Artificial neural networks

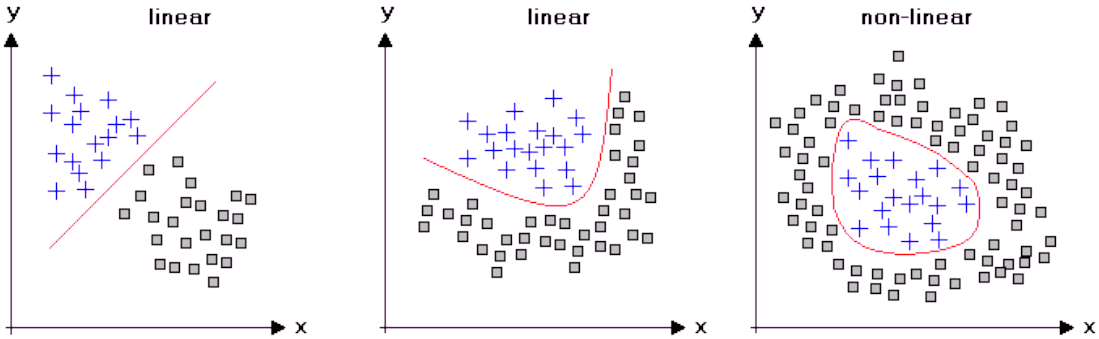
Olga Razuvaeva[1, 2], Sergey Korpachev[3, 4] and Stepan Zakharov [5]

[1]Institute for Theoretical and Experimental Physics
[2]National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
[3]Moscow Institute of Physics and Technology
[4]Lebedev Physical Institute of the Russian Academy of Sciences
[5]Novosibirsk State University

Outline

- Non-linearity in data
- Feature extraction
- ANN structure
- How to train?
- Evolution of GD
- Deep Learning
- The Achilles heel of the DL models
- Reaching stability
- Network architectures

Non-linearity in data



Do you know how to describe the non-linear data in the right picture?

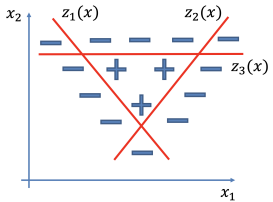
Linear model, seriously?

Source

Other link

Feature extraction

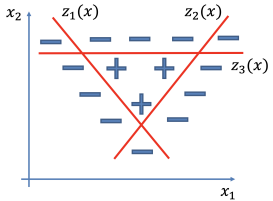
- Consider some simpler data



Source

Feature extraction

- Consider some simpler data

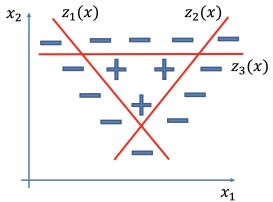


- Let's try to describe it with a linear model $z_i(\boldsymbol{x}) = \boldsymbol{\omega}_{0,i} + \boldsymbol{\omega}_{1,i} \cdot \boldsymbol{x}_1 + \boldsymbol{\omega}_{2,i} \cdot \boldsymbol{x}_2$

Source

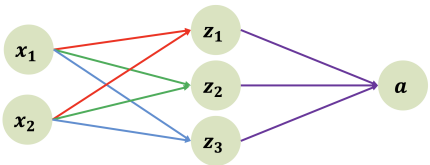
Feature extraction

- Consider some simpler data



- Let's try to describe it with a linear model $z_i(\boldsymbol{x}) = \boldsymbol{\omega}_{0,i} + \boldsymbol{\omega}_{1,i} \cdot \boldsymbol{x}_1 + \boldsymbol{\omega}_{2,i} \cdot \boldsymbol{x}_2$

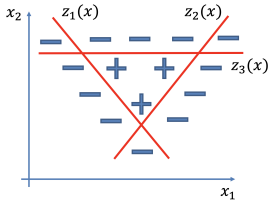
- Let's rewrite new variables $z_i(\boldsymbol{x})$ in the form of a computational graph



Source

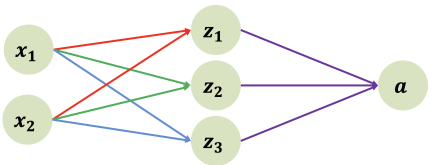
Feature extraction

- Consider some simpler data



- Let's try to describe it with a linear model $z_i(\boldsymbol{x}) = \omega_{0,i} + \omega_{1,i} \cdot x_1 + \omega_{2,i} \cdot x_2$

- Let's rewrite new variables $z_i(\boldsymbol{x})$ in the form of a computational graph

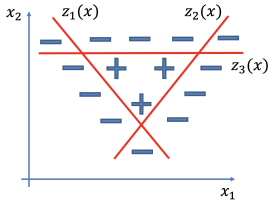


- Now our model is:
$a(\boldsymbol{x}) = k_0 + k_1 \cdot (\omega_{0,1} + \omega_{1,1} \cdot x_1 + \omega_{2,1} \cdot x_2) + k_2 \cdot (\omega_{0,2} + \omega_{1,2} \cdot x_1 + \omega_{2,2} \cdot x_2) + k_3 \cdot (\omega_{0,3} + \omega_{1,3} \cdot x_1 + \omega_{2,3} \cdot x_2)$

Source

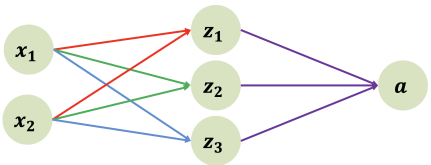
Feature extraction

- Consider some simpler data



- Let's try to describe it with a linear model $z_i(\boldsymbol{x}) = \omega_{0,i} + \omega_{1,i} \cdot x_1 + \omega_{2,i} \cdot x_2$

- Let's rewrite new variables $z_i(\boldsymbol{x})$ in the form of a computational graph



- Now our model is:
  $a(\boldsymbol{x}) = k_0 + k_1 \cdot (\omega_{0,1} + \omega_{1,1} \cdot x_1 + \omega_{2,1} \cdot x_2) + k_2 \cdot (\omega_{0,2} + \omega_{1,2} \cdot x_1 + \omega_{2,2} \cdot x_2) + k_3 \cdot (\omega_{0,3} + \omega_{1,3} \cdot x_1 + \omega_{2,3} \cdot x_2)$
- The model is linear, how to create new features?

Source

Computation graph

- Add non-linearity to our model to create new features

Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$

Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$
- A characteristic of an artificial neural network (ANN) is the presence of a superposition of neurons with a non-linear activation function

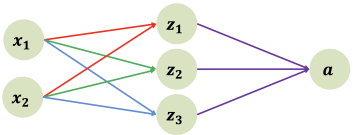
Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$
- A characteristic of an artificial neural network (ANN) is the presence of a superposition of neurons with a non-linear activation function
- Finally, our model is: $a(x) = \sigma\big(k_0 + k_1 * z_1(x) + k_2 * z_2(x) + k_3 * z_3(x)\big)$

Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$
- A characteristic of an artificial neural network (ANN) is the presence of a superposition of neurons with a non-linear activation function
- Finally, our model is: $a(x) = \sigma\big(k_0 + k_1 * z_1(x) + k_2 * z_2(x) + k_3 * z_3(x)\big)$

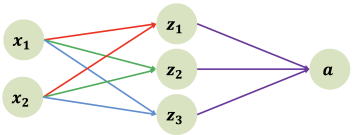
- Consider the structure of our graph



Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$
- A characteristic of an artificial neural network (ANN) is the presence of a superposition of neurons with a non-linear activation function
- Finally, our model is: $a(x) = \sigma\big(k_0 + k_1 * z_1(x) + k_2 * z_2(x) + k_3 * z_3(x)\big)$

- Consider the structure of our graph

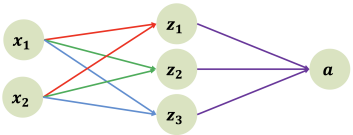


- nodes (vertices) x_1 and x_2 - features from an input layer of the ANN

Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$
- A characteristic of an artificial neural network (ANN) is the presence of a superposition of neurons with a non-linear activation function
- Finally, our model is: $a(x) = \sigma\big(k_0 + k_1 * z_1(x) + k_2 * z_2(x) + k_3 * z_3(x)\big)$

- Consider the structure of our graph

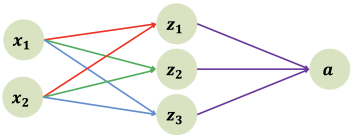


- nodes (vertices) x_1 and x_2 - features from an input layer of the ANN
- nodes (vertices) z_1, z_2 and z_3 - neurons from a hidden layer of the ANN

Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$
- A characteristic of an artificial neural network (ANN) is the presence of a superposition of neurons with a non-linear activation function
- Finally, our model is: $a(x) = \sigma\big(k_0 + k_1 * z_1(x) + k_2 * z_2(x) + k_3 * z_3(x)\big)$

- Consider the structure of our graph

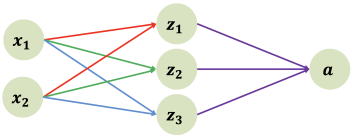


- nodes (vertices) x_1 and x_2 - features from an input layer of the ANN
- nodes (vertices) z_1, z_2 and z_3 - neurons from a hidden layer of the ANN
- node (vertex) a - neuron from a output layer of the ANN

Computation graph

- Add non-linearity to our model to create new features
- Now we have: $a(x) = \sigma\big(k_0 + k_1 * \sigma(\omega_{0,1} + \omega_{1,1} * x_1 + \omega_{2,1} * x_2) + k_2 * \sigma(\omega_{0,2} + \omega_{1,2} * x_1 + \omega_{2,2} * x_2) + k_3 * \sigma(\omega_{0,3} + \omega_{1,3} * x_1 + \omega_{2,3} * x_2)\big)$
- A characteristic of an artificial neural network (ANN) is the presence of a superposition of neurons with a non-linear activation function
- Finally, our model is: $a(x) = \sigma\big(k_0 + k_1 * z_1(x) + k_2 * z_2(x) + k_3 * z_3(x)\big)$

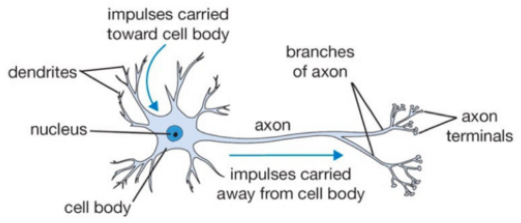
- Consider the structure of our graph

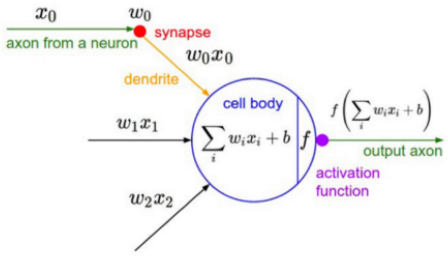


- nodes (vertices) x_1 and x_2 - features from an input layer of the ANN
- nodes (vertices) z_1, z_2 and z_3 - neurons from a hidden layer of the ANN
- node (vertex) a - neuron from a output layer of the ANN
- Straight lines (edges) correspond to weights between neurons

Human brain

It follows the way of human brain processing. ANN modelled how a neuron works in the human brain. So, that's why it brought into the world of computer science.





Neurons on human brain consist of a nucleus, dendrites, cell body, and axon. The number of neurons in humans is approximately 140 billion, consist of 100 billion neurons and 40 billion synapses in neurons. Neurons used to send an impulse to other connected neurons. An impulse is a receptor which getting from sense organs. Source

How to train?

- We know how to train one neuron (e.g. logistic regression): GD or SGD.
- Let's do the same for the whole network!

GD and SGD: recap

Gradient Descent:

$$\theta_{i+1} = \theta_i - \eta \cdot \nabla Q(\theta_i)$$

- Pro: Fewer oscillations and noisy steps
- Pro: It produces a more stable gradient descent convergence
- Con: A stable error gradient can lead to a local minimum and unlike stochastic gradient descent no noisy steps are there to help get out of the local minimum
- Con: It can take too long for processing all the training sample

Stochastic Gradient Descent:

$$\theta_{i+1} = \theta_i - \eta \cdot \nabla Q_j(\theta_i)$$

- Pro: It is computationally fast
- Pro: There are oscillations which can help to get out of local minimums of the loss function
- Con: It may take longer to achieve convergence to the minimum of the loss function
- Con: Oscillations can lead the gradient descent into other directions

Complex function

How to find the derivative of a complex function?

Chain rule

- We know how to find the derivative of a simple function.
- Let's remember the rule for differentiating complex functions.

$$\frac{\partial f\big(t(x)\big)}{\partial x} = \frac{\partial f(t)}{\partial t(x)} \cdot \frac{\partial t(x)}{\partial x}$$

- What is the derivative of following function with respect to x: $sin(x^2 + 5)$?
- What is the derivative of the sigmoid function?

  Can you convert it to $\sigma'(x) = f\big(\sigma(x)\big)$?

- Sigmoid function: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

Chain rule

- We know how to find the derivative of a simple function.
- Let's remember the rule for differentiating complex functions.

$$\frac{\partial f\big(t(x)\big)}{\partial x} = \frac{\partial f(t)}{\partial t(x)} \cdot \frac{\partial t(x)}{\partial x}$$

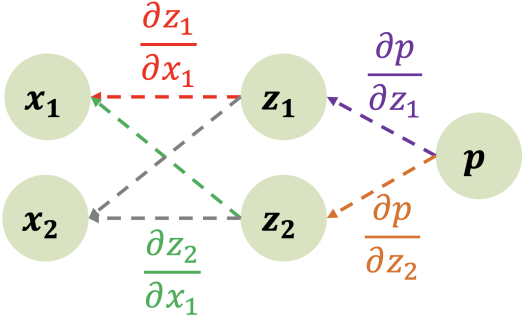
- What is the derivative of following function with respect to x: $sin(x^2 + 5)$?
- What is the derivative of the sigmoid function?

  Can you convert it to $\sigma'(x) = f\big(\sigma(x)\big)$?

- Sigmoid function: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

Answer: $\sigma'(x) = \sigma(x) \cdot \big(1 - \sigma(x)\big)$

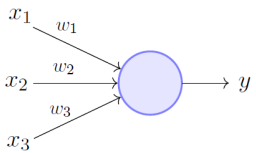
Chain rule for ANN



$$\frac{\partial p}{\partial x_i} = \frac{\partial p}{\partial z_1} \cdot \frac{\partial z_1}{\partial x_i} + \frac{\partial p_2}{\partial x_i} \cdot \frac{\partial z_2}{\partial x_i},$$

where x_i is x_1 or x_2

Back propagation



Perceptron Model (Minsky-Papert in 1969)

$$\overset{\text{Source}}{Q\big(y_{model}, y_{target}\big)} = Q\big(y(x, w, b), y_{target}\big)$$

$$out = y = \sigma\big(sum\big) = \sigma\Big(\sum_{i=1}^{3}(x_i \cdot w_i) + b\Big)$$

$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial sum} \cdot \frac{\partial sum}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot 1 \cdot \frac{\partial \sigma}{\partial sum} \cdot \frac{\partial sum}{\partial w_i}$$

$$\frac{\partial \sigma}{\partial sum} = \sigma(sum) \cdot \big(1 - \sigma(sum)\big)$$

Back propagation: an example

$$Q\big(y(x, w, b), y_{target}\big)$$

$$Q = \big(y_{target} - y(x_1, w_1, b)\big)^2$$

$$out = y = (x \cdot w) + b$$

$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w_i}$$

Input: x = 1, w = 0.1 and b = 1

$y_{target} = 2$ and learning rate: $\eta = 0.1$

Back propagation: an example

$$Q\big(y(x, w, b), y_{target}\big)$$
$$Q = \big(y_{target} - y(x_1, w_1, b)\big)^2$$
$$out = y = (x \cdot w) + b$$
$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w_i}$$

Input: x = 1, w = 0.1 and b = 1

$y_{target} = 2$ and learning rate: $\eta = 0.1$

How to find w_{new} and b_{new}?

$$\frac{\partial Q}{\partial y} = -2 \cdot (y_{target} - y)$$
$$\frac{\partial Q}{\partial w} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w} = -2 \cdot x \cdot (y_{target} - y)$$
$$\frac{\partial Q}{\partial b} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial b} = -2 \cdot (y_{target} - y)$$
$$[w_{new}, b_{new}] \Rightarrow \theta_{i+1} = \theta_i - \eta \cdot \frac{\partial Q(\theta_i)}{\partial \theta_i}$$

Back propagation: an example

$$Q\big(y(x, w, b), y_{target}\big)$$
$$Q = \big(y_{target} - y(x_1, w_1, b)\big)^2$$
$$out = y = (x \cdot w) + b$$
$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w_i}$$

Input: x = 1, w = 0.1 and b = 1

$y_{target} = 2$ and learning rate: $\eta = 0.1$

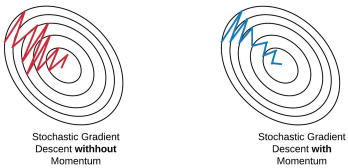
How to find w_{new} and b_{new}?

$$\frac{\partial Q}{\partial y} = -2 \cdot (y_{target} - y)$$
$$\frac{\partial Q}{\partial w} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w} = -2 \cdot x \cdot (y_{target} - y)$$
$$\frac{\partial Q}{\partial b} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial b} = -2 \cdot (y_{target} - y)$$
$$[w_{new}, b_{new}] \Rightarrow \theta_{i+1} = \theta_i - \eta \cdot \frac{\partial Q(\theta_i)}{\partial \theta_i}$$

y	Q	$\frac{\partial Q}{\partial y}$	$\frac{\partial Q}{\partial w}$	$\frac{\partial Q}{\partial b}$	w_{new}	b_{new}
1.1	0.81	-1.8	-1.8	-1.8	0.28	1.18

Momentum and Root Mean Square Propagation (RMSProp)

SGD with momentum:

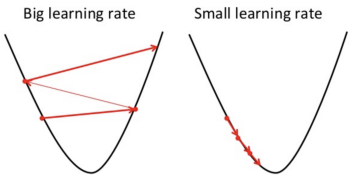


Source

$$\theta_{i+1} = \theta_i - \Delta\theta_{i+1}$$
$$\Delta\theta_{i+1} = \alpha \cdot \Delta\theta_i + \eta \cdot \nabla Q_j(\theta_i)$$

- α is an exponential decay factor between 0 and 1 that determines the relative contribution of the current gradient and earlier gradients to the weight change

RMSProp:



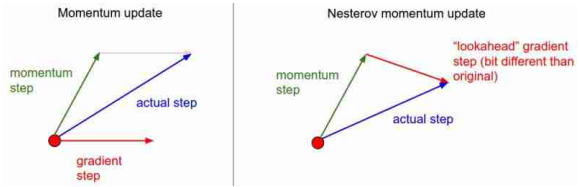
Source

$$\theta_{i+1} = \theta_i - \frac{\eta}{\sqrt{v_{i+1} + \epsilon} + \epsilon} \cdot \nabla Q_j(\theta_i)$$
$$v_{i+1} = \rho \cdot v_i + (1 - \rho) \cdot \left(\nabla Q_j(\theta_i)\right)^2$$

- ρ is moving average parameter (the forgetting factor)

Nesterov momentum and Adaptive Moment Estimation (Adam)

SGD with nesterov momentum:



Source

$$\theta_{i+1} = \theta_i - \Delta\theta_{i+1}$$
$$\Delta\theta_{i+1} = \alpha \cdot \Delta\theta_i + \eta \cdot \nabla Q_j(\theta_i - \alpha \cdot \Delta\theta_i)$$

Adam and Nadam:

- Adam optimizer is essentially RMSProp with momentum
- Nadam optimizer is Adam with Nesterov momentum

Break time = this is big brain time

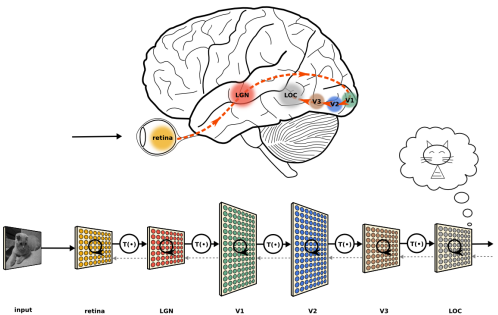
- Is it necessary to use a bias neuron and why?
- What values can the weights of neurons take?

Deep Learning

!!ADD LINKS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

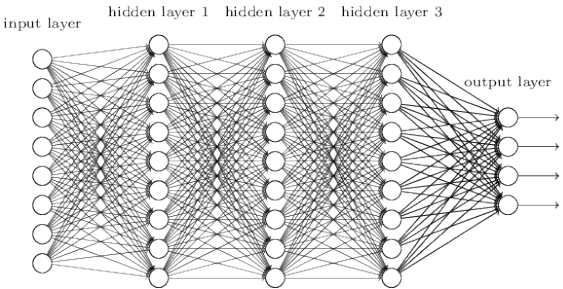
Motivation

Let's take a look at the human brain



- Visual signals travel through multiple areas of a different organization
- This makes our visual system incredibly robust

The same approach is valid for ANN:

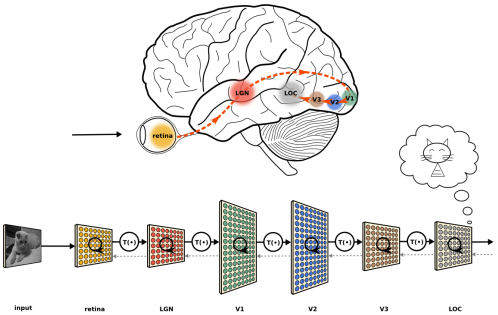


Source

Deep Learning ⇔ complex models?

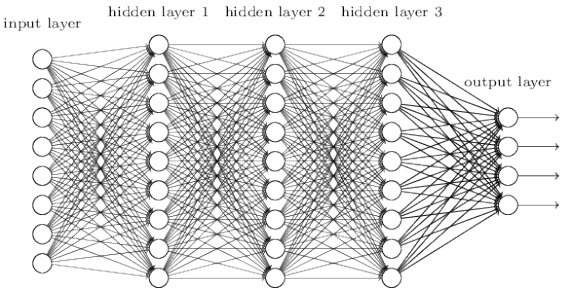
Motivation

Let's take a look at the human brain



- Visual signals travel through multiple areas of a different organization
- This makes our visual system incredibly robust

The same approach is valid for ANN:



Source

Deep Learning ⇔ complex models?

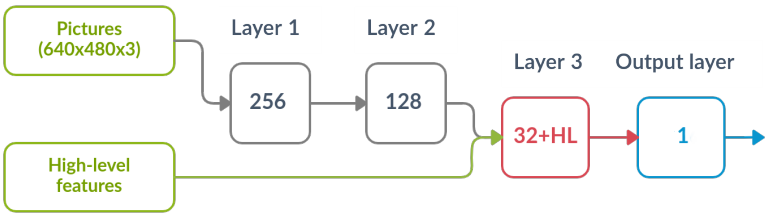
Not exactly!

The meaning of "Deep"

- Dummy explanation refers to a number of layers ($N_\ell \geq 3 \Rightarrow$ DNN)
- More profound viewpoint:
  Deep learning is a deep understanding of data structure
- Here is an example:

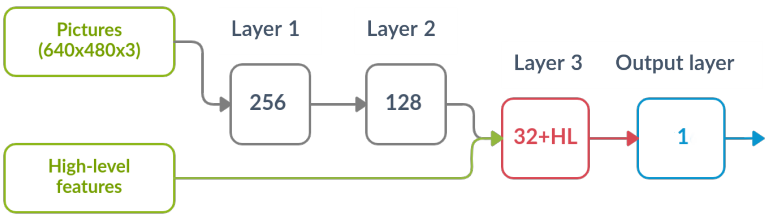
The meaning of "Deep"

- Dummy explanation refers to a number of layers ($N_\ell \geq 3 \Rightarrow$ DNN)
- More profound viewpoint:
  Deep learning is a deep understanding of data structure
- Here is an example:



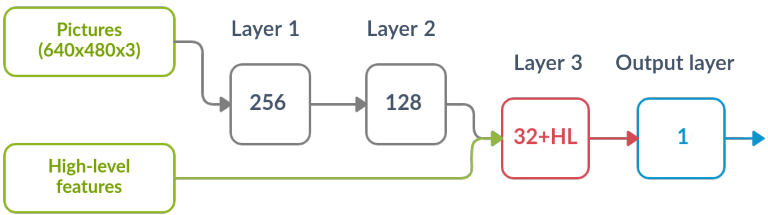
The meaning of "Deep"

- Dummy explanation refers to a number of layers ($N_\ell \geq 3 \Rightarrow$ DNN)
- More profound viewpoint:
  Deep learning is a deep understanding of data structure
- Here is an example:
- Input data contain high and low-level features



The meaning of "Deep"

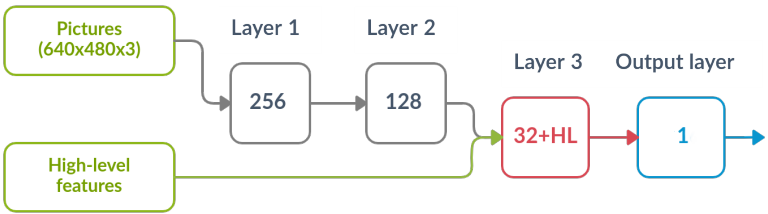
- Dummy explanation refers to a number of layers ($N_\ell \geq 3 \Rightarrow$ DNN)
- More profound viewpoint:
  Deep learning is a deep understanding of data structure
- Here is an example:
- Input data contain high and low-level features
- We don't know the exact set of features and need to invent them



The meaning of "Deep"

- Dummy explanation refers to a number of layers ($N_\ell \geq 3 \Rightarrow$ DNN)
- More profound viewpoint:
  Deep learning is a deep understanding of data structure
- Here is an example:

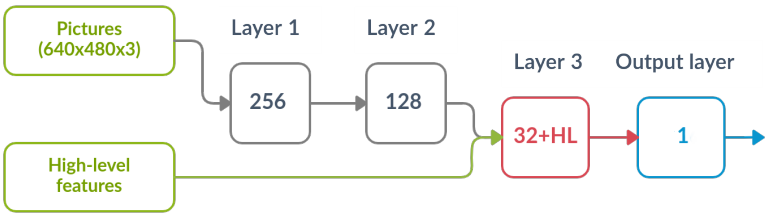
- Input data contain high and low-level features
- We don't know the exact set of features and need to invent them
- ANN has lots of inputs



The meaning of "Deep"

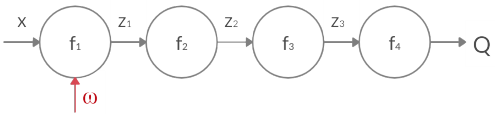
- Dummy explanation refers to a number of layers ($N_\ell \geq 3 \Rightarrow$ DNN)
- More profound viewpoint:
  Deep learning is a deep understanding of data structure
- Here is an example:

- Input data contain high and low-level features
- We don't know the exact set of features and need to invent them
- ANN has lots of inputs
- See the next lecture on more complex ANN



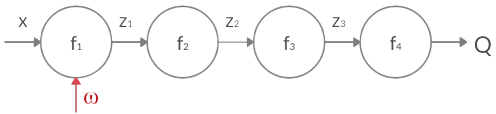
The Achilles heel of the DL models

- Train a DNN with the SGD method



The Achilles heel of the DL models

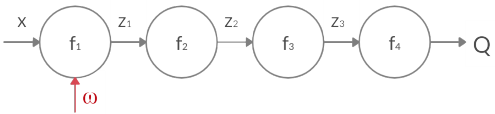
- Train a DNN with the SGD method
- Layer $f_i(z_{i-1})$ takes the output z_{i-1} from the previous layer and returns z_i



The Achilles heel of the DL models

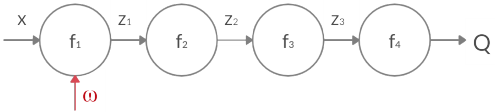
- Train a DNN with the SGD method
- Layer $f_i(z_{i-1})$ takes the output z_{i-1} from the previous layer and returns z_i
- Making backprop we calculate:

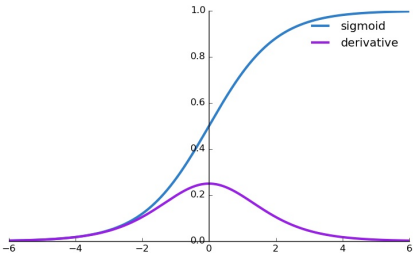
$$\frac{\partial Q}{\partial \omega_j} = \frac{\partial Q}{\partial f_i}\frac{\partial f_i}{\partial f_{i-1}} \cdots \frac{\partial f_1}{\partial \omega_j}$$



The Achilles heel of the DL models

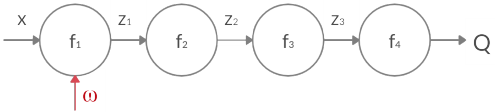
- Train a DNN with the SGD method
- Layer $f_i(z_{i-1})$ takes the output z_{i-1} from the previous layer and returns z_i
- Making backprop we calculate:
$$\frac{\partial Q}{\partial \omega_j} = \frac{\partial Q}{\partial f_i}\frac{\partial f_i}{\partial f_{i-1}}\cdots\frac{\partial f_1}{\partial \omega_j}$$
- Here $\frac{\partial f_i}{\partial f_{i-1}} = \sigma(z_{i-1})\big(1-\sigma(z_{i-1})\big)$

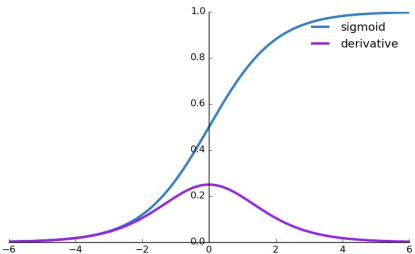




The Achilles heel of the DL models

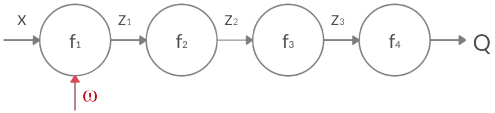
- Train a DNN with the SGD method
- Layer $f_i(z_{i-1})$ takes the output z_{i-1} from the previous layer and returns z_i
- Making backprop we calculate:
$$\frac{\partial Q}{\partial \omega_j} = \frac{\partial Q}{\partial f_i}\frac{\partial f_i}{\partial f_{i-1}}\cdots\frac{\partial f_1}{\partial \omega_j}$$
- Here $\dfrac{\partial f_i}{\partial f_{i-1}} = \sigma(z_{i-1})\big(1-\sigma(z_{i-1})\big)$
- $\left|\dfrac{\partial f_i}{\partial f_{i-1}}\right| \leq \dfrac{1}{4} \Rightarrow \Delta\omega_i \to 0$

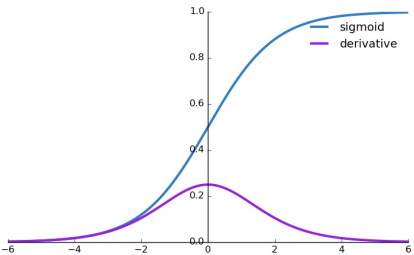




The Achilles heel of the DL models

- Train a DNN with the SGD method
- Layer $f_i(\boldsymbol{z_{i-1}})$ takes the output $\boldsymbol{z_{i-1}}$ from the previous layer and returns $\boldsymbol{z_i}$
- Making backprop we calculate:
$$\frac{\partial Q}{\partial \omega_j} = \frac{\partial Q}{\partial f_i}\frac{\partial f_i}{\partial f_{i-1}}\cdots\frac{\partial f_1}{\partial \omega_j}$$
- Here $\dfrac{\partial f_i}{\partial f_{i-1}} = \sigma(z_{i-1})\big(1-\sigma(z_{i-1})\big)$
- $\left|\dfrac{\partial f_i}{\partial f_{i-1}}\right| \leq \dfrac{1}{4} \Rightarrow \Delta\omega_i \to 0$

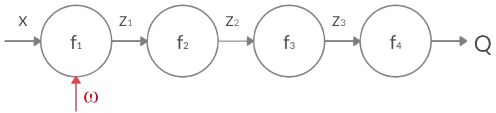




Problem: Gradient step vanishes for the initial layers ⇒ undertraining!

The Achilles heel of the DL models

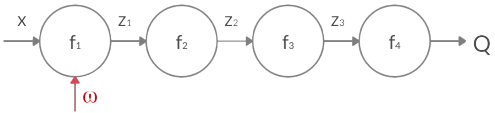
An artificial example:



The Achilles heel of the DL models

An artificial example:

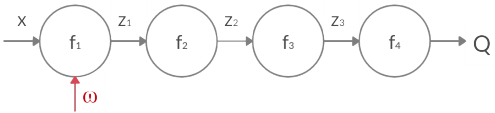
- Initialize all $\omega_j = 100$ (bad)



The Achilles heel of the DL models

An artificial example:

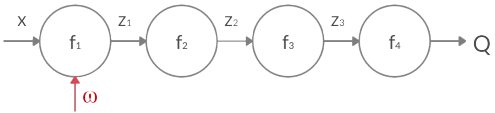
- Initialize all $\omega_j = 100$ (bad)
- Choose all biases that all $z_i = 0$



The Achilles heel of the DL models

An artificial example:

- Initialize all $\omega_j = 100$ (bad)
- Choose all biases that all $z_i = 0$
- Then each $\dfrac{\partial f_i}{\partial f_{i-1}} = \dfrac{1}{4}$

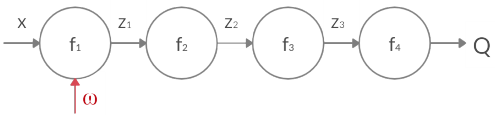


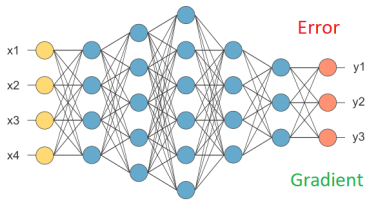
The Achilles heel of the DL models

An artificial example:

- Initialize all $\omega_j = 100$ (bad)
- Choose all biases that all $z_i = 0$
- Then each $\dfrac{\partial f_i}{\partial f_{i-1}} = \dfrac{1}{4}$
- $\Delta\omega_j = \left(100 \cdot \dfrac{1}{4}\right)^{N-j+1}$,

  where N is a total number of layers



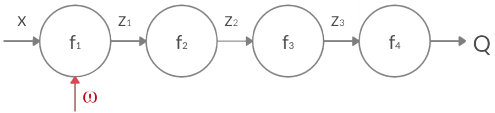


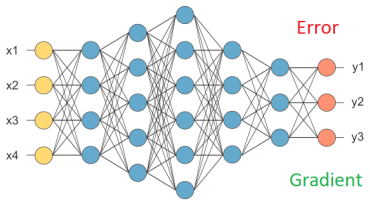
The Achilles heel of the DL models

An artificial example:

- Initialize all $\omega_j = 100$ (bad)
- Choose all biases that all $z_i = 0$
- Then each $\dfrac{\partial f_i}{\partial f_{i-1}} = \dfrac{1}{4}$
- $\Delta\omega_j = \left(100 \cdot \dfrac{1}{4}\right)^{N-j+1}$,

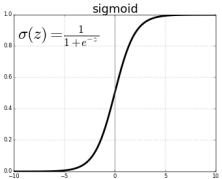
  where N is a total number of layers
- Exploding and vanishing gradients evidence the problem of model instability

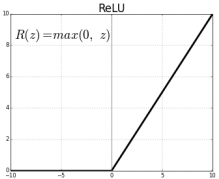




Reaching stability

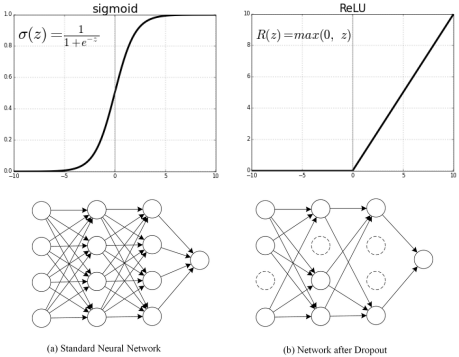
- Different activation functions (ReLU)





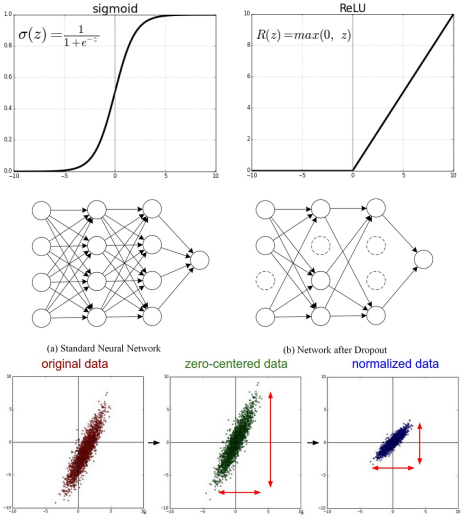
Reaching stability

- Different activation functions (ReLU)
- Regularization of network architecture



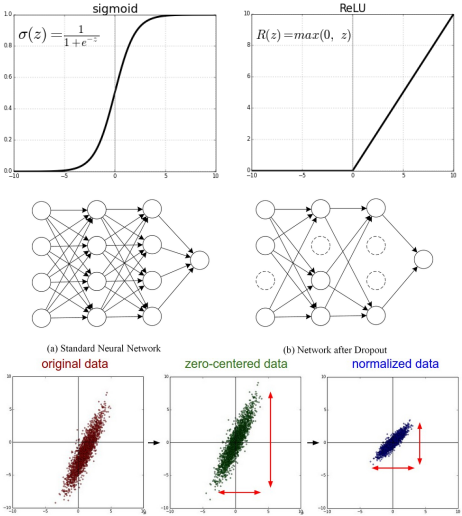
Reaching stability

- Different activation functions (ReLU)
- Regularization of network architecture
- Normalization:
  - Input variables (data preprocessing)
  - Batch normalization



Reaching stability

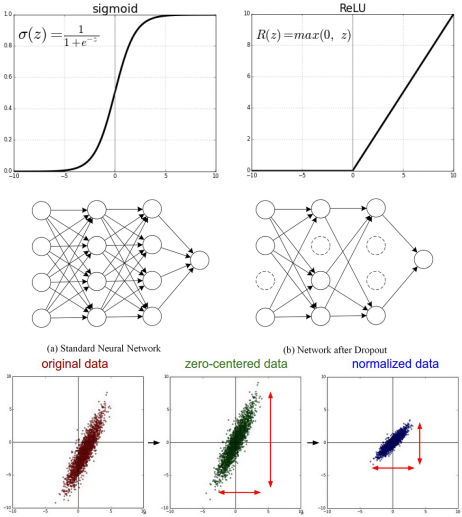
- Different activation functions (ReLU)
- Regularization of network architecture
- Normalization:
  - Input variables (data preprocessing)
  - Batch normalization
- Different training methods



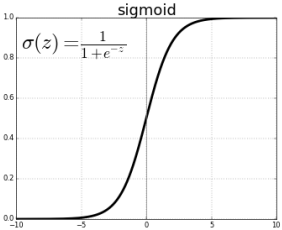
Reaching stability

- Different activation functions (ReLU)
- Regularization of network architecture
- Normalization:
  - Input variables (data preprocessing)
  - Batch normalization
- Different training methods

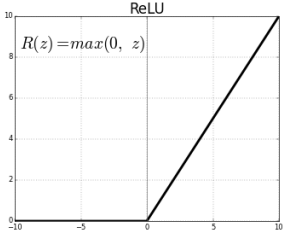
We'll partially cover them at this lecture



Activation functions: Sigmoid and Rectified Linear Unit (ReLU)

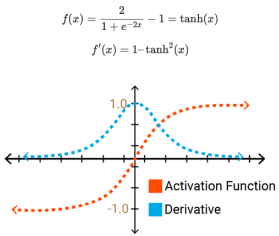


- Differentiable
- Shrinks parameter domain into $[0, 1]$
- $|\sigma'(x)| \leq \frac{1}{4}$

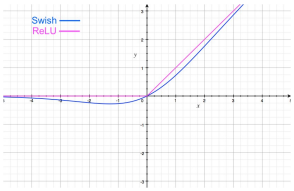


- Non-differentiable at 0, but we still can define $R'(0) := 0$ or 1
- Shrinks parameter domain into $\mathbb{R}^+$
- $R'(x) \in \{0, 1\}$
- There are different variations of ReLU: Leaky ReLU, Parametric ReLU (PReLU), Exponential ReLU (ELU), ...

Activation functions: Tanh and Swish



- Hyperbolic Tangent Function:
  $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$
- Differentiable
- Shrinks parameter domain into $[-1, 1]$
- $|\tanh'(x)| \leq 1$



- $\text{swish}(x) = \dfrac{x}{1 + e^{-\beta \cdot x}} = x \cdot \sigma(\beta \cdot x)$
- Differentiable
- $\beta \to 1 \Rightarrow$ Sigmoid Linear Unit (SiLU)
- $\beta \to 0 \Rightarrow \text{swish}(x) = \dfrac{x}{2}$
- $\beta \to \infty \Rightarrow \text{swish}(x) \sim \mathit{ReLU}$

Regularization: two common methods

L2 regularization (Tikhonov)

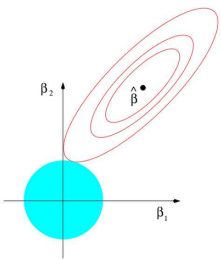
$$Q(a) = \frac{1}{N}\sum_{j=1}^{N}\left(\langle \boldsymbol{x}, \boldsymbol{\beta}\rangle - y_j\right)^2 + \lambda \sum_{i=1}^{K}(\boldsymbol{\beta}_i)^2$$

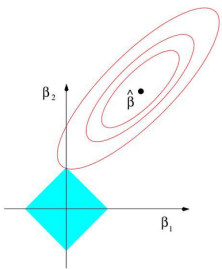
L1 regularization (LASSO)

least absolute shrinkage and selection operator

$$Q(a) = \frac{1}{N}\sum_{j=1}^{N}\left(\langle \boldsymbol{x}, \boldsymbol{\beta}\rangle - y_j\right)^2 + \lambda \sum_{i=1}^{K}|\boldsymbol{\beta}_i|$$

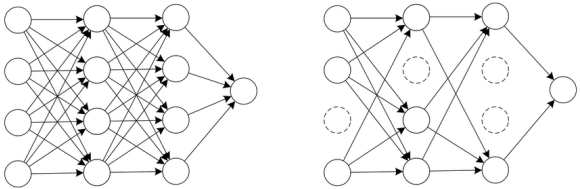
The red term brings up a penalty when the model includes lots of terms or the weights are too high





Regularization: Dropout

- Initialize a fully connected network
- With probability p switch off neurons
- Train this architecture
- Repeat for several configurations and take average results for weights ω
- Apply weights to a fully connected network and modify activation function: $a'(x, \omega) = (1 - p)a(x, \omega)$



(a) Standard Neural Network

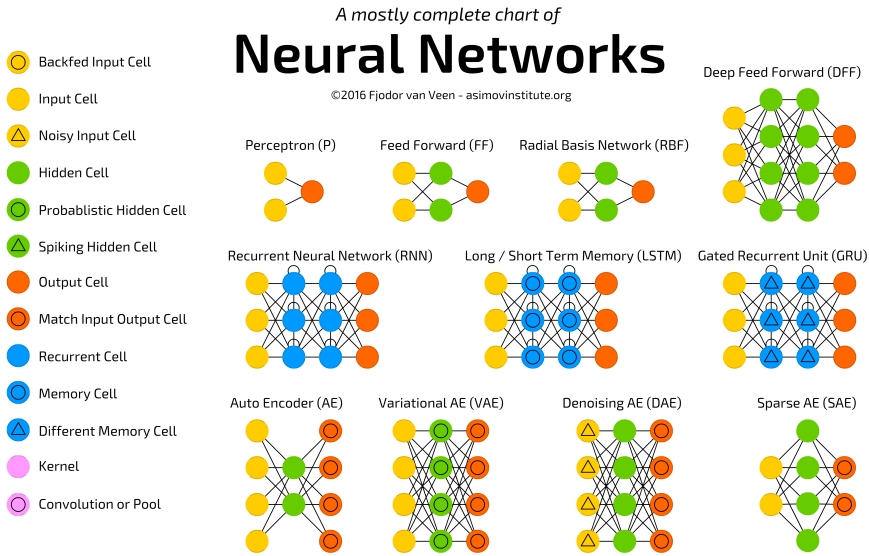
(b) Network after Dropout

Dropout prevents complex co-adaptations of neurons and provides reasonable improvement of ANN results.
Original article

ANN architectures' examples

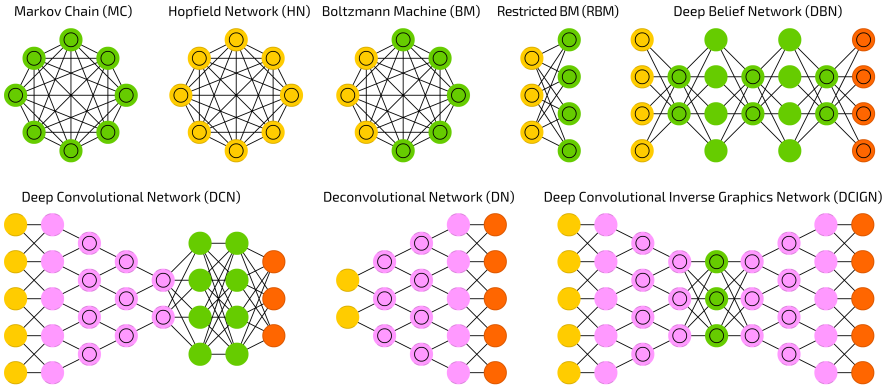
Source

Network architectures

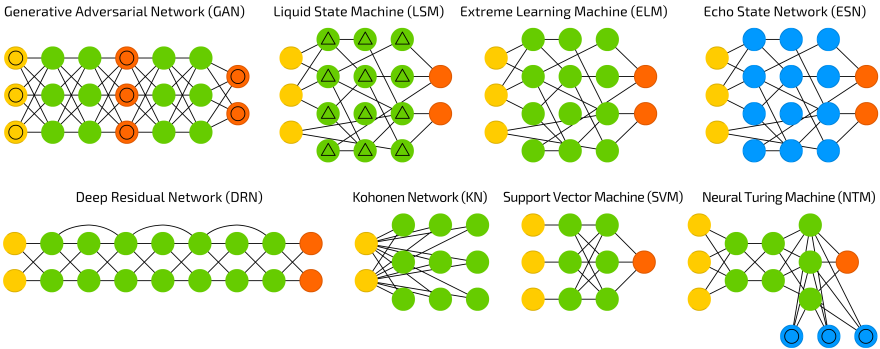


Network architectures

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool



Network architectures



Visual pathway of the human brain