Introducing Deep Learning

Dr. Eran Raviv

July 2022

A warm welcome

Agenda for today

- Introduction + logistics

Agenda for today

- Introduction + logistics
- Deep learning overview and examples

Agenda for today

- Introduction + logistics
- Deep learning overview and examples
- Deep learning simple first steps

Agenda for today

- Introduction + logistics
- Deep learning overview and examples
- Deep learning simple first steps
- Basic concepts and terminology

A bit about myself

Trained by:

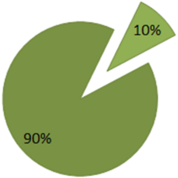
Here:

tinbergen institute

Erasmus University Rotterdam

Work for: apg

And for



Business Data Science

More on my personal blog

About this course

→ Understanding the fundamental building blocks of deep learning methods

About this course

- –> Understanding the fundamental building blocks of deep learning methods
- –> Understand different DL architectures

About this course

- –> Understanding the fundamental building blocks of deep learning methods
- –> Understand different DL architectures
- –> Gain familiarity with advanced architectures ideas and concepts

About this course

- –> Understanding the fundamental building blocks of deep learning methods
- –> Understand different DL architectures
- –> Gain familiarity with advanced architectures ideas and concepts
- –> Gain familiarity with DL programming frameworks

About this course

—> Understanding the fundamental building blocks of deep learning methods
—> Understand different DL architectures
—> Gain familiarity with advanced architectures ideas and concepts
—> Gain familiarity with DL programming frameworks
—> Ability to apply and judge DL models and performance

Logistics

- You can script in whichever language you feel comfortable with

Logistics

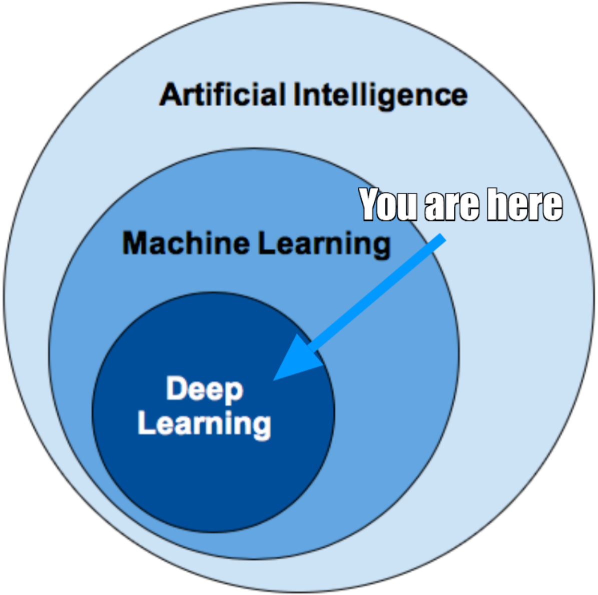
- You can script in whichever language you feel comfortable with
- Social event

Logistics

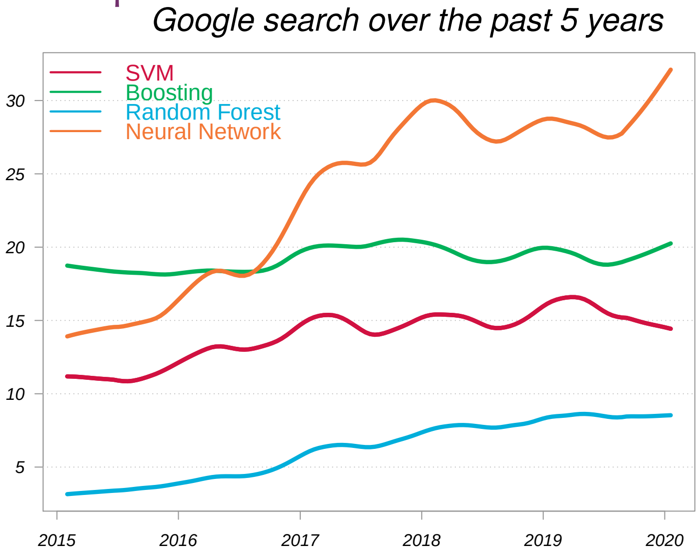
- You can script in whichever language you feel comfortable with
- Social event
- Graduation?

Introduction to Deep Learning

Coordinates

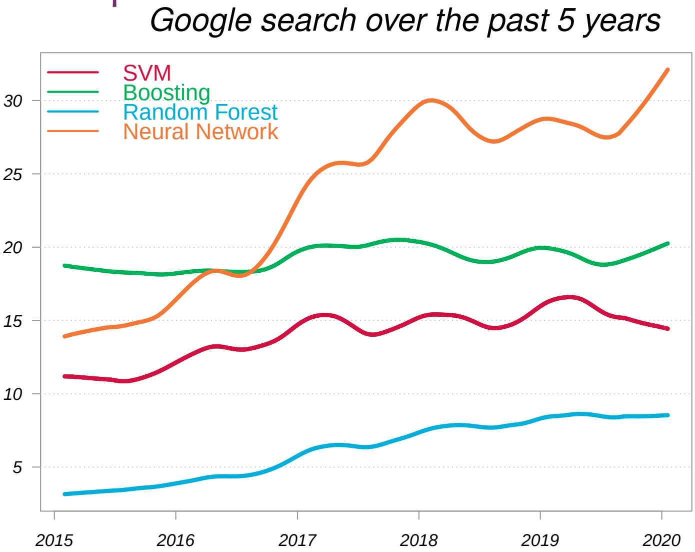


Hot topic



Google search over the past 5 years
SVM
Boosting
Random Forest
Neural Network
30
25
20
15
10
5
2015
2016
2017
2018
2019
2020

Hot topic


Google search over the past 5 years
SVM
Boosting
Random Forest
Neural Network
30
25
20
15
10
5
2015
2016
2017
2018
2019
2020

Applications galore

- Self-driving cars

Applications galore

- Self-driving cars
- Digital marketing

Applications galore

- Self-driving cars
- Digital marketing
- Videos and images editing

Applications galore

- Self-driving cars
- Digital marketing
- Videos and images editing
- Security, face recognition

Applications galore

- Self-driving cars
- Digital marketing
- Videos and images editing
- Security, face recognition
- Medical diagnosis

Applications galore

- Self-driving cars
- Digital marketing
- Videos and images editing
- Security, face recognition
- Medical diagnosis
- Banking and finance (prediction, benchmarking, default models, anomaly detection, time series prediction)

Applications galore

- Self-driving cars
- Digital marketing
- Videos and images editing
- Security, face recognition
- Medical diagnosis
- Banking and finance (prediction, benchmarking, default models, anomaly detection, time series prediction)
- Educational development

Applications galore

- Self-driving cars
- Digital marketing
- Videos and images editing
- Security, face recognition
- Medical diagnosis
- Banking and finance (prediction, benchmarking, default models, anomaly detection, time series prediction)
- Educational development
- Audio analytic (voice and audio recognition, music generation)

Applications galore

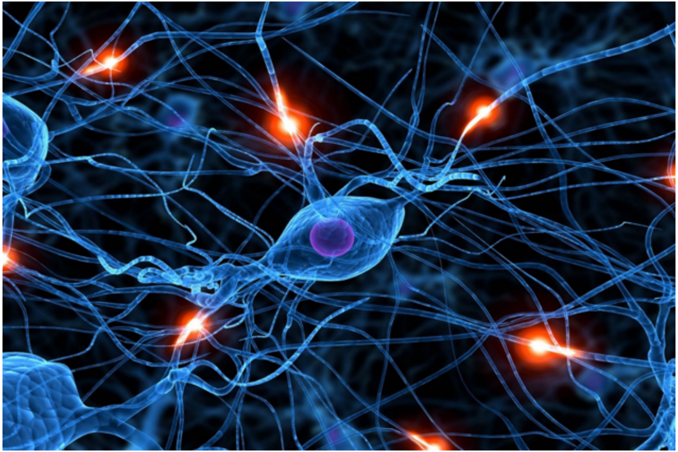
- Self-driving cars
- Digital marketing
- Videos and images editing
- Security, face recognition
- Medical diagnosis
- Banking and finance (prediction, benchmarking, default models, anomaly detection, time series prediction)
- Educational development
- Audio analytic (voice and audio recognition, music generation)
- Taking the fun out of games (Go, Texas hold'em)

Applications galore

- Self-driving cars
- Digital marketing
- Videos and images editing
- Security, face recognition
- Medical diagnosis
- Banking and finance (prediction, benchmarking, default models, anomaly detection, time series prediction)
- Educational development
- Audio analytic (voice and audio recognition, music generation)
- Taking the fun out of games (Go, Texas hold'em)
- More!

Loosely inspired by the human brain

Sensors sends information/signal. If it is strong (enough), the neuron is activated and transmitting the signal to the next neuron.



Take the basic regression setup for example

- We have the pair $(\mathbf{x}_i, y_i) \quad i = \{1, \ldots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.

Take the basic regression setup for example

- We have the pair $(\mathbf{x}_i, y_i) \quad i = \{1, \ldots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.
- We predict using $\widehat{y}_i = \mathbf{x}_i^T \mathbf{w}$

Take the basic regression setup for example

- We have the pair $(\mathbf{x}_i, y_i)$ $\quad i = \{1, \ldots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.
- We predict using $\widehat{y}_i = \mathbf{x}_i^T \mathbf{w}$
- We need to "find" good (best, in some sense) $\mathbf{w}$ such that the (quadratic in this case) risk of the loss $\mathcal{L}_i = (y_i - \widehat{y}_i)^2$ is minimal.

Take the basic regression setup for example

- We have the pair $(\mathbf{x}_i, y_i)$ $\quad i = \{1, \dots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.
- We predict using $\widehat{y}_i = \mathbf{x}_i^T \mathbf{w}$
- We need to "find" good (best, in some sense) $\mathbf{w}$ such that the (quadratic in this case) risk of the loss $\mathcal{L}_i = (y_i - \widehat{y}_i)^2$ is minimal.
- We create a mapping $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$, and $f(\cdot)$ is called the activation function.

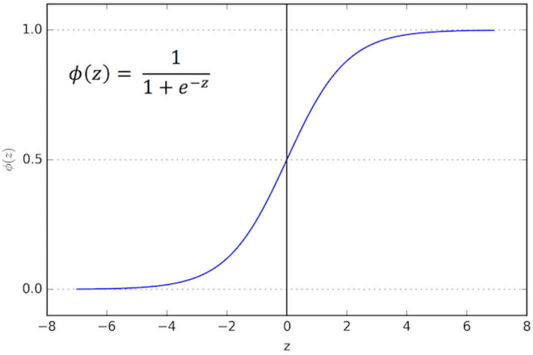
Take the basic regression setup for example

- We have the pair $(\mathbf{x}_i, y_i) \quad i = \{1, \ldots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.
- We predict using $\widehat{y}_i = \mathbf{x}_i^T \mathbf{w}$
- We need to "find" good (best, in some sense) $\mathbf{w}$ such that the (quadratic in this case) risk of the loss $\mathcal{L}_i = (y_i - \widehat{y}_i)^2$ is minimal.
- We create a mapping $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$, and $f(\cdot)$ is called the activation function.
- So a neural network with a linear activation function is simply a linear regression model.

Now with logistic regression

The logistic distribution function:

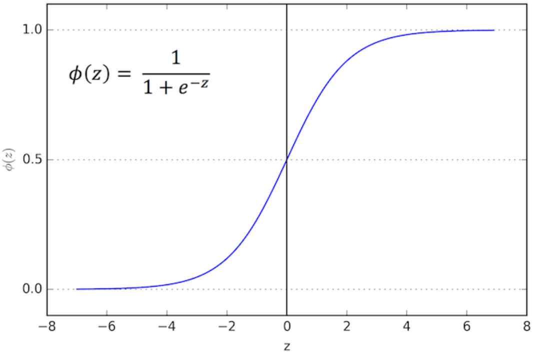
- Instead of taking the linear combination as is, we map it back to the $[0, 1]$ interval.



Now with logistic regression

The logistic distribution function:

- Instead of taking the linear combination as is, we map it back to the $[0, 1]$ interval.
- Hence, sometimes it is also referred to as a "squashing" activation function.



Now with logistic regression

- We have the pair $(\mathbf{x}_i, y_i) \quad i = \{1, \ldots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.

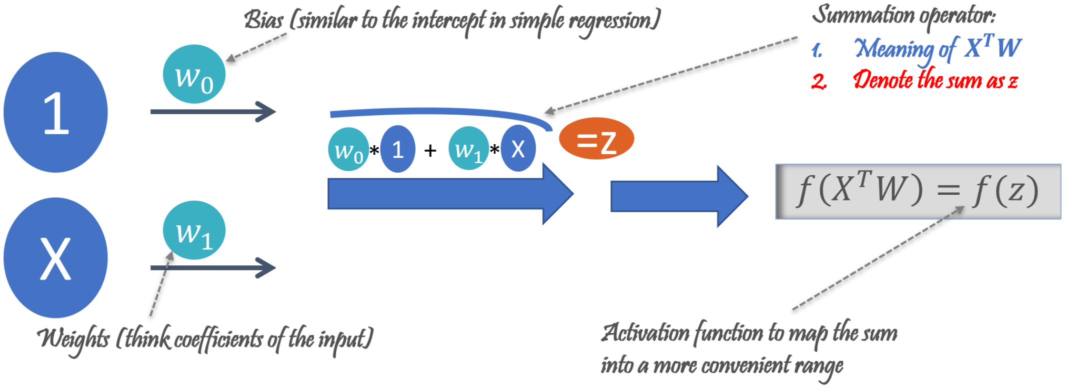
Now with logistic regression

- We have the pair $(\mathbf{x}_i, y_i)$ $\quad i = \{1, \ldots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.
- We predict using $\widehat{y}_i = f(z)$ say, $= f(\mathbf{x}_i^T \mathbf{w})$, where $f(\cdot)$ is the logistic distribution function: $f(o) = \frac{1}{1+\exp^{-o}}$.

Now with logistic regression

- We have the pair $(\mathbf{x}_i, y_i) \quad i = \{1, \ldots, m\}$ and $\mathbf{x}_i$ is $d \times 1$ vector of features, y_i is simply the target.
- We predict using $\widehat{y}_i = f(z)$ say, $= f(\mathbf{x}_i^T \mathbf{w})$, where $f(\cdot)$ is the logistic distribution function: $f(o) = \frac{1}{1+\exp^{-o}}$.
- So a (one-layer) neural network with a logistic activation function (aka sigmoid function) boils down to simply a logistic regression model.

Now with the DL jargon



What is the difference between supervised learning and unsupervised learning

What is the difference linear model, general linear model and nonlinear model?

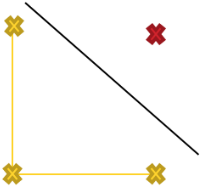
What is the main difference between traditional statistics and Machine Learning/ modern statistics?

What is meant by nonlinearity?

What are nonlinear problems?

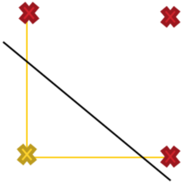
AND gate

x1	x2	Y
1	1	1
0	0	0
1	0	0
0	1	0



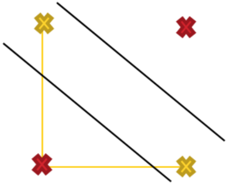
OR gate

x1	x2	Y
1	1	1
0	0	0
1	0	1
0	1	1



Exclusive (XOR): OR & !AND

x1	x2	Y
1	1	0
0	0	0
1	0	1
0	1	1



Machine learning and deep learning

- Many of existing machine learning algorithms could be represented as shallow neural network.

Machine learning and deep learning

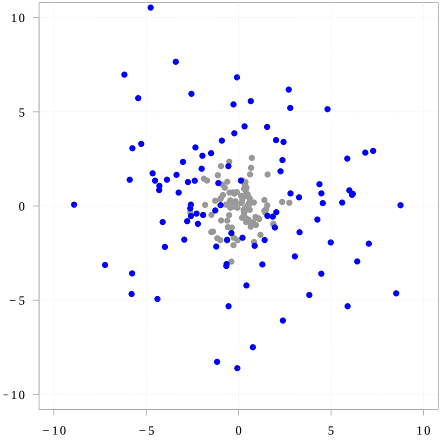
- Many of existing machine learning algorithms could be represented as shallow neural network.
- The main power of deep learning models is the ability to create a **flexible**, **expressive** models.

Machine learning and deep learning

- Many of existing machine learning algorithms could be represented as shallow neural network.
- The main power of deep learning models is the ability to create a **flexible**, **expressive** models.
- Let's see what does that mean

What is Learned in deep learning? (example)

Cartesian Coordinates

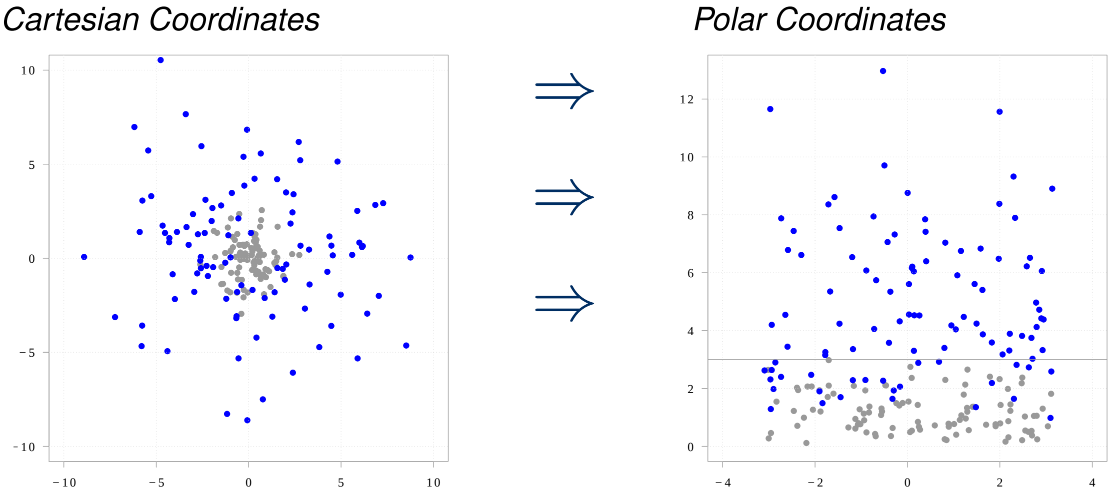


$\Rightarrow$

$\Rightarrow$

$\Rightarrow$

What is Learned in deep learning? (example)



But we want and "automatic" procedure for these transformations.

What did we do before deep learning?

- We highly valued models with a good story/rationale

What did we do before deep learning?

- We highly valued models with a good story/rationale
- We designed feature by hand

What did we do before deep learning?

- We highly valued models with a good story/rationale
- We designed feature by hand
- We had much less expressive machine learning algorithms

General concepts and formulation

In general terms

We define a predictor for the output $\mathbf{y}$ given input $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_d)$. The output could be continuous or discrete.

We learn a mapping $F : \mathbf{X} \rightarrow \mathbf{y}$, and the prediction is

In general terms

We define a predictor for the output $\mathbf{y}$ given input $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_d)$. The output could be continuous or discrete.

We learn a mapping $F : \mathbf{X} \rightarrow \mathbf{y}$, and the prediction is

$\hat{\mathbf{y}} := F(\mathbf{X})$ where each layer is

In general terms

We define a predictor for the output $\mathbf{y}$ given input $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_d)$. The output could be continuous or discrete.

We learn a mapping $F : \mathbf{X} \to \mathbf{y}$, and the prediction is

$\hat{\mathbf{y}} := F(\mathbf{X})$ where each layer is

$f_l^{W,b} := f_l \left(\sum_{j=1}^{N_l} \mathbf{w}_{lj} \mathbf{x}_j + \mathbf{b}_l \right) = f_l \left(\mathbf{W}_l \mathbf{X}_l + \mathbf{b}_l \right), \quad 1 \leq l \leq L$ and the overall prediction function is

In general terms

We define a predictor for the output $\mathbf{y}$ given input $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_d)$. The output could be continuous or discrete.

We learn a mapping $F : \mathbf{X} \to \mathbf{y}$, and the prediction is

$\hat{\mathbf{y}} := F(\mathbf{X})$ where each layer is

$f_l^{W,b} := f_l \left(\sum_{j=1}^{N_l} \mathbf{w}_{lj} \mathbf{x}_j + \mathbf{b}_l \right) = f_l \left(\mathbf{W}_l \mathbf{X}_l + \mathbf{b}_l \right), \quad 1 \leq l \leq L$ and the overall prediction function is

$\hat{\mathbf{y}}(\mathbf{X}; \theta) := F(\mathbf{X}; \theta) = \left(f_1^{\mathbf{W}_1, \mathbf{b}_1} \circ \ldots \circ f_L^{\mathbf{W}_L, \mathbf{b}_L} \right) (\mathbf{X})$ where $\theta = \{\mathbf{W}, \mathbf{B}\}$ represents the entire collection of parameters (bias parameters included).

What is so deep about deep neural networks?

- Nothing (I am so sorry about this..). There is nothing deep going on. It is a highly non-linear transformation of the inputs, but it is simple. E.g.: $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$

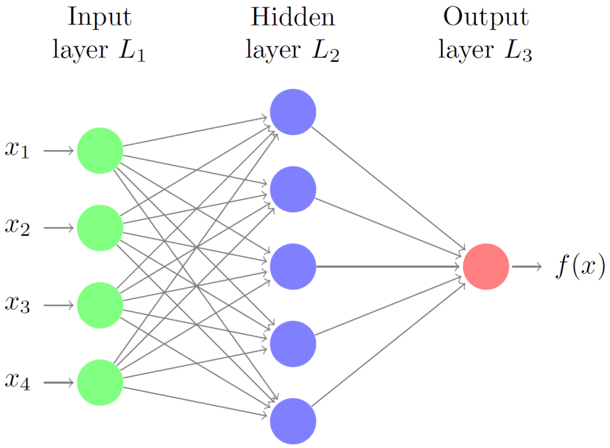
What is so deep about deep neural networks?

- Nothing (I am so sorry about this..). There is nothing deep going on. It is a highly non-linear transformation of the inputs, but it is simple. E.g.: $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$
- $f^{(l)}$ is the lth layer.

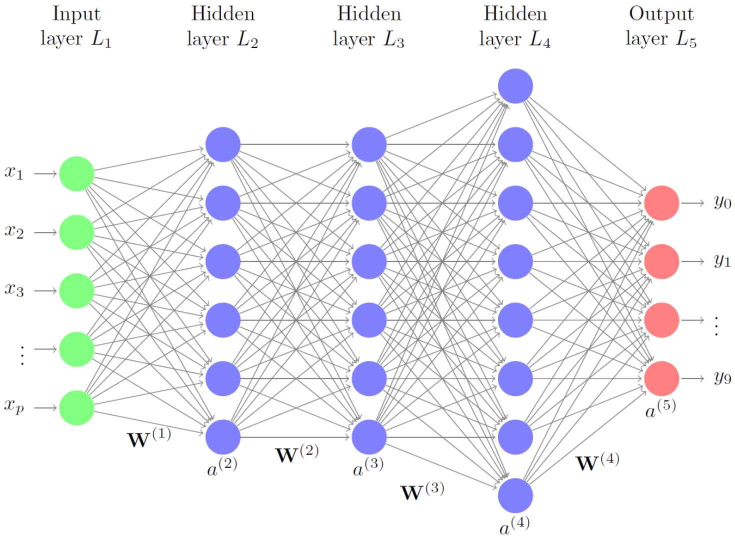
What is so deep about deep neural networks?

- Nothing (I am so sorry about this..). There is nothing deep going on. It is a highly non-linear transformation of the inputs, but it is simple. E.g.: $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$
- $f^{(l)}$ is the lth layer.
- The last layer is called the output layer.

Schematic basic neural networks



Schematic basic deep neural network



Historical example: The Perceptron

- A *linear* model for binary classification.

Historical example: The Perceptron

- A *linear* model for binary classification.
- Rosenblatt (1957)

Historical example: The Perceptron

- A *linear* model for binary classification.
- Rosenblatt (1957)
- Think logistic regression, but with a Heaviside activation function.

Historical example: The Perceptron

- A *linear* model for binary classification.
- Rosenblatt (1957)
- Think logistic regression, but with a Heaviside activation function.
- We predict using $\widehat{y}_i = f(z)$ say, $= f(\mathbf{x}_i^T \mathbf{w})$, where $f(\cdot)$ is the Heaviside step function:

$$h(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0, \end{cases}$$

Historical example: The Perceptron

- A *linear* model for binary classification.
- Rosenblatt (1957)
- Think logistic regression, but with a Heaviside activation function.
- We predict using $\hat{y}_i = f(z)$ say, $= f(\mathbf{x}_i^T \mathbf{w})$, where $f(\cdot)$ is the Heaviside step function:

$$h(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0, \end{cases}$$

- Takes input through connections with weights (while synapses pass information to other neurons)

Neurons communicate with one another at junctions called synapses

There are many other activation functions

* Heaviside step function, aka unit step function, aka binary step function:

$$h(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases}$$

* Sigmoid function, aka logistic function:

$$s(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

More activation functions

* Tanh function, aka tangent hyperbolic function:

$$t(x) = \frac{sin(x)}{cos(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2sigmoid(2x) - 1$$

* Hard Tanh

$$ht(x) = \begin{cases} -1 & : x < -1 \\ x & : -1 \leq x \leq 1 \\ 1 & : x > 1 \end{cases}$$

More activation functions

* Rectified linear units (ReLU):

$$r(x) = \max(0, x)$$

* Leaky ReLU:

$$lr(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

* Leaky ReLU is a special case of parametric ReLU.
* Exponential ReLU, aka scaled exponential:

$$elu(x) = \begin{cases} x & \text{if } x > 0 \\ a\,(e^x - 1) & \text{otherwise} \end{cases}$$

More activation functions

* Linear:

$$l(x) = ax \qquad \text{e.g. } a = 1$$

* Swish:

$$sw(x) = x sigmoid(x)$$

* There are more.
* More are coming.

Why so many? What are the advantages/disadvantages of each?

Cost- loss-, error-, objective-, and risk-function

Sometimes different definitions for different purposes. Here we use the following:

- Loss function, or error function is the a function of a single error:

$$\mathcal{L}(\widehat{y}_i, y_i) = \mathcal{L}(\widehat{y}_i - y_i) = \mathcal{L}(e)$$

Cost- loss-, error-, objective-, and risk-function

Sometimes different definitions for different purposes. Here we use the following:

- Loss function, or error function is the a function of a single error:

$$\mathcal{L}(\widehat{y}_i, y_i) = \mathcal{L}(\widehat{y}_i - y_i) = \mathcal{L}(e)$$

- Cost function is used to describe the entire cumulative error for the entire sample:

$$\mathcal{J}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\widehat{y}_i, y_i)$$

Cost- loss-, error-, objective-, and risk-function

- Risk is the expectation of the cost function. It is a "theoretical" quantity, which we estimate from the data.

$$\mathcal{R}(\mathbf{w}) = \mathrm{E}\left(\mathcal{J}(\mathbf{w})\right)$$

Cost- loss-, error-, objective-, and risk-function

- Risk is the expectation of the cost function. It is a "theoretical" quantity, which we estimate from the data.

$$\mathcal{R}(\mathbf{w}) = \mathrm{E}\left(\mathcal{J}(\mathbf{w})\right)$$

- The risk function is a type of objective function. Objective function is a general term describing what you would like to optimize for.

Recap

What have we covered today?

- DL are pure-prediction algorithms

Recap

What have we covered today?

- DL are pure-prediction algorithms
- Representation learning

Recap

What have we covered today?

- DL are pure-prediction algorithms
- Representation learning
- Neurons/hidden units

Recap

What have we covered today?

- DL are pure-prediction algorithms
- Representation learning
- Neurons/hidden units
- Layers

Recap

What have we covered today?

- DL are pure-prediction algorithms
- Representation learning
- Neurons/hidden units
- Layers
- Architecture

Recap

What have we covered today?

- DL are pure-prediction algorithms
- Representation learning
- Neurons/hidden units
- Layers
- Architecture
- Activation functions

Recap

What have we covered today?

- DL are pure-prediction algorithms
- Representation learning
- Neurons/hidden units
- Layers
- Architecture
- Activation functions
- Loss/cost/error/objective functions

Multinomial logistic regression

- AKA: softmax Regression, Maximum Entropy Classifier, Multi-class Logistic Regression, softmax classifier.

Multinomial logistic regression

- AKA: softmax Regression, Maximum Entropy Classifier, Multi-class Logistic Regression, softmax classifier.
- Generalizes logistic regression to multi-class classification (under the assumption that the classes are mutually exclusive).

Multinomial logistic regression

- AKA: softmax Regression, Maximum Entropy Classifier, Multi-class Logistic Regression, softmax classifier.
- Generalizes logistic regression to multi-class classification (under the assumption that the classes are mutually exclusive).
- The softmax function thus provides a “softened” version of the argmax (winner takes all..).

Multinomial logistic regression

- AKA: softmax Regression, Maximum Entropy Classifier, Multi-class Logistic Regression, softmax classifier.
- Generalizes logistic regression to multi-class classification (under the assumption that the classes are mutually exclusive).
- The softmax function thus provides a “softened” version of the argmax (winner takes all..).

  It would perhaps be better to call the softmax function “softargmax”, but the current name is an entrenched convention.

Multinomial logistic regression

$$P(y = k|x, W) = \text{softmax}_k(Wx) = \frac{\exp\left(w_k x\right)}{\sum_{j=1}^{j=k} \exp\left(w_j x\right)}$$

$$J(W) = -\left[\sum_{i=1}^{m} \sum_{k=1}^{k} 1 \left\{y^{(i)} = k\right\} \log \frac{\exp(w_k x)}{\sum_{j=1}^{j=1} \exp\left(w_j x\right)}\right],$$

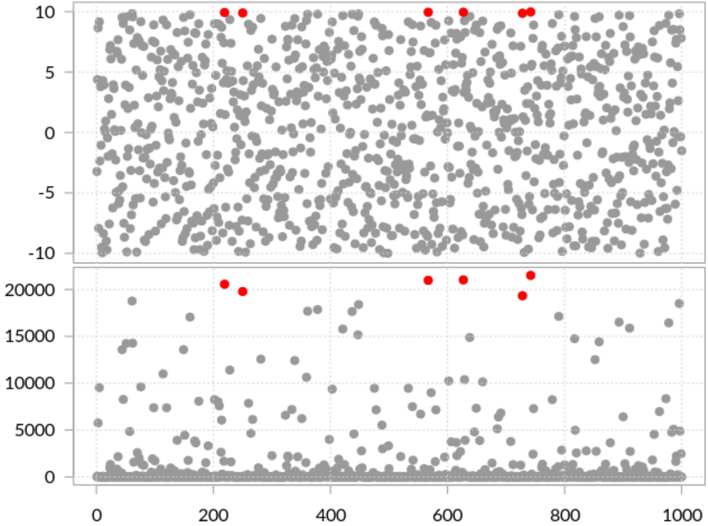
$$\mathcal{I}\left\{y^{(i)} = k\right\} = \begin{cases} 1, & \text{if } y^{(i)} = k \\ 0, & \text{otherwise} \end{cases}$$

$$w := w - \eta \Delta w$$

$$\Delta w_k = \frac{\partial}{\partial w_j} J(w) = -\sum_{i=1}^{m} \left[x^{(i)} \left(1 \left\{y^{(i)} = k\right\} - P\left(y^{(i)} = k|x^{(i)}, W\right)\right)\right]$$

Softmax

What does the softmax function do?



Cross entropy loss

$$J = -\sum_i p_i \log q_i$$

where you can think of q_i as the observable of category i and p_i is the calculated probability for that category.

- Classification error is a crude measure. Disregard level of confidence.

Cross entropy loss

$$J = -\sum_i p_i \log q_i$$

where you can think of q_i as the observable of category i and p_i is the calculated probability for that category.

- Classification error is a crude measure. Disregard level of confidence.
- Still, two competing models could have opposite ranking of the cross entropy loss and classification error.

Cross entropy loss

$$J = -\sum_i p_i \log q_i$$

where you can think of q_i as the observable of category i and p_i is the calculated probability for that category.

- Classification error is a crude measure. Disregard level of confidence.
- Still, two competing models could have opposite ranking of the cross entropy loss and classification error.
- Sometimes it is useful to look at both.

Now Let's code!

References

Deep Learning - main textbook

Deep Learning: A Practitioner's Approach

Neural Networks and Learning Machines

Neural Networks with R

Neural Networks and Deep Learning

Advanced Deep Learning with R

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828.

Heaton, J., Polson, N., and Witte, J. H. (2017). Deep learning for finance: deep portfolios. Applied Stochastic Models in Business and Industry, 33(1):3–12.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017a). Searching for activation functions. arXiv preprint arXiv:1710.05941.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017b). Swish: a self-gated activation function. arXiv preprint arXiv:1710.05941, 7.

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.