Neural Networks

Lecture 1: Introduction
Objectives

• to acquire main concepts of the theory and practice of modern neural networks
Objectives

• to acquire main concepts of the theory and practice of modern neural networks

• to apply deep learning approaches to different research areas
Objectives

• to acquire main concepts of the theory and practice of modern neural networks

• to apply deep learning approaches to different research areas

• to develop a critical view of recent literature and news about deep learning technology
Organization

https://courses.cs.ut.ee/2021/nn/spring
<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raul Vicente</td>
<td><a href="mailto:raulvicente@gmail.com">raulvicente@gmail.com</a></td>
<td>L</td>
</tr>
<tr>
<td>Kallol Roy</td>
<td><a href="mailto:kallol.roy@ut.ee">kallol.roy@ut.ee</a></td>
<td>L</td>
</tr>
<tr>
<td>Tarun Khajuria</td>
<td><a href="mailto:tarunkhajuria42@gmail.com">tarunkhajuria42@gmail.com</a></td>
<td>P1</td>
</tr>
<tr>
<td>Youssef Mohamed</td>
<td><a href="mailto:youssef.mohamed@ut.ee">youssef.mohamed@ut.ee</a></td>
<td>P1/2</td>
</tr>
<tr>
<td>Victor Pinheiro</td>
<td><a href="mailto:victor.pinheiro@ut.ee">victor.pinheiro@ut.ee</a></td>
<td>P2</td>
</tr>
</tbody>
</table>
Organization

https://courses.cs.ut.ee/2021/nn/spring

Raul Vicente        raulvicente@gmail.com
Kallol Roy           kallol.roy@ut.ee
Tarun Khajuria         tarunkhajuria42@gmail.com
Youssef Mohamed    youssef.mohamed@ut.ee
Victor Pinheiro        victor.pinheiro@ut.ee

Group of Computational Neuroscience
Delta: room 3072
Schedule of the course

• Lectures (Tue 14:15 @ Zoom)

• Practical (Thu 14:15 @ Zoom)

• Practical (Thu 16:15 @ Zoom)
<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>Software and Practical methods</td>
</tr>
<tr>
<td>11</td>
<td>Autoencoders &amp; GANs</td>
</tr>
<tr>
<td>12</td>
<td>Attention &amp; Transformers</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>
<tr>
<td>Lesson</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------</td>
</tr>
<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>Software and Practical methods</td>
</tr>
<tr>
<td>11</td>
<td>Autoencoders &amp; GANs</td>
</tr>
<tr>
<td>12</td>
<td>Attention &amp; Transformers</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>
Homework

- 7
- Small exercises/questionnaire (about concepts in lecture)
- **Practical session (Python notebook)**

http://cs231n.github.io/python-numpy-tutorial/
Project

- 1-4 people
- Small research (data analysis, competition, analytical calculation, replication of a model, …)
- Blog post
- Oral/poster presentation (10 min)
Project (ANN playing Atari)
Project (self-driving minicar)
Test

- Test practice
- Similar to a small homework
- Alone
- End of the course
Grading system

Differentiated: A-F

- Homework (30)
- Project (40)
- Test practice (30)

You can collect up to 110 points, 90+ will give you A, 80+ B, 70+ C and so on.

50% for each component to get a final assessment

Bonus exercises (up to 10%)
Pre-requisites

- Programming (Python)
- Calculus, basic probability & statistics, linear algebra, machine learning...
ALEXANDRE ROBICQUET
CHERCHEUR EN INTELLIGENCE ARTIFICIELLE
LEARNING CURVE
Self-taught AI software attains human-level performance in video games

ALL SYSTEMS GO
At last—a computer program that can beat a champion Go player

SHARE DATA IN OUTBREAKS
TLA954

A GIANT IN THE EARLY UNIVERSE
A supermassive black hole at redshift of 6.5

TELEPORTATION FOR TWO
Transforming properties of entangled photons

SONGBIRDS À LA CARTE
Illegal barrows of millions of Mediterranean birds

SAFEGUARD TRANSPARENCY
Don’t let open-source backfire on Nashville

WHEN GENES GOT ‘SELFISH’
Dominance’s undoing came 400 years on
Why Artificial Intelligence?

Input:
- Sensors
- Data

Artificial Intelligence

Output:
- Movement
- Language
What is Deep Learning?

Methods that learn *multiple* levels of representation to *model* the *relation* between *input data and output*.
Deep Learning = Artificial Neural Networks
Artificial neural network

- A collection of simple trainable mathematical units, which collaborate to compute a complicated function
- Compatible with supervised, unsupervised, and reinforcement
- Brain inspired (loosely)
The neuron

- Different weights compute different functions

\[ y_i = F \left( \sum_i w_i x_i \right) \]
The neuron

- Different weights compute different functions

\[ y_i = F \left( \sum_i w_i x_i \right) \]

\[ F(x) = \frac{1}{1 + \exp(-x)} \]
The neuron

- Different weights compute different functions

\[ y_i = F \left( \sum_i w_i x_i \right) \]

\[ F(x) = \frac{1}{1 + \exp(-x)} \]
CHAPTER 6. DEEP FEEDFORWARD NETWORKS

Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units.

In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space.

In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact. Sometimes we annotate the edges in this graph with the name of the parameters that describe the relationship between two layers. Here, we indicate that a matrix \( W \) describes the mapping from \( x \) to \( h \), and a vector \( w \) describes the mapping from \( h \) to \( y \). We typically omit the intercept parameters associated with each layer when labeling this kind of drawing.

In this network, we used a vector of weights and a scalar bias parameter to describe an affine transformation from an input vector to an output scalar. Now, we describe an affine transformation from a vector \( x \) to a vector \( h \), so an entire vector of bias parameters is needed. The activation function \( g \) is typically chosen to be a function that is applied element-wise, with

\[
  h_i = g(x_i > W_i + c) + b_i
\]

In modern neural networks, the default recommendation is to use the rectified linear unit or ReLU (Jarrett et al., 2009; Nair and Hinton, 2010; Glorot et al., 2011a) defined by the activation function

\[
  g(z) = \max\{0, z\}
\]
depicted in figure 6.3.

We can now specify our complete network as

\[
  f(x; W, c, w, b) = w > \max\{0, W > x + c\} + b.
\]

We can now specify a solution to the XOR problem. Let

\[
  W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},
\]

\[
  c = \begin{bmatrix} 0 \\ 1 \end{bmatrix},
\]

\[
  w = \begin{bmatrix} 1 \\ 1 \end{bmatrix},
\]

\[
  b = 0.
\]
CHAPTER 6. DEEP FEEDFORWARD NETWORKS

Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units.

(Left) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space.

(Right) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact. Sometimes we annotate the edges in this graph with the name of the parameters that describe the relationship between two layers. Here, we indicate that a matrix $W$ describes the mapping from $x$ to $h$, and a vector $w$ describes the mapping from $h$ to $y$. We typically omit the intercept parameters associated with each layer when labeling this kind of drawing.

Previously, we used a vector of weights and a scalar bias parameter to describe an affine transformation from an input vector to an output scalar. Now, we describe an affine transformation from a vector $x$ to a vector $h$, so an entire vector of bias parameters is needed. The activation function $g$ is typically chosen to be a function that is applied element-wise, with $h_i = g(x > W_i + c_i)$. In modern neural networks, the default recommendation is to use the rectified linear unit or ReLU (Jarrett et al., 2009; Nair and Hinton, 2010; Glorot et al., 2011a) defined by the activation function $g(z) = \max\{0, z\}$ depicted in figure 6.3.

We can now specify our complete network as

$$f(x; W, c, w, b) = w > \max\{0, W > x + c\} + b. \tag{6.3}$$

We can now specify a solution to the XOR problem. Let $W = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, $c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, and $w = \begin{pmatrix} 1 \end{pmatrix}$.
The network

\[ y = f^{(2)}(h; w, b) \]

\[ h = f^{(1)}(x; W, c) \]
The network

\[ f(x; W, c, w, b) = f^{(2)}(f^{(1)}(x)) \]
Neural network

Input:

Output:
Neural network

Output: “Dog”  “Cat”

Input:
Neural network

Input:

Output:
Neural network
Neural network

Output:

“Dog”  “Cat”

Input:
Learning algorithm

- **while** not done
  - pick a random training case \((x, y)\)
  - run neuronal network on input \(x\)
  - modify connection weights to make prediction closer to \(y\)
McCulloch & Pitts (1943)
A Logical Calculus of the Ideas
Immanent in Nervous Activity

Rosenblatt (1957)
Perceptron

Minsky & Papert (1969)
Perceptrons: an introduction to computational geometry

Rumelhart, Hinton & Williams (1986)
Learning representations by back-propagating errors
Why now?
Why now?
Volume of data

90% data created globally between 2013-2015

10% datos creados Before 2013

Source: IBM
5000 samples
to work reasonably

10000000 samples
to rival human level
Why now?

× 1000000
Why now?
• **Pre-training (weights initialization)**
  (complex landscape)

• **Efficient descent algorithms**
  (complex landscape)

• **Activation**
  (vanishing gradient)

• **Dropout**
  (overfitting)

• **Domain Prior Knowledge**
Now that we are deep...

- Instead of hand-crafted features, let the algorithm build the relevant features for your problem
- More representational power for learning
- Powerful function approximator
With deep networks...

- the network can build blocks made of blocks...

- the key of deep learning is that allows a better representation of the data for each task
Object recognition

ImageNet competition

1000 categories

http://www.clarifai.com/
2012

25% → 15%
Automatic description of images

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."
The activity is

**A: Mountain Biking**
... because he is riding a bicycle down a mountain path in a mountainous area.

**A: Road Biking**
... because he is wearing a cycling uniform and riding a bicycle down the road.
https://youtu.be/kSLJriaOumA
<table>
<thead>
<tr>
<th>Images and video</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical imaging</td>
<td>Speech</td>
</tr>
<tr>
<td></td>
<td>Machine translation</td>
</tr>
<tr>
<td></td>
<td>Sentiment analysis</td>
</tr>
<tr>
<td><img src="image.png" alt="Image" /></td>
<td>Bioinformatics</td>
</tr>
<tr>
<td><img src="image.png" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td><img src="image.png" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td><img src="image.png" alt="Image" /></td>
<td></td>
</tr>
</tbody>
</table>
Playing Atari with Deep Reinforcement Learning
IA that in few hours learns to play better than humans!
State

Agent

Reward

Action $a_t$

Reward $r_t$

State $s_{t+1}$

Environment
It does not know the game rules, nor what is a ball, nor that blocks can be broken,...

Its only obsession in life is to increase the reward!
AlphaGo Zero
Starting from scratch

Alpha Zero

Alpha Zero vs Stockfish

Hanabi
If all you have is a hammer in the toolbox, everything looks like a nail.”

- Bernard Baruch
Challenges

• Why NN are so efficient? / Mathematical theory
• Neuroscience inspirations and back
• Explainability
• Transfer learning/causality
• Reinforcement learning
• Neuromorphic hardware
• ...

Take home message

- Deep Learning = Neural Networks 3.0
- Feature learning & approximate wild functions
- Pushing ML and AI to unthinkable applications a few years ago
<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>Software and Practical methods</td>
</tr>
<tr>
<td>11</td>
<td>Autoencoders &amp; GANs</td>
</tr>
<tr>
<td>12</td>
<td>Attention &amp; Transformers</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>
Reminder

Thursday at 14:15: LECTURE
To know more

*Deep Learning*, Goodfellow, Bengio, Courville, MIT press 2017