CSI5180. Machine Learning for Bioinformatics Applications

Deep learning: fundamentals

by

Marcel Turcotte

Version **November 12, 2019**

Preamble

Preamble

Deep learning: fundamentals

In this lecture, we study artificial neural networks (ANN) and specifically the multilayer architectures known as deep learning. This is the first of three lectures on this topic. Herein, we focus on the building blocks, namely the units, their connectivity, and the training algorithms.

General objective :

- **Discuss** the similarities and differences between other machine learning algorithms and deep learning.

Learning objectives

- **Explain** in your own words the threshold logic unit
- **Discuss** the role of the activation function
- **Describe** the multilayer perceptron

Reading:

- James Zou, Mikael Huss, Abubakar Abid, Pejman Mohammadi, Ali Torkamani, and Amalio Telenti, A primer on deep learning in genomics, *Nat Genet* **51**:1, 1218, 2019.
- Webb, S. Deep Learning for Biology. *Nature* **554**, 555557 2018.

Reading

- Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby, An introduction to deep learning on biological sequence data: examples and solutions, *Bioinformatics* **33**:22, 36853690, 2017.
- Mufti Mahmud, Mohammed Shamim Kaiser, Amir Hussain, and Stefano Vassanelli, Applications of deep learning and reinforcement learning to biological data, *IEEE Transactions on Neural Networks and Learning Systems* **29**, 20632079, 2018.

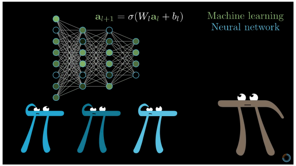
Reading

- Seonwoo Min, Byunghan Lee, and Sungroh Yoon, Deep learning in bioinformatics, *Brief Bioinform* **18**:5, 851869, 2017.
- Gökcen Eraslan, Ziga Avsec, Julien Gagneur, and Fabian J Theis, Deep learning: new computational modelling techniques for genomics, *Nat Rev Genet* **20**:7, 389403, 2019.
- Binhua Tang, Zixiang Pan, Kang Yin, and Asif Khateeb, Recent advances of deep learning in bioinformatics and computational biology, *Frontiers in Genetics* **10**, 214, 2019.

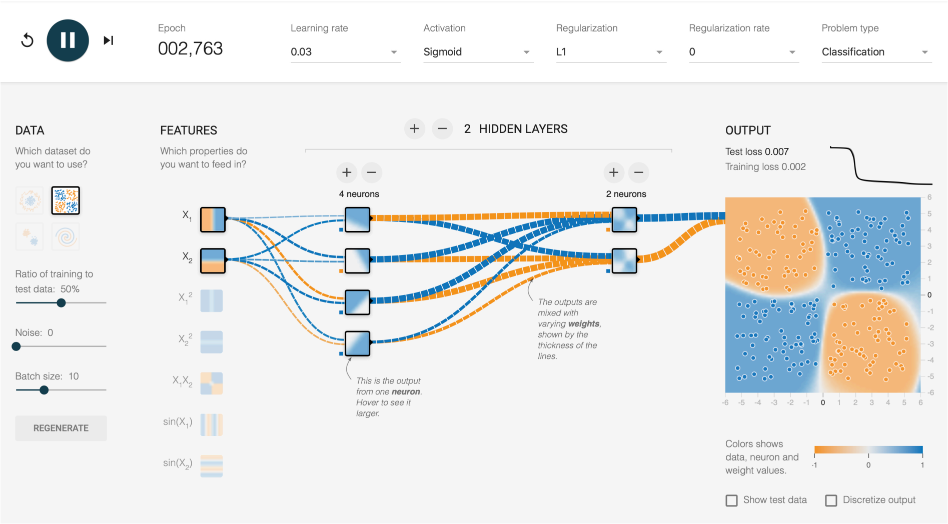
Plan

3Blue1Brown on deep learning (videos)

- **But what is a Neural Network?**
  https://youtu.be/aircAruvnKk
  19 minutes
- **Gradient descent, how neural networks learn**
  https://youtu.be/IHZwWFHWa-w
  21 minutes
- **What is backpropagation really doing?**
  https://youtu.be/Ilg3gGewQ5U
  14 minutes
- **Backpropagation calculus**
  https://youtu.be/tIeHLnjs5U8
  10 minutes

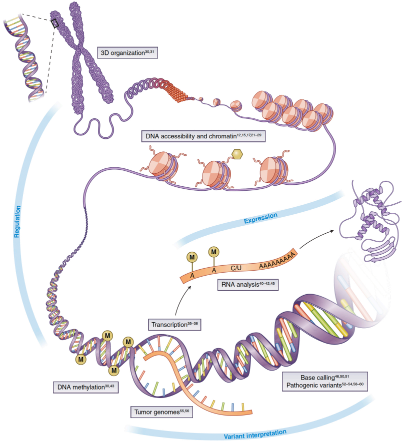


playground.tensorflow.org



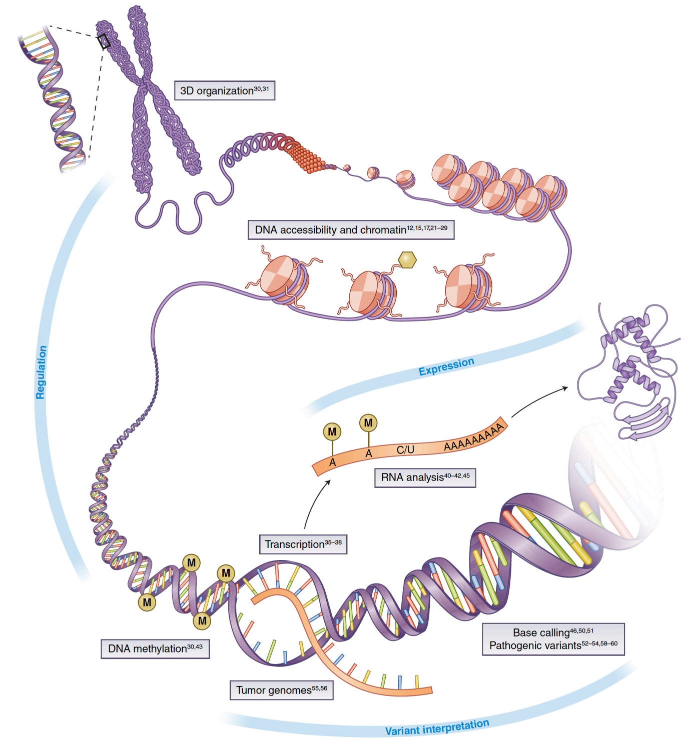
Application

What are the applications?



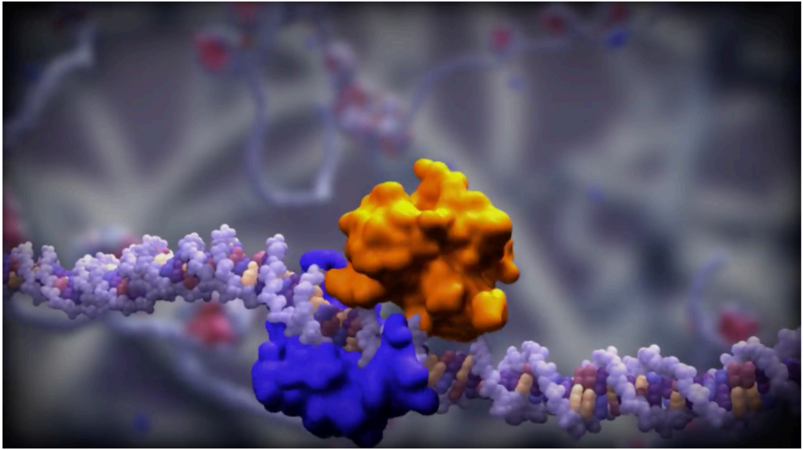
Source [Zou et al., 2019] Figure 2

What are



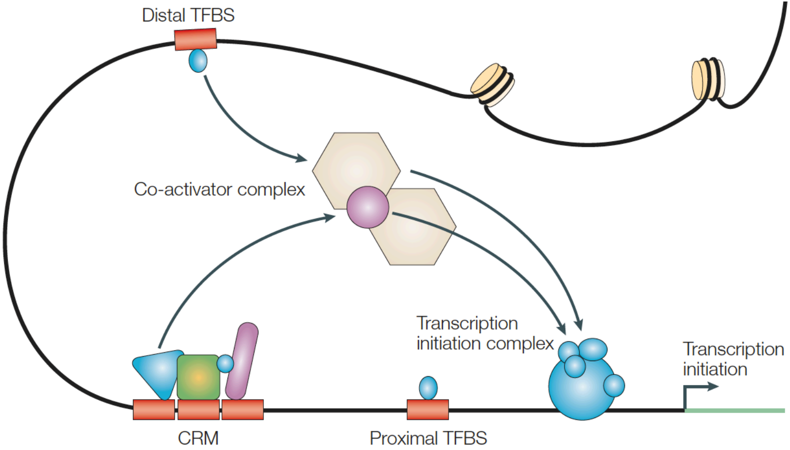
3D organization[30,31]
DNA accessibility and chromatin[12,15,17,21–29]
Regulation
Expression
M
M
A
A
C/U
AAAAAAAAA
RNA analysis[40–42,45]
Transcription[35–38]
M
M
M
M
DNA methylation[30,43]
Tumor genomes[55,56]
Base calling[46,50,51]
Pathogenic variants[52–54,58–60]
Variant interpretation

Transcription factors



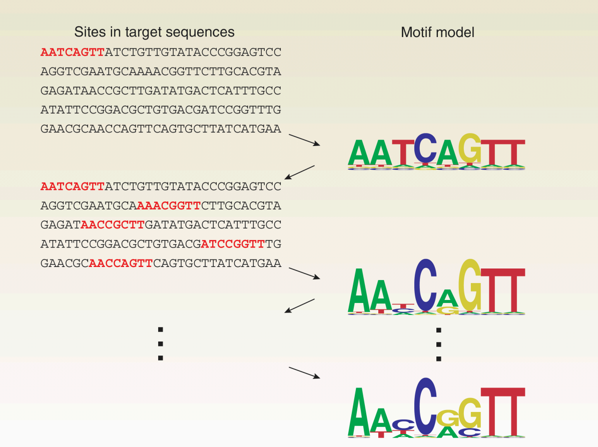
Source: `https://youtu.be/MkUgkDLp2iE`

Essential Cellular Biology: Gene Regulation



Wasserman, W. & Sandelin, A. Applied bioinformatics for the identification of regulatory elements. *Nat Rev Genet* **5**, 276287 (2004).

DNA sequence motif discovery



- Patrik Dhaeseleer, How does DNA sequence motif discovery work?, *Nat Biotechnol* **24**:8, 95961, 2006.

Deep Learning in Genomics Primer

- James Zou, Mikael Huss, Abubakar Abid, Pejman Mohammadi, Ali Torkamani, and Amalio Telenti, A primer on deep learning in genomics, *Nat Genet* **51**:1, 1218, 2019.
  - Google Colab Notebook

Introduction

Artificial neural networks

- "Although **planes** were inspired by **birds**, **they don't have to flap their wings**." [Géron, 2019]

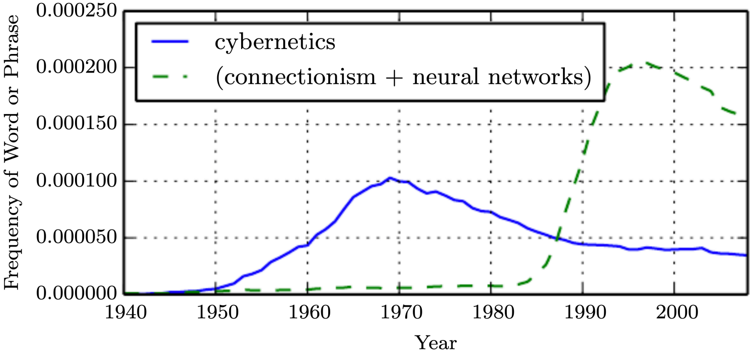
Artificial neural networks

- "Although **planes** were inspired by **birds**, **they don't have to flap their wings**." [Géron, 2019]
- Likewise, **artificial neural networks** (**ANN**) were inspired by the **human brain**.

Artificial neural networks

- "Although **planes** were inspired by **birds**, **they don't have to flap their wings**." [Géron, 2019]
- Likewise, **artificial neural networks** (**ANN**) were inspired by the **human brain**.
  - However, ANN do **not** (necessarily) **mimic** the brain or **explain** its functioning.

1940-1960, 1980-mid 1990, 2010-



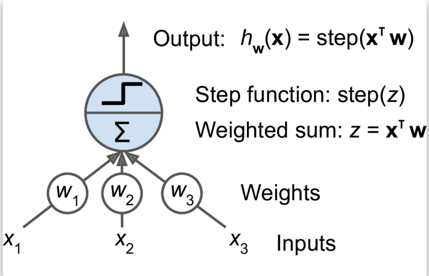
Source: [Goodfellow et al., 2016] Figure 1.7

- McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* **5**, 115133 (1943).

What has changed?

- **Vast amounts of data** are now available
- **Powerful computers** (including fast GPU)
- Improved **training algorithms**

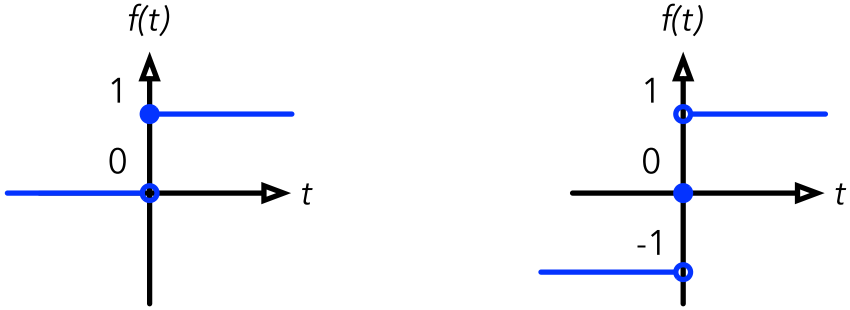
Neuron - threshold logic unit



Source: [Géron, 2019] Figure 10.4

- Common **step functions** include the **heavyside** function (0 if the input is negative and 1 otherwise) or the **sign** function (-1 if the input is negative, 0 if the input is zero, 1 otherwise).

Simple step functions - heavyside and sign



heavyside(t) =

- 1, if $t \geq 0$
- 0, if $t < 0$

sign(t) =

- 1, if $t > 0$
- 0, if $t = 0$
- -1, if $t = 0$

Does this sound familiar?

- One **threshold logic unit** (TLU) is similar to a **logistic regression** and both are solving the same kinds of problems.

Logistic (Logit) Regression

- Despite its name, **Logistic Regression** is a **classification** algorithm.

Logistic (Logit) Regression

- Despite its name, **Logistic Regression** is a **classification** algorithm.
- The **labels** are binary values, $y_i \in \{0, 1\}$.

Logistic (Logit) Regression

- Despite its name, **Logistic Regression** is a **classification** algorithm.
- The **labels** are binary values, $y_i \in \{0, 1\}$.
- It is formulated to answer the question, **"what is the probability that x_i is a positive example, i.e. $y_i = 1$?"**

Logistic (Logit) Regression

- Despite its name, **Logistic Regression** is a **classification** algorithm.
- The **labels** are binary values, $y_i \in \{0, 1\}$.
- It is formulated to answer the question, **"what is the probability that x_i is a positive example, i.e. $y_i = 1$?"**
- Just like the **Linear Regression**, the **Logistic Regression** computes a weighted sum of the input features:

$$\theta_0 + \theta_1 x_i^{(1)} + \theta_2 x_i^{(2)} + \ldots + \theta_D x_i^{(D)}$$

Logistic (Logit) Regression

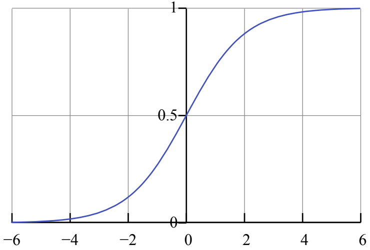
- Despite its name, **Logistic Regression** is a **classification** algorithm.
- The **labels** are binary values, $y_i \in \{0, 1\}$.
- It is formulated to answer the question, **"what is the probability that x_i is a positive example, i.e. $y_i = 1$?"**
- Just like the **Linear Regression**, the **Logistic Regression** computes a weighted sum of the input features:

$$\theta_0 + \theta_1 x_i^{(1)} + \theta_2 x_i^{(2)} + \ldots + \theta_D x_i^{(D)}$$

- The image of this function is $-\infty$ to ∞!

Logistic Regression

- In mathematics, a **standard logistic function** maps a real value ($\mathbb{R}$) to the interval $(0, 1)$:



Source: Wikipedia

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Logistic Regression

- The **Logistic Regression** model, in its vectorized form is:

$$h_\theta(x_i) = \sigma(\theta x_i) = \frac{1}{1 + e^{-\theta x_i}}$$

Logistic Regression

- The **Logistic Regression** model, in its vectorized form is:

$$h_\theta(x_i) = \sigma(\theta x_i) = \frac{1}{1 + e^{-\theta x_i}}$$

- **Predictions** are made as follows:

Logistic Regression

- The **Logistic Regression** model, in its vectorized form is:

$$h_\theta(x_i) = \sigma(\theta x_i) = \frac{1}{1 + e^{-\theta x_i}}$$

- **Predictions** are made as follows:
  - $y_i = 0$, if $h_\theta(x_i) < 0.5$

Logistic Regression

- The **Logistic Regression** model, in its vectorized form is:

$$h_\theta(x_i) = \sigma(\theta x_i) = \frac{1}{1 + e^{-\theta x_i}}$$

- **Predictions** are made as follows:
  - $y_i = 0$, if $h_\theta(x_i) < 0.5$
  - $y_i = 1$, if $h_\theta(x_i) \geq 0.5$

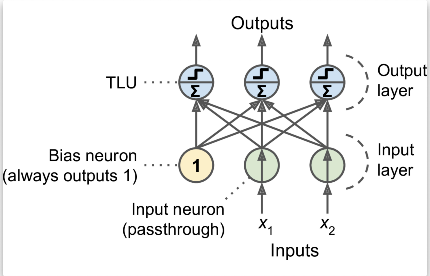
Logistic Regression

- The **Logistic Regression** model, in its vectorized form is:

$$h_\theta(x_i) = \sigma(\theta x_i) = \frac{1}{1 + e^{-\theta x_i}}$$

- **Predictions** are made as follows:
  - $y_i = 0$, if $h_\theta(x_i) < 0.5$
  - $y_i = 1$, if $h_\theta(x_i) \geq 0.5$
- The values of θ are learnt using **gradient descent**.

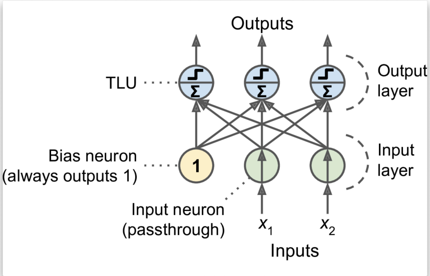
Perceptron



Source: [Géron, 2019] Figure 10.5

- A **Perceptron** consists of a single layer of threshold logic units.

Perceptron



Source: [Géron, 2019] Figure 10.5

- A **Perceptron** consists of a single layer of threshold logic units.
- It computes the following function:

$$h_{W,b}(X) = \phi(WX + b)$$

Definitions

- **Input neuron:** a special type of neuron that simply **returns the value of its input**.

Definitions

- **Input neuron:** a special type of neuron that simply **returns the value of its input**.
- **Bias neuron:** a neuron that **always return 1**.

Definitions

- **Input neuron:** a special type of neuron that simply **returns the value of its input**.
- **Bias neuron:** a neuron that **always return 1**.
- **Fully connected layer** or **dense layer: all** the neurons are connected to **all** the neurons of the previous layer.

Definitions

- **Input neuron:** a special type of neuron that simply **returns the value of its input**.
- **Bias neuron:** a neuron that **always return 1**.
- **Fully connected layer** or **dense layer: all** the neurons are connected to **all** the neurons of the previous layer.
- **X: input matrix** (rows are instances, columns are features).

Definitions

- **Input neuron:** a special type of neuron that simply **returns the value of its input**.
- **Bias neuron:** a neuron that **always return 1**.
- **Fully connected layer** or **dense layer: all** the neurons are connected to **all** the neurons of the previous layer.
- **X: input matrix** (rows are instances, columns are features).
- **W: weight matrix** (# rows corresponds to the number of inputs, # columns corresponds to the number of neurons in the output layer).

Definitions

- **Input neuron:** a special type of neuron that simply **returns the value of its input**.
- **Bias neuron:** a neuron that **always return 1**.
- **Fully connected layer** or **dense layer:** **all** the neurons are connected to **all** the neurons of the previous layer.
- **X: input matrix** (rows are instances, columns are features).
- **W: weight matrix** (# rows corresponds to the number of inputs, # columns corresponds to the number of neurons in the output layer).
- **b: bias vector** (same size as the number of neurons in the output layer).

Definitions

- **Input neuron:** a special type of neuron that simply **returns the value of its input**.
- **Bias neuron:** a neuron that **always return 1**.
- **Fully connected layer** or **dense layer: all** the neurons are connected to **all** the neurons of the previous layer.
- **X: input matrix** (rows are instances, columns are features).
- **W: weight matrix** (# rows corresponds to the number of inputs, # columns corresponds to the number of neurons in the output layer).
- **b: bias vector** (same size as the number of neurons in the output layer).
- **Activation function:** maps its input domain to a restricted set of values (heavyside and sign are commonly used with threshold logic unit perceptrons).

sklearn.linear_model.Perceptron

```python
from sklearn.linear_model import Perceptron

# ...

model = Perceptron()
model.fit(X, y)

y_pred = model.predict(X_new)
```

Activation functions

- Standard logistic (**sigmoid**) function,

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Activation functions

- Standard logistic (**sigmoid**) function,

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- **Hyperbolic tangent** function,

$$\tanh(z) = 2\sigma(2z) - 1$$

Activation functions

- Standard logistic (**sigmoid**) function,

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

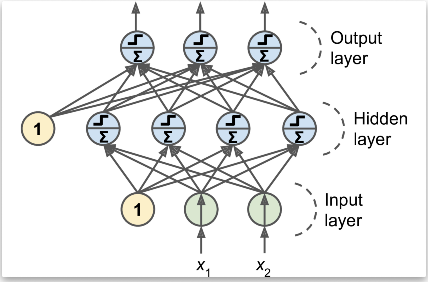
- **Hyperbolic tangent** function,

$$\tanh(z) = 2\sigma(2z) - 1$$

- Rectified Linear Unit function (**ReLU**),

$$\mathrm{ReLU}(z) = \max(0, z)$$

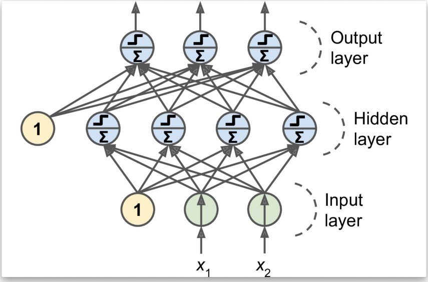
Multilayer Perceptron



Source: [Géron, 2019] Figure 10.7

- One **input layer**

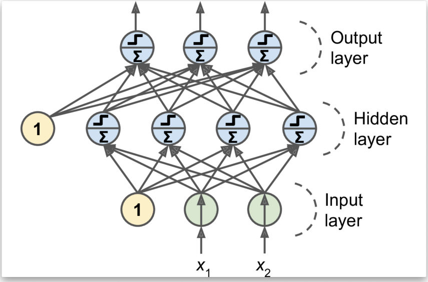
Multilayer Perceptron



Source: [Géron, 2019] Figure 10.7

- One **input layer**
- One or more **hidden layers**

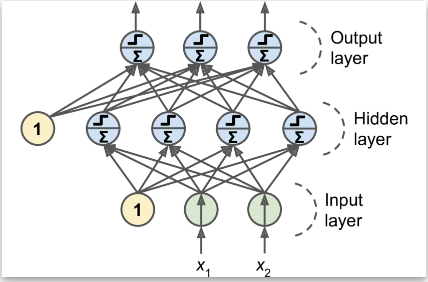
Multilayer Perceptron



Source: [Géron, 2019] Figure 10.7

- One **input layer**
- One or more **hidden layers**
- One **output layer**

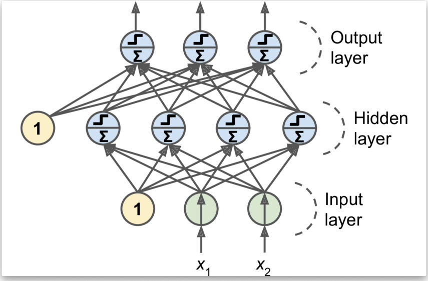
Multilayer Perceptron



Source: [Géron, 2019] Figure 10.7

- One **input layer**
- One or more **hidden layers**
- One **output layer**
- With the exception of the output layer, every layer has a **bias unit**

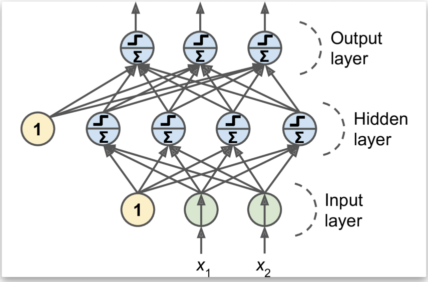
Feed-forward network (FFN)



Source: [Géron, 2019] Figure 10.7

- Since information flows from the **input to the output**, this architecture is called a **feed-forward network** (**FFN**)

Feed-forward network (FFN)



Source: [Géron, 2019] Figure 10.7

- Since information flows from the **input to the output**, this architecture is called a **feed-forward network** (**FFN**)
- Generally, FFN with more than three (3) layers are called **deep learning networks**.

Activation functions

- The **activation functions** are important, since chaining together several transformations would result into a linear transformation.

Activation functions

- The **activation functions** are important, since chaining together several transformations would result into a linear transformation.
- With these **activation functions** the network is calculating a **non-linear** function.

Backpropagation

- Introduced in 1986 by David Rumelhart, **Geoffrey Hinton**, and Ronald Williams.

Backpropagation

- Introduced in 1986 by David Rumelhart, **Geoffrey Hinton**, and Ronald Williams.
- It is a **gradient descent** algorithm with an automatic method to compute the gradients (automatic differentiation or autodiff).

Backpropagation

- Introduced in 1986 by David Rumelhart, **Geoffrey Hinton**, and Ronald Williams.
- It is a **gradient descent** algorithm with an automatic method to compute the gradients (automatic differentiation or autodiff).
- Two passes, **forward** and **backward**:

Backpropagation

- Introduced in 1986 by David Rumelhart, **Geoffrey Hinton**, and Ronald Williams.
- It is a **gradient descent** algorithm with an automatic method to compute the gradients (automatic differentiation or autodiff).
- Two passes, **forward** and **backward**:
  - **Forward.** Computes the output value(s) for a given example.

Backpropagation

- Introduced in 1986 by David Rumelhart, **Geoffrey Hinton**, and Ronald Williams.
- It is a **gradient descent** algorithm with an automatic method to compute the gradients (automatic differentiation or autodiff).
- Two passes, **forward** and **backward**:
  - **Forward.** Computes the output value(s) for a given example.
  - **Backpropagation.** Compute the necessary changes for all the weights of the model.

Backpropagation

- Introduced in 1986 by David Rumelhart, **Geoffrey Hinton**, and Ronald Williams.
- It is a **gradient descent** algorithm with an automatic method to compute the gradients (automatic differentiation or autodiff).
- Two passes, **forward** and **backward**:
  - **Forward.** Computes the output value(s) for a given example.
  - **Backpropagation.** Compute the necessary changes for all the weights of the model.
- **Repeat** the gradient descent until the network converges to a solution.

Backpropagation (2)

- Going through all the examples for every iteration would be too slow.

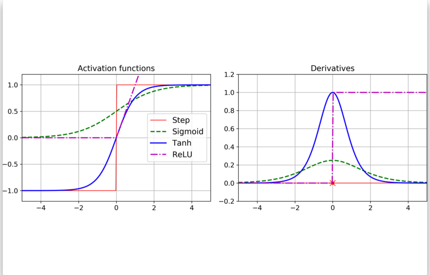
Backpropagation (2)

- Going through all the examples for every iteration would be too slow.
  - Rather, examples are grouped together into **mini-batches** (32).

Backpropagation (2)

- Going through all the examples for every iteration would be too slow.
  - Rather, examples are grouped together into **mini-batches** (32).
  - Going through all the examples is called an **epoch**.

Backpropagation (3)



Source: [Géron, 2019] Figure 10.8

Regularization

- Early stopping

Regularization

- Early stopping
- L1 and L2 norm

Regularization

- Early stopping
- L1 and L2 norm
- Dropouts

Implementations

Frameworks

Widely popular implementations include,

- **TensorFlow** (`https://www.tensorflow.org`)

Frameworks

Widely popular implementations include,

- **TensorFlow** (https://www.tensorflow.org)
- **PyTorch** (https://pytorch.org)

Frameworks

Widely popular implementations include,

- **TensorFlow** (https://www.tensorflow.org)
- **PyTorch** (https://pytorch.org)
- **Keras** (https://keras.io/) is a high-level API on the **top** of **TensorFlow**, **Theano** or **Microsoft Cognitive Toolkit (CNTK)** — these are called backends, many more backends are available.

Frameworks

Widely popular implementations include,

- **TensorFlow** (https://www.tensorflow.org)
- **PyTorch** (https://pytorch.org)
- **Keras** (https://keras.io/) is a high-level API on the **top** of **TensorFlow**, **Theano** or **Microsoft Cognitive Toolkit (CNTK)** — these are called backends, many more backends are available.

Frameworks

Widely popular implementations include,

- **TensorFlow** (`https://www.tensorflow.org`)
- **PyTorch** (`https://pytorch.org`)
- **Keras** (`https://keras.io/`) is a high-level API on the **top** of **TensorFlow**, **Theano** or **Microsoft Cognitive Toolkit (CNTK)** — these are called backends, many more backends are available.

Reading:

- Ladislav Rampasek and Anna Goldenberg, TensorFlow: Biologys gateway to deep learning?, *Cell Syst* **2**:, no. 1, 124, (2016).
- Kathleen M Chen, Evan M Cofer, Jian Zhou, and Olga G Troyanskaya, Selene: a PyTorch-based deep learning library for sequence data, *Nat Methods* **16**:4, 315318, (2019).
- Avsec, Z. et al. Kipoi: accelerating the community exchange and reuse of predictive models for genomics. *bioRxiv* **375345** (2018). doi:10.1101/375345

Suggested exercises

1. Review the Google Colab Notebook by [Zou et al., 2019]
2. Experiment with `https://playground.tensorflow.org`

Prologue

Summary

- The basic units of artificial neural networks are called **threshold logic unit** (TLU).

Summary

- The basic units of artificial neural networks are called **threshold logic unit** (TLU).
- Each **TLU** solves a problem similar to that of a **logistic regression**.

Summary

- The basic units of artificial neural networks are called **threshold logic unit** (TLU).
- Each **TLU** solves a problem similar to that of a **logistic regression**.
- An **input layer** of passthrough neurons connected to an **one output** layer of TLU forms a **Perceptron**.

Summary

- The basic units of artificial neural networks are called **threshold logic unit** (TLU).
- Each **TLU** solves a problem similar to that of a **logistic regression**.
- An **input layer** of passthrough neurons connected to an **one output** layer of TLU forms a **Perceptron**.
- In a **Perceptron**, the units of the output layer are connected to **all** the input units, forming a **dense layer**.

Summary

- The basic units of artificial neural networks are called **threshold logic unit** (TLU).
- Each **TLU** solves a problem similar to that of a **logistic regression**.
- An **input layer** of passthrough neurons connected to an **one output** layer of TLU forms a **Perceptron**.
- In a **Perceptron**, the units of the output layer are connected to **all** the input units, forming a **dense layer**.
- The **multilayer perceptron** has intermediate layers called **hidden layers**.

Summary

- The basic units of artificial neural networks are called **threshold logic unit** (TLU).
- Each **TLU** solves a problem similar to that of a **logistic regression**.
- An **input layer** of passthrough neurons connected to an **one output** layer of TLU forms a **Perceptron**.
- In a **Perceptron**, the units of the output layer are connected to **all** the input units, forming a **dense layer**.
- The **multilayer perceptron** has intermediate layers called **hidden layers**.
- **Activation functions** are playing a key role, allowing the network to compute a non-linear function of its input. This is important to model complex phenomenon.

Next module

- **Deep learning** (continued)

References

Burkov, A. (2019).
The Hundred-Page Machine Learning Book.
Andriy Burkov.

Chen, K. M., Cofer, E. M., Zhou, J., and Troyanskaya, O. G. (2019).
Selene: a PyTorch-based deep learning library for sequence data.
Nat Methods, 16(4):315–318.

Chollet, F. (2017).
Deep learning with Python.
Manning Publications.

D'haeseleer, P. (2006).
How does DNA sequence motif discovery work?
Nat Biotechnol, 24(8):959–61.

Eraslan, G., Avsec, Ž., Gagneur, J., and Theis, F. J. (2019).
Deep learning: new computational modelling techniques for genomics.
Nat Rev Genet, 20(7):389–403.

References

Géron, A. (2019).
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.
O'Reilly Media, 2nd edition.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.

Jurtz, V. I., Johansen, A. R., Nielsen, M., Almagro Armenteros, J. J., Nielsen, H., Sønderby, C. K., Winther, O., and Sønderby, S. K. (2017).
An introduction to deep learning on biological sequence data: examples and solutions.
Bioinformatics, 33(22):3685–3690.

Mahmud, M., Kaiser, M. S., Hussain, A., and Vassanelli, S. (2018).
Applications of deep learning and reinforcement learning to biological data.
IEEE Transactions on Neural Networks and Learning Systems, 29:2063–2079.

Min, S., Lee, B., and Yoon, S. (2017).
Deep learning in bioinformatics.
Brief Bioinform, 18(5):851–869.

References

Rampasek, L. and Goldenberg, A. (2016).
TensorFlow: Biology's gateway to deep learning?
Cell Syst, 2(1):12–4.

Tang, B., Pan, Z., Yin, K., and Khateeb, A. (2019).
Recent advances of deep learning in bioinformatics and computational biology.
Frontiers in Genetics, 10:214.

Webb, S. (2018).
Deep learning for biology.
Nature, 554(7693):555–557.

Zou, J., Huss, M., Abid, A., Mohammadi, P., Torkamani, A., and Telenti, A. (2019).
A primer on deep learning in genomics.
Nat Genet, 51(1):12–18.

Marcel **Turcotte**

Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EECS)

University of Ottawa