^{(t)}), \\
o^{(t)} & = c + Vh^{(t)}, \\
\hat{y}^{(t)} & = \text{softmax}(o^{(t)}),
\end{align*}
\]
Design patterns of RNNs

Fixed vector as input: vector to sequence
Design patterns of RNNs

Caption generation

man in black shirt is playing guitar.
construction worker in orange safety vest is working on road.
two young girls are playing with lego toy.
boy is doing backflip on wakeboard.
Design patterns of RNNs

CHAPTER 10. SEQUENCE MODELING: RECURRENT AND RECURSIVE NETS

10.4 Encoder-Decoder Sequence-to-Sequence Architectures

We have seen in figure 10.5 how an RNN can map an input sequence to a fixed-size vector. We have seen in figure 10.9 how an RNN can map a fixed-size vector to a sequence. We have seen in figures 10.3, 10.4, 10.10, and 10.11 how an RNN can map an input sequence to an output sequence of the same length.

Figure 10.12: Example of an encoder-decoder or sequence-to-sequence RNN architecture, for learning to generate an output sequence \( (y^{(1)}, \ldots, y^{(n_y)}) \) given an input sequence \( (x^{(1)}, x^{(2)}, \ldots, x^{(n_x)}) \). It is composed of an encoder RNN that reads the input sequence and a decoder RNN that generates the output sequence (or computes the probability of a given output sequence). The final hidden state of the encoder RNN is used to compute a generally fixed-size context variable \( C \) which represents a semantic summary of the input sequence and is given as input to the decoder RNN.

**Encoder-Decoder:** sequence to sequence architecture
CHAPTER 10. SEQUENCE MODELING: RECURRENT AND RECURSIVE NETS

Figure 10.16: Block diagram of the LSTM recurrent network “cell.” Cells are connected recurrently to each other, replacing the usual hidden units of ordinary recurrent networks. An input feature is computed with a regular artificial neuron unit. Its value can be accumulated into the state if the sigmoidal input gate allows it. The state unit has a linear self-loop whose weight is controlled by the forget gate. The output of the cell can be shut off by the output gate. All the gating units have a sigmoid nonlinearity, while the input unit can have any squashing nonlinearity. The state unit can also be used as an extra input to the gating units. The black square indicates a delay of a single time step. Leaky units allow the network to accumulate information (such as evidence for a particular feature or category) over a long duration. However, once that information has been used, it might be useful for the neural network to forget the old state. For example, if a sequence is made of sub-sequences and we want a leaky unit to accumulate evidence inside each sub-subsequence, we need a mechanism to forget the old state by setting it to zero. Instead of manually deciding when to clear the state, we want the neural network to learn to decide when to do it.

3 blocks of parameters

1. Input to internal state
2. Previous internal state to the next
3. Internal state to output/hidden state
**LSTM cell**

![LSTM diagram]

\[
\begin{align*}
    f_i^{(t)} &= \sigma \left( b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)} \right) \\
    s_i^{(t)} &= f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma \left( b_i + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right) \\
    g_i^{(t)} &= \sigma \left( b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right) \\
    h_i^{(t)} &= \tanh \left( s_i^{(t)} \right) q_i^{(t)} \\
    q_i^{(t)} &= \sigma \left( b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)} \right)
\end{align*}
\]}
<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Probability and Information Theory</td>
</tr>
<tr>
<td>3</td>
<td>Basics of Machine Learning</td>
</tr>
<tr>
<td>4</td>
<td>Deep Feed-forward networks</td>
</tr>
<tr>
<td>5</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>6</td>
<td>Optimization and regularization</td>
</tr>
<tr>
<td>7</td>
<td>Convolutional Networks</td>
</tr>
<tr>
<td>8</td>
<td>Sequence Modeling: recurrent networks</td>
</tr>
<tr>
<td>9</td>
<td>Applications</td>
</tr>
<tr>
<td>10</td>
<td>Guest lecture: MindTitan</td>
</tr>
<tr>
<td>11</td>
<td>Software and Practical methods</td>
</tr>
<tr>
<td>12</td>
<td>Autoencoders &amp; GANs</td>
</tr>
<tr>
<td>13</td>
<td>Deep Reinforcement Learning</td>
</tr>
<tr>
<td>14</td>
<td>Deep Learning and the Brain</td>
</tr>
<tr>
<td>15</td>
<td>Conclusions and Future Perspective</td>
</tr>
<tr>
<td>16</td>
<td>Project presentations</td>
</tr>
</tbody>
</table>
Learning objectives

• Understand the needs of large scale deep learning

• Know most popular applications of modern deep networks in different fields
CHAPTER 1. INTRODUCTION

Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from Wikipedia (2015).

1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart et al., 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio et al., 1991)
7. Mean field sigmoid belief network (Saul et al., 1996)
8. LeNet-5 (LeCun et al., 1998b)
10. Deep belief network (Hinton et al., 2006)
11. GPU-accelerated convolutional network (Chellapilla et al., 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina et al., 2009)
14. Unsupervised convolutional network (Jarrett et al., 2009)
15. GPU-accelerated multilayer perceptron (Ciresan et al., 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le et al., 2012)
18. Multi-GPU convolutional network (Krizhevsky et al., 2012)
19. COTS HPC unsupervised convolutional network (Coates et al., 2013)
20. GoogLeNet (Szegedy et al., 2014a)

Improvement in accuracy
Increase in networks size
High performance hardware and software infrastructure
Fast implementations

CPU

• Exploit fixed point arithmetic in CPU families where this offers a speedup

• Insufficient for large scale applications

GPU

• NN training hardly requires branching, appropriate for GPU

• Parallelism & High memory bandwidth

• Low-level: Tensorflow, PyTorch, Theano (RIP)

• High-level: Keras, Caffe
Distributed implementations

- Multi-GPU
- Multi-machine
  - Model parallelism (multiple machines on single data point)
  - Data parallelism (each example run by separate machine)
    - Trivial at test time
    - Asynchronous SGD at train time
Model compression

- Commercial applications need low time and memory cost during testing
- If no personalization, then train once and deploy to many users
- End user is resource constrained (e.g. train speech recognition in computer cluster, then deploy it in mobile phone)
- Model compression aims to train a small model to mimic a large one (by sampling x and f(x) from the large model)
- Better than directly training a small model!
Dynamic structure

Mixture of experts: select which network will be used to compute the output given the current input
Specialized hardware

ASICS (application-specific integrated circuits)

FPGA (field programmable gated array)

GPU (with lower precision) / TPU

Optical systems (lasers systems)
Computer vision

Convolutional Neural Networks achieve many state-of-the-art results

Specialized pre-processing is no longer very important (but normalise the contrast of images... helps)
Computer vision

Classification

Retrieval
Computer vision

Detection (bounding box)

Segmentation
Computer vision

Segmentation (annotating each pixel as cell or not cell)
Can you find the cancer?
Computer vision

Face detection
Computer vision

Other body parts recognition…
Computer vision

Transcription

[Images of various traffic signs, including stop signs, no parking signs, and others related to traffic regulations.]
Visual tracking
Computer vision

Detection (intrusion)
Fill in your image!

edges2shoes

TOOL

line

eraser

INPUT

undo  clear  random

OUTPUT

pix2pix

process

save

https://affinelayer.com/pixsrv/
Source → Target domain transfer

- horse → zebra
- zebra → horse
- apple → orange
- summer Yosemite
- winter Yosemite

Text → Images synthesis

- this small bird has a pink breast and crown, and black primaries and secondaries.
- this magnificent fellow is almost all black with a red crest, and white cheek patch.
Computer vision

Caption generation

A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.
Speech recognition

Acoustic signal (natural language utterance) to sequence of words

Most industrial products now use NN (voice recognition in mobiles...)
Speech recognition

Spectrogram: spectrum of frequencies as they vary with time
Speech recognition

(Convnets, LSTMs)
Speech synthesis

From text (in human language) to acoustic signal

Input: Sound wave of me saying “Hello”
Neural Network
Output: “Hello”
Plain text

https://deepmind.com/blog/wavenet-launches-google-assistant/

Speech synthesis

Synthesis (Wavenets)

Simulates the sound of speech at low-level (building the waveform, @16000 samples per second)

"If you train it with an American’s speech, it produces American speech. If you train it with German, it produces German. And if you train it with Chopin, it produces… well, not quite Chopin, but piano in a logical, one might even be tempted to say creative vein."

https://deepmind.com/blog/wavenet-generative-model-raw-audio/
Natural Language Processing (NLP)

Natural occurring language are ambiguous and defy formal description.

Example: machine translation requires to read a sentence in one human language and emit an equivalent sentence in another human language.

NLP applications are based on language models (probability distribution over sequences of words).
Natural Language Processing (NLP)

Natural occurring language are ambiguous and defy formal description

Example: machine translation requires to read a sentence in one human language and emit an equivalent sentence in another human language

NLP applications are based on language models (probability distribution over sequences of words)

**N-grams models**: sequence of n tokens (words)
Estimate probability of n-gram by how many times each possible n-gram occurs in the training set.

\[
P(\text{THE DOG RAN AWAY}) = P_3(\text{THE DOG RAN})P_3(\text{DOG RAN AWAY})/P_2(\text{DOG RAN}).
\]
Natural Language Processing (NLP)

N-grams models suffer from

curse of dimensionality: $|\mathbb{V}|^n$ n-grams. Even in massive training dataset, most n-grams do not occur.

each word being a one-hot vector: any two different words have the same distance from each other. Neighbours to leverage information are only training examples that repeat literally the same context.

"Cat" = (0,0,0,0,0,0,1,0,0,0,0,0)
"Dog" = (0,0,1,0,0,0,0,0,0,0,0,0)
Natural Language Processing (NLP)

**Neural language model:** use a distributed representation of words (also called word embeddings)

![Diagram of Skip-gram model](Image)
Natural Language Processing (NLP)

**Neural language model:** use a distributed representation of words (also called word embeddings)
Natural Language Processing (NLP)

**Neural language model:** use a distributed representation of words (also called word embeddings)

Vectors are positioned such that words that share common context in the corpus are located in close proximity in the new space

"Cat" = (0.12, 0.31, 1.23, 0.05, 0.91)

"Dog" = (0.17, 0.23, 1.05, 0.04, 0.6)
Natural Language Processing (NLP)

**Neural language model:** use a distributed representation of words (also called word embeddings)
Natural Language Processing (NLP)

**Neural language model:** use a distributed representation of words (also called word embeddings)

**word2vec**

\[ \text{(WATER} - \text{WET}) + \text{FIRE} = \text{FLAMES} \]

\[ \text{(PARIS} - \text{FRANCE}) + \text{ITALY} = \text{ROME} \]

\[ \text{(WINTER} - \text{COLD}) + \text{SUMMER} = \text{WARM} \]

\[ \text{(MINOTAUR} - \text{MAZE}) + \text{DRAGON} = \text{SIMCITY} \]
Economic growth has slowed down in recent years.

Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt.
Neural machine translation: models a conditional probability of a natural language sentence
Reinforcement Learning
Playing video games
Reinforcement Learning
AlphaGo Zero

Starting from scratch
Knowledge and relations

\[(\text{entity}_i, \text{relation}_j, \text{entity}_k)\] \quad (\text{subject, verb, object}) \quad \text{Relation}

\[(\text{entity}_i, \text{attribute}_j)\] \quad \text{Attribute}

Many applications require representing relations and reasoning about them
Knowledge and relations

\[(\text{entity}_i, \text{relation}_j, \text{entity}_k)\] \quad (\text{subject, verb, object}) \quad \text{Relation}

\[(\text{entity}_i, \text{attribute}_j)\] \quad \text{Attribute}

Many applications require representing relations and reasoning about them.

**Data**: Knowledge/relational databases (Freebase, Opencyc, WordNet, Wikibase, ..., GeneOntology).

**Knowledge graphs**: store facts about the world as relation between entities.

**Entities** are no longer just strings but real world objects with attributes, taxonomic information and relations to other objects.
Knowledge and relations

Neural language model: embedding vector for each entity and relations

Knowledge Graph Triples

Latent Variable Model

ULM

Albert Einstein

bornIn

Latent representations (or embeddings) for Entities and Relation-Types that disentangle complex relationships observed in the data (semantics).

Similarities

1. Link-Prediction
2. Link-based Clustering
3. Disambiguation
Visual Q&A

What vegetable is on the plate?
Neural Net: **broccoli**
Ground Truth: **broccoli**

What color are the shoes on the person's feet?
Neural Net: **brown**
Ground Truth: **brown**

How many school busses are there?
Neural Net: **2**
Ground Truth: **2**

What sport is this?
Neural Net: **baseball**
Ground Truth: **baseball**

What is on top of the refrigerator?
Neural Net: **magnets**
Ground Truth: **cereal**

What uniform is she wearing?
Neural Net: **shorts**
Ground Truth: **girl scout**

What is the table number?
Neural Net: **4**
Ground Truth: **40**

What are people sitting under in the back?
Neural Net: **bench**
Ground Truth: **tent**
Visual Q&A

Question
What object is flying?

Extract visual features

Embedding

Merge

Predict answer

Answer
Kite
Art

Neural style transfer: decompose style + content
<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Probability and Information Theory</td>
</tr>
<tr>
<td>3</td>
<td>Basics of Machine Learning</td>
</tr>
<tr>
<td>4</td>
<td>Deep Feed-forward networks</td>
</tr>
<tr>
<td>5</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>6</td>
<td>Optimization and regularization</td>
</tr>
<tr>
<td>7</td>
<td>Convolutional Networks</td>
</tr>
<tr>
<td>8</td>
<td>Sequence Modeling: recurrent networks</td>
</tr>
<tr>
<td>9</td>
<td>Applications</td>
</tr>
<tr>
<td>10</td>
<td>Guest lecture: MindTitan</td>
</tr>
<tr>
<td>11</td>
<td>Software and Practical methods</td>
</tr>
<tr>
<td>12</td>
<td>Autoencoders and GANs</td>
</tr>
<tr>
<td>13</td>
<td>Deep Reinforcement Learning</td>
</tr>
<tr>
<td>14</td>
<td>Deep Learning and the Brain</td>
</tr>
<tr>
<td>15</td>
<td>Conclusions and Future Perspective</td>
</tr>
<tr>
<td>16</td>
<td>Project presentations</td>
</tr>
</tbody>
</table>