Applied Machine Learning for Business Analytics
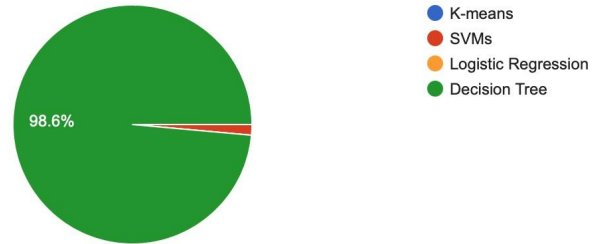
Lecture 3: Neural Networks and Deep Learning

Lecturer: Zhao Rui

Logistics

- Next Friday, Sanjay Saha will give a tutorial on the deep learning library: Keras after the lecture
- We will finalize the grouping information during this weekend
  - If your group ID is odd number,   Sanjay will be your mentor
  - If your group ID is even number,   Xiaohui will be your mentor
- The survey for learning mode has been sent out.
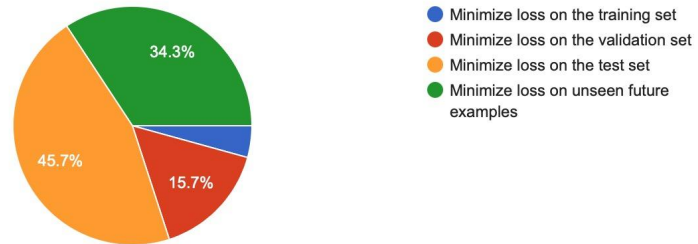- Appreciate if you keeps video on!

Which of these following machine learning models is not sensitive to scales of input features?
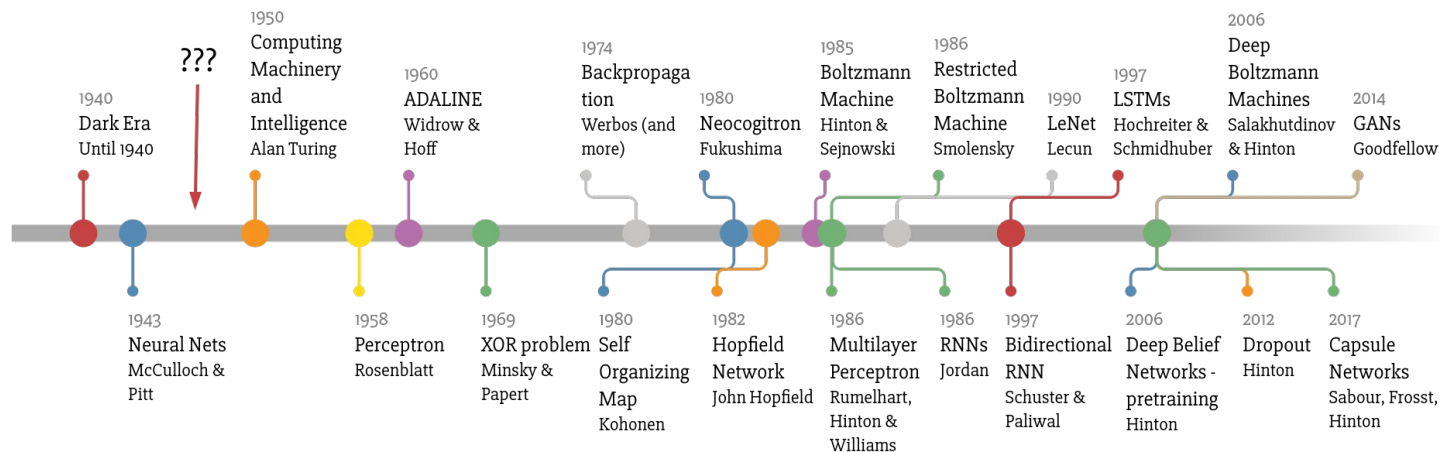
70 responses

- K-means
- SVMs
- Logistic Regression
- Decision Tree

98.6%

What should be the targets set for machine learning?

70 responses

- Minimize loss on the training set
- Minimize loss on the validation set
- Minimize loss on the test set
- Minimize loss on unseen future examples

34.3%

45.7%

15.7%

DL/NN is not New

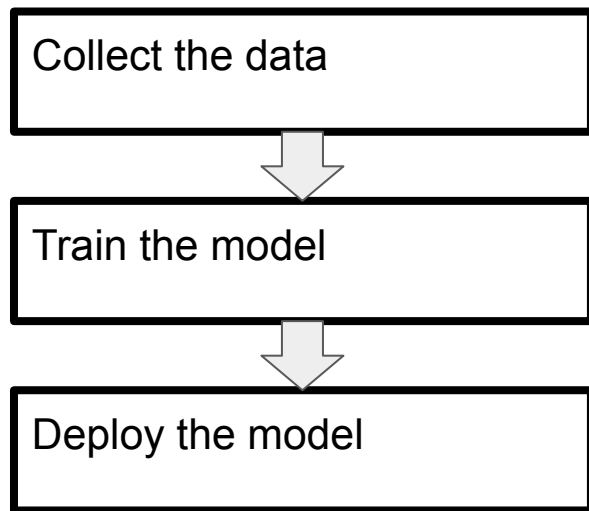## Deep Learning Timeline

Made by Favio Vázquez

Why DL is Powerful Now?

- Feature engineering require high-level expert knowledge, which are easily over-specified and incomplete.
- Large amounts of training data
- Modern multi-core CPUs/GPUs/TPUs
- Better deep learning 'tricks' such as regularization, optimization, transfer learning etc.

Deep Learning Myth: Three Steps

- To deploy deep learning (or other machine learning) systems

```
┌─────────────────────────┐
│ Collect the data        │
└─────────────────────────┘
            ↓
┌─────────────────────────┐
│ Train the model         │
└─────────────────────────┘
            ↓
┌─────────────────────────┐
│ Deploy the model        │
└─────────────────────────┘
```

The Truth

1. Select a metric for optimization 💪
2. Collect data 🙇‍♀️
3. Train model 👨‍💻
4. Realize many labels are wrong 😱
5. Relabel data ✍️
6. Train model 👨‍💻
7. Model performs poorly on one class 🤦
8. Collect more data for that class 🙇
9. Train model 👨‍💻
10. Model performs poorly on most recent data 🤦
11. Collect more recent data 🙇
12. Train model 👨‍💻
13. Deploy model 👷‍♀️
14. Dream 🤑
15. Get a call at 3am about complaints that model is biased 👱
16. Revert to the older version
17. Collect more data, do more training and testing 👨‍💻
18. Deploy model 👷‍♀️
19. Pray 🙆
20. Model performs well but revenue decreasing 🤷
21. Cry 😭
22. Choose a different metric 💪
23. Start over 👶

Three Steps in Deep Learning

To approximate the true function, define a function **space**

Neural network structures

Learning Representation

Need a **measure** to evaluate the quality of each potential function in the previous space
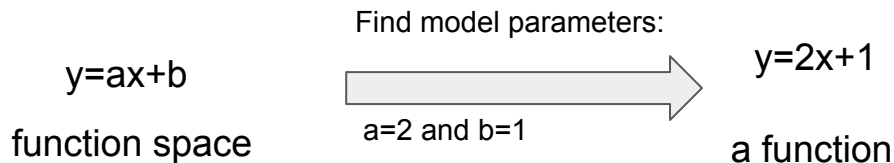
Evaluate the function

Objective Function

Search the function space to find the best function based on the measure.

Pick the best function

Optimization

Find model parameters:

y=ax+b

a=2 and b=1

y=2x+1

function space

a function

Agenda

1. Linear Regression
2. Neural Networks
3. Evaluation of Functions
4. Optimization
5. Deep Learning
6. Application of DL

1. Linear Regression

Linear Regression (Single Variable)

- Model architecture: y=ax+b
- Objective function: Mean Squared Error Function

$$J(a, b) = \frac{1}{n} \sum_{i=0}^{n} (y_i - (ax_i + b))^2$$
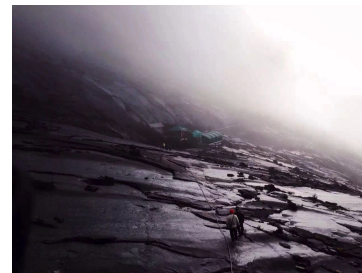
- Optimization: Gradient Descent Algorithm

Gradient Descent
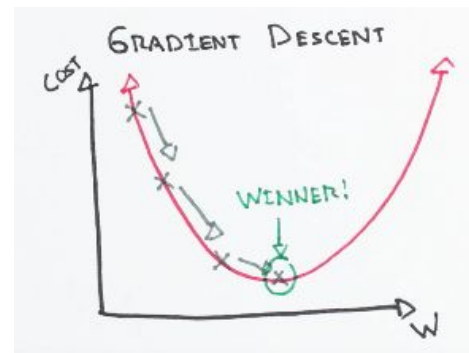
Gradient for the total loss function over parameters,

$$\theta_{t+1} = \theta_t - \alpha \triangledown J(\theta_t)$$

New
Parameters
Guess

Current
Parameters
Guess

Learning
Rate

Like hiking down a mountain

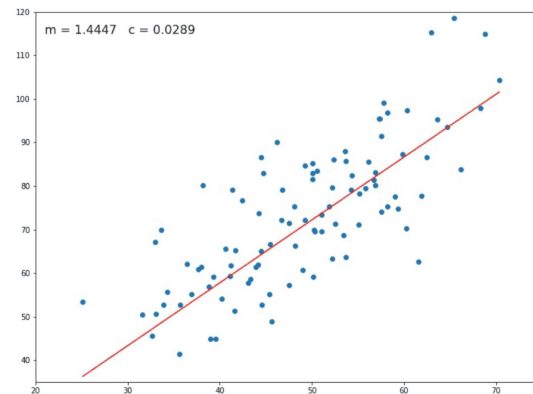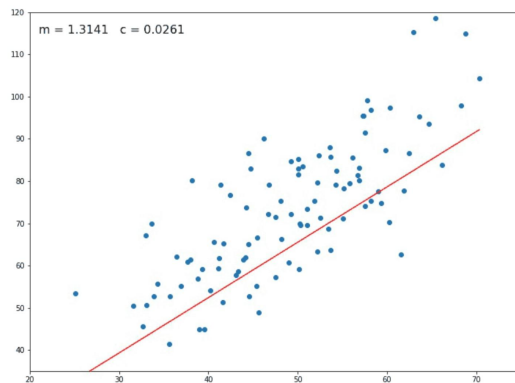Credit: https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html
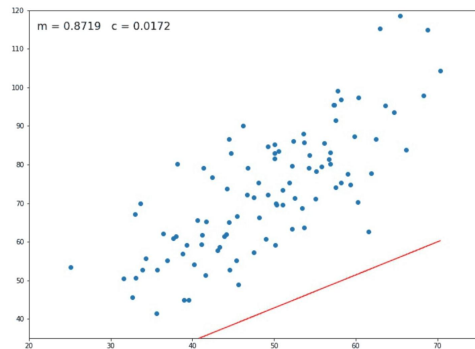
Simple Math

- Gradients for parameters:

$$\frac{\partial}{\partial a} J(a, b) = \frac{1}{n} \sum_{i=0}^{n} 2(y_i - (ax_i + b))(-x_i)$$

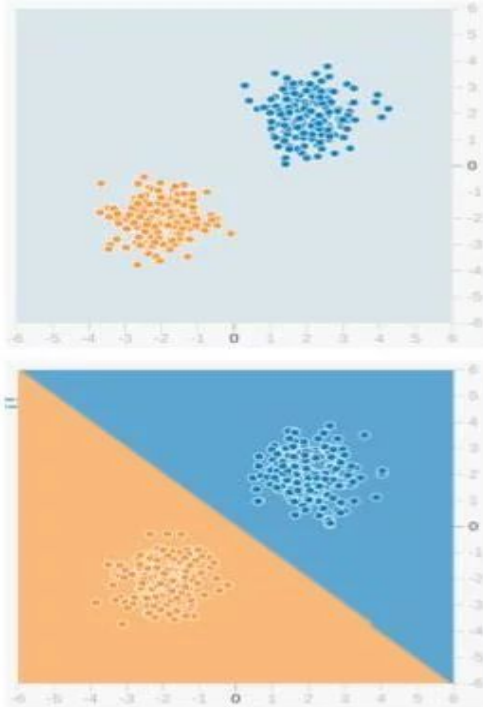$$\frac{\partial}{\partial a} J(a, b) = \frac{-2}{n} \sum_{i=0}^{n} (y_i - y_i^{'})x_i$$

$$\frac{\partial}{\partial b} J(a, b) = \frac{-2}{n} \sum_{i=0}^{n} (y_i - y_i^{'})$$

Optimization

2. Neural Networks
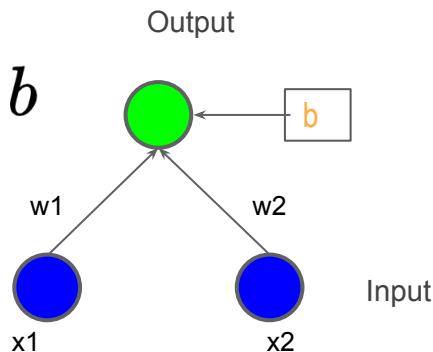
A "Simple" Classification Problem

$$y = \mathbf{w}\mathbf{x} + b$$

A Linear Model

- Linear Regression if output is continuous
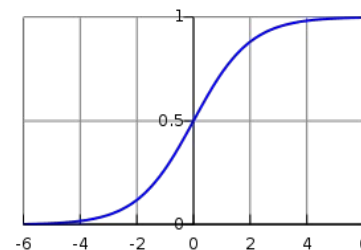
- Logistic Regression if output is discrete

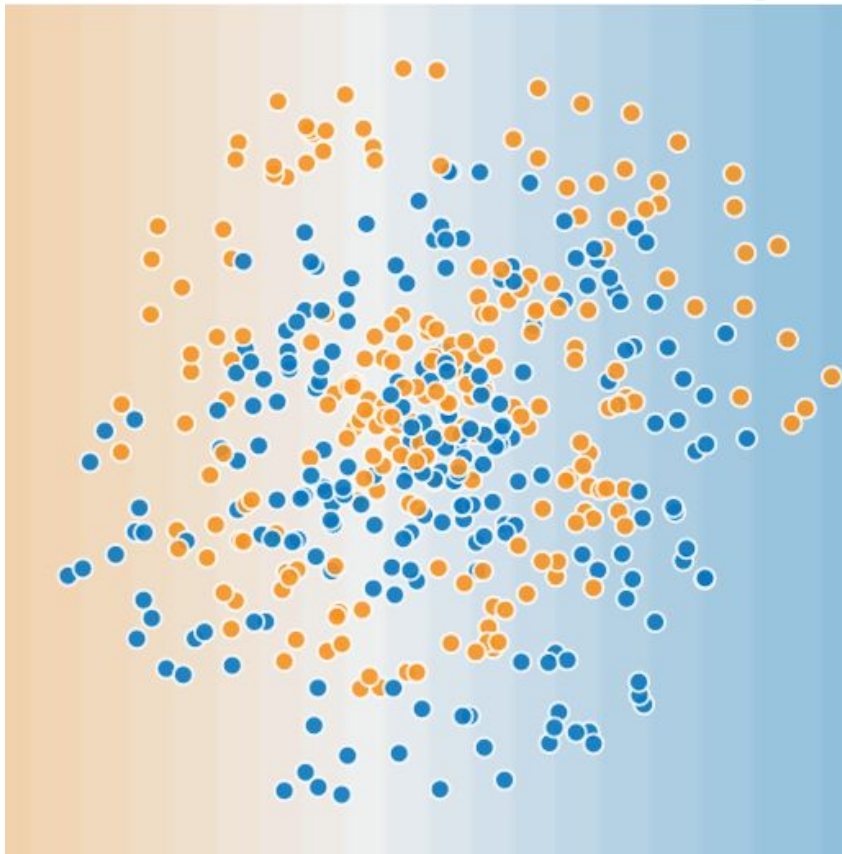*Linear Regression*

$$y = w_1 x_1 + w_2 x_2 + b$$

Output

w1   w2

x1   x2

b

Input

Logistic Regression

$$y = \sigma(\mathbf{w}\mathbf{x} + b)$$

17

How about this classification problem?

Linear model can not solve it

x → [function] → y
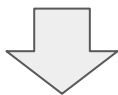
We need **non-linear functions**

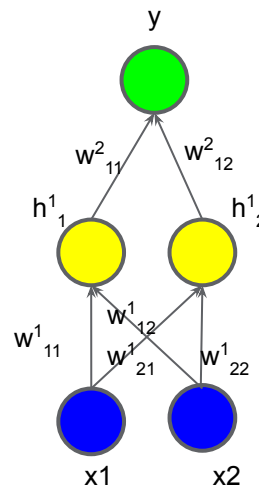# Add Complexity

For Simplicity, the bias term is ignored here.

$$h_1^1 = w_{11}^1 x1 + w_{12}^1 x2$$

$$h_2^1 = w_{21}^1 x1 + w_{22}^1 x2$$

$$y = w_{11}^2 h_1^1 + w_{12}^2 h_2^1 \quad = \quad y = \mathbf{wx}$$

$$y = (w_{11}^2 w_{11}^1 + w_{21}^2 w_{12}^1)x1 + (w_{12}^2 w_{12}^1 + w_{12}^2 w_{22}^1)x2$$
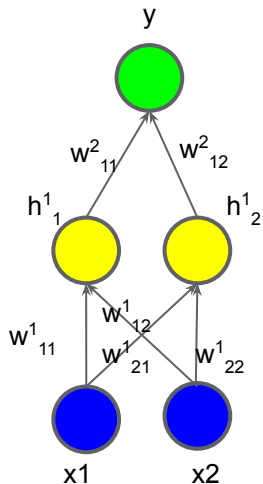

Output

Hidden Layer

Input

Add Complexity

Associative Law

$$h_1^1 = w_{11}^1 x1 + w_{12}^1 x2$$

$$h_2^1 = w_{21}^1 x1 + w_{22}^1 x2$$

$$y = w_{11}^2 h_1^1 + w_{12}^2 h_2^1$$

Matrix Format

Output

$$\begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

Hidden Layer

$$y = \begin{bmatrix} w_{11}^2 & w_{12}^2 \end{bmatrix} \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix}$$
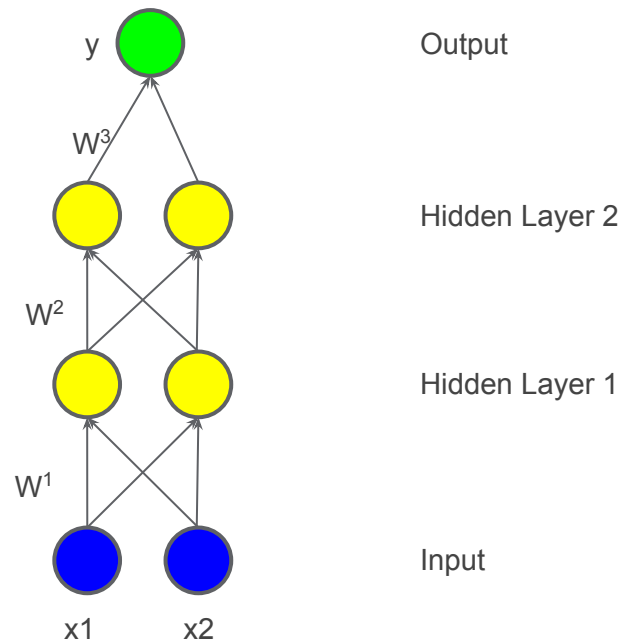
Input

$$y = W^2 W^1 \begin{bmatrix} x1 \\ x2 \end{bmatrix} = (W^2 W^1) \begin{bmatrix} x1 \\ x2 \end{bmatrix} = W \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

Add Complexity

$$y = \mathbf{W}^3 \mathbf{W}^2 \mathbf{W}^1 \begin{bmatrix} x1 \\ x2 \end{bmatrix} = (\mathbf{W}^3 \mathbf{W}^2 \mathbf{W}^1) \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$
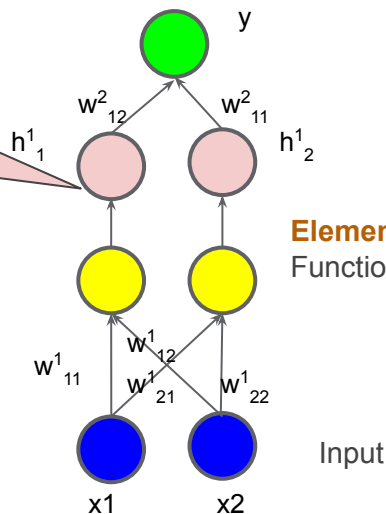
y

Output

W^3

Hidden Layer 2

W^2

Hidden Layer 1

W^1

Input

x1   x2

Make it non-linear

$$h_1^1 = \sigma(w_{11}^1 x1 + w_{12}^1 x2)$$
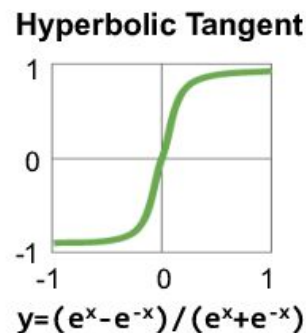
$$h_2^1 = \sigma(w_{21}^1 x1 + w_{22}^1 x2)$$

$$y = w_{11}^2 h_1^1 + w_{12}^2 h_2^1 \neq y = \mathbf{wx}$$
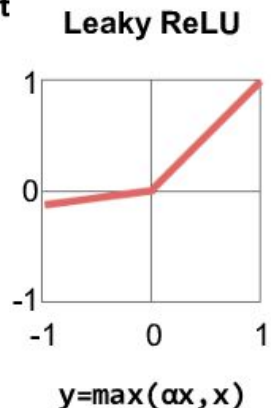
We Usually Don't Draw Non-Linear Transforms

y

$w^2{}_{12}$ $w^2{}_{11}$

$h^1{}_1$ $h^1{}_2$

Element-wise Non-linear Activation Function \sigma

$w^1{}_{11}$ $w^1{}_{12}$ $w^1{}_{21}$ $w^1{}_{22}$

Input

x1   x2

Non-linear Activation Functions

Traditional Non-Linear Activation Functions

Sigmoid

$y=1/(1+e^{-x})$

Hyperbolic Tangent

$y=(e^{x}-e^{-x})/(e^{x}+e^{-x})$
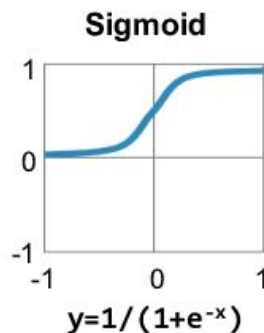
Modern Non-Linear Activation Functions

Rectified Linear Unit (ReLU)

$y=max(0,x)$

Leaky ReLU

$y=max(\alpha x, x)$

Exponential LU

$y=\begin{cases} x, & x \geq 0 \\ \alpha(e^{x}-1), & x < 0 \end{cases}$
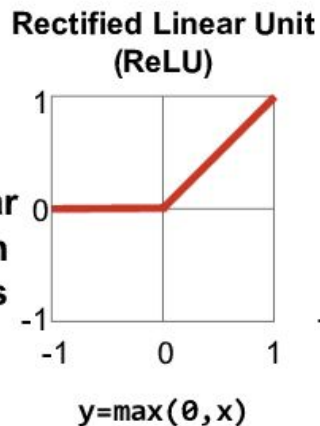
α = small const. (e.g. 0.1)
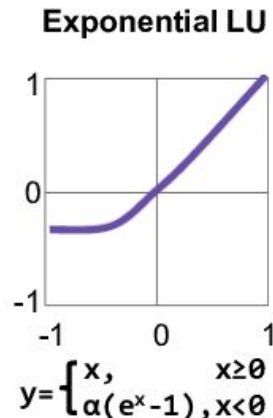
Add Non-linear Activation Function

$$y = \mathbf{W}^3 \sigma(\mathbf{W}^2 \sigma(\mathbf{W}^1 \mathbf{x}))$$

$y = \mathbf{W}^3 \mathbf{h}^2$ y   Output

W³

$\mathbf{h}^2 = \sigma(\mathbf{W}^2 \mathbf{h}^1)$ Hidden Layer 2

W²

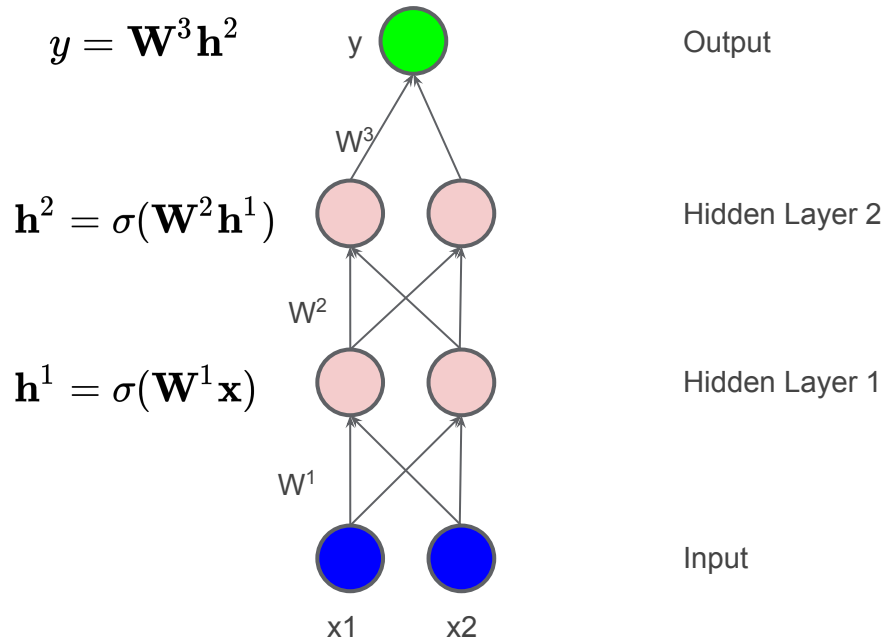$\mathbf{h}^1 = \sigma(\mathbf{W}^1 \mathbf{x})$ Hidden Layer 1
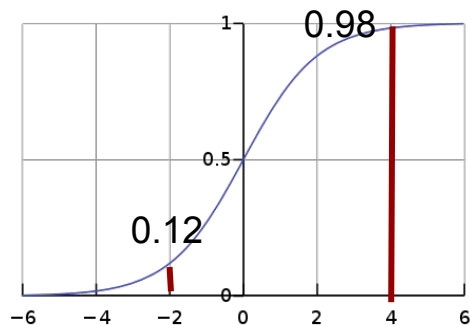
W¹

          Input

x1   x2

Why Non-linear Activation

- The non-linearities activation function increases the capacity of model

- Without non-linearities, deep neural networks is meaningless: each extra layer is just one linear transform.

- How to select activation functions?
  You can select an activation function which will approximate the distribution faster leading to faster training process.

Forward Computation
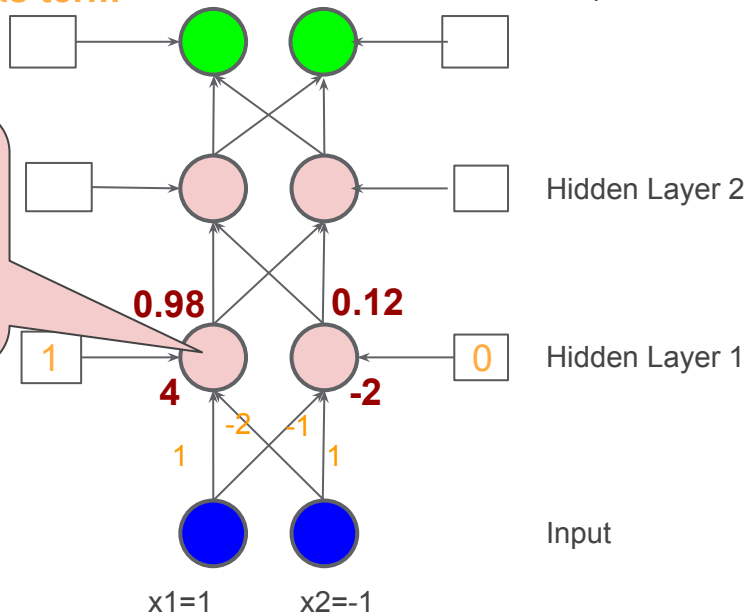

bias term
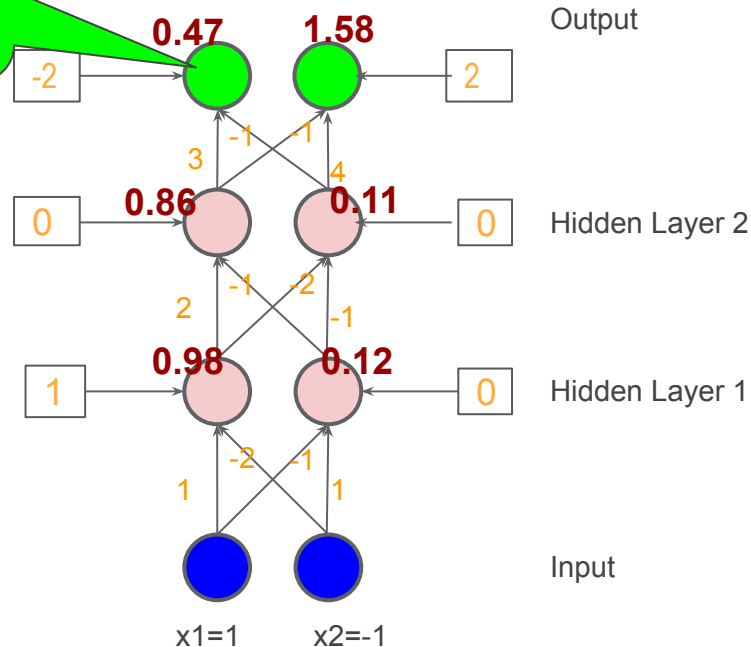
Output

Hidden Layer 2

Sigmoid function:

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

0.98   **0.12**

1   **4**   **-2**   0   Hidden Layer 1

-2   -1

1   1

0.98

0.12

x1=1   x2=-1   Input

Forward Computation

Forward Computation

1. Neural Network acts as a function that transforms the input vector into the output vector (target)

$$\begin{bmatrix} 0.47 \\ 1.58 \end{bmatrix} = f_\theta \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \right)$$

$$\begin{bmatrix} 0.04 \\ 1.7 \end{bmatrix} = f_\theta \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

$$\mathbf{W}^1 = \begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \quad \mathbf{b}^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix} 2 & -1 \\ -2 & -1 \end{bmatrix} \quad \mathbf{b}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
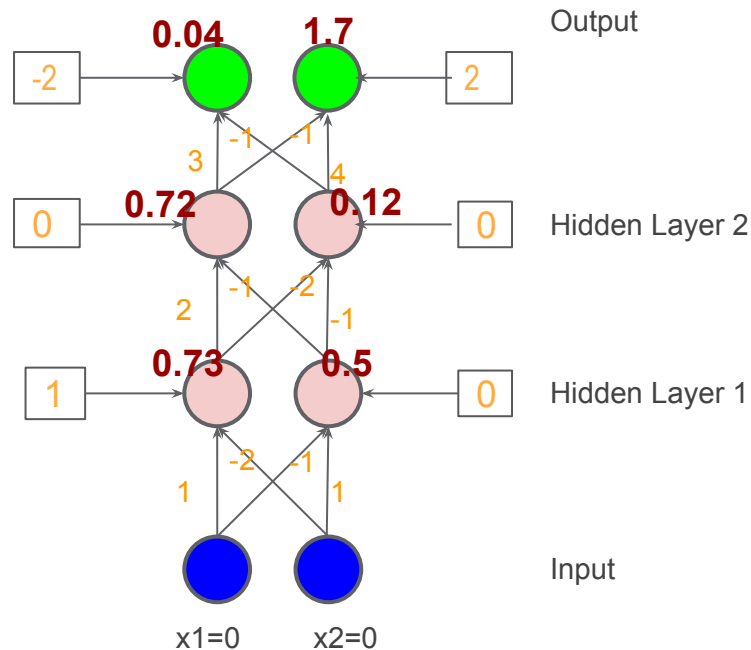
$$\mathbf{W}^3 = \begin{bmatrix} 3 & -1 \\ -1 & 4 \end{bmatrix} \quad \mathbf{b}^3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$
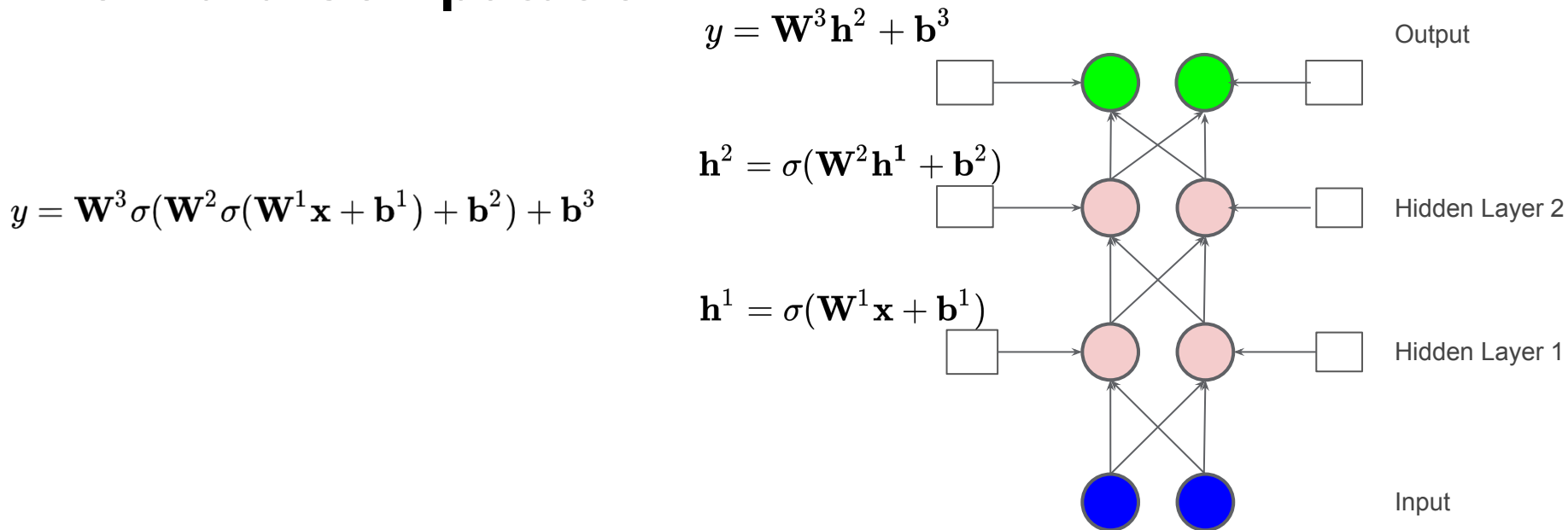
one param. set

2. It is actually a function space parameterized by weights matrices and bias vectors.

$$\theta = \{\mathbf{W}^1, \mathbf{b}^1, \mathbf{W}^2, \mathbf{b}^2, \mathbf{W}^3, \mathbf{b}^3\}$$
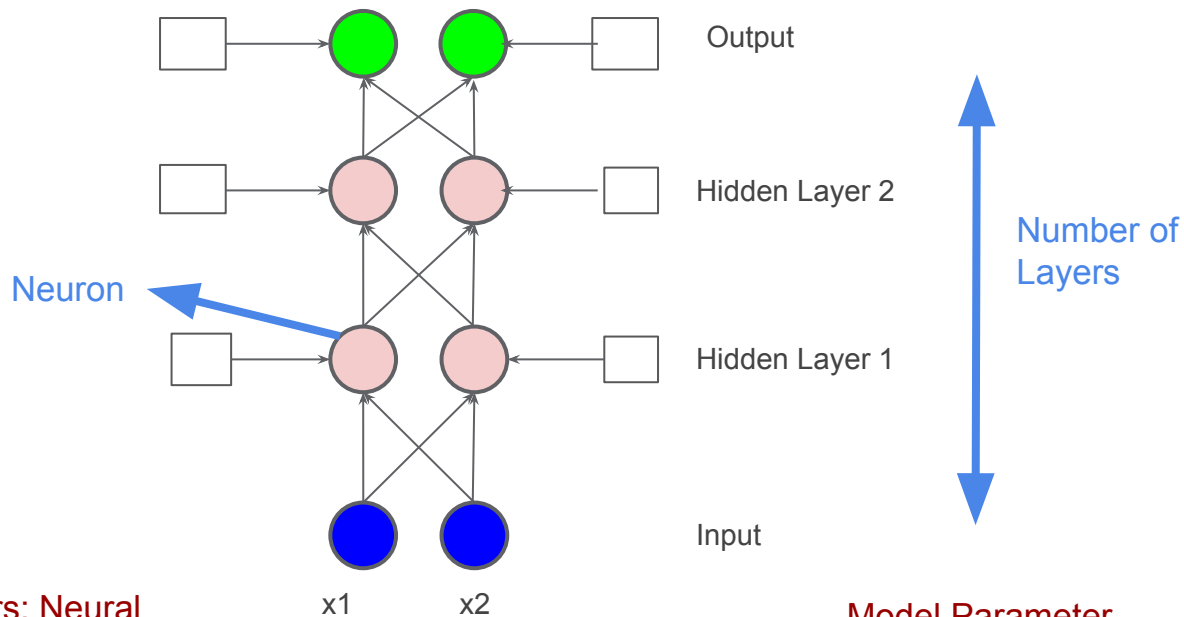
Parameter space

Output

Hidden Layer 2

Hidden Layer 1

Input

x1=0   x2=0

Forward Computation

$$y = \mathbf{W}^3 \mathbf{h}^2 + \mathbf{b}^3$$

Output

$$y = \mathbf{W}^3 \sigma(\mathbf{W}^2 \sigma(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3$$

$$\mathbf{h}^2 = \sigma(\mathbf{W}^2 \mathbf{h}^1 + \mathbf{b}^2)$$

Hidden Layer 2

$$\mathbf{h}^1 = \sigma(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1)$$

Hidden Layer 1

Input

1. Neural Network is a model that **recursively** applies the matrix multiplication and non-linear activation function.
2. Parallel computing techniques can be used to speed up matrix operation.

Neural Network: Function Set

Output

Hidden Layer 2

Number of Layers

Neuron

Hidden Layer 1

Input

x1   x2

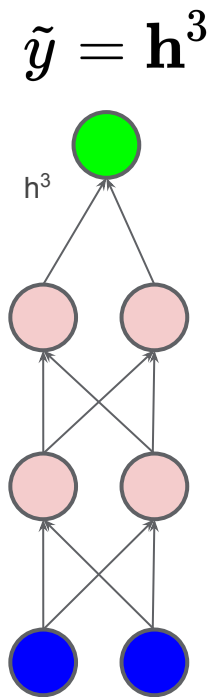Hyperparameters: Neural Network Structure

Model Parameter

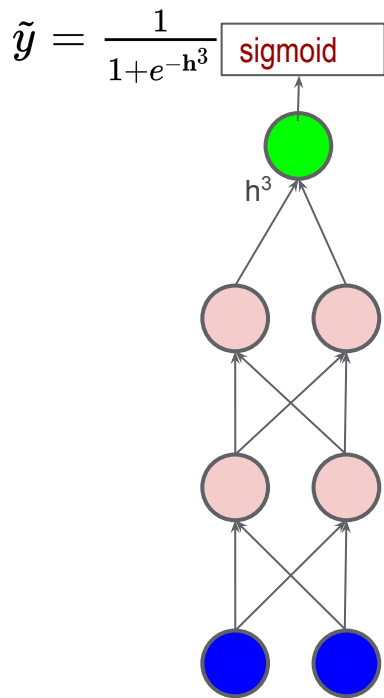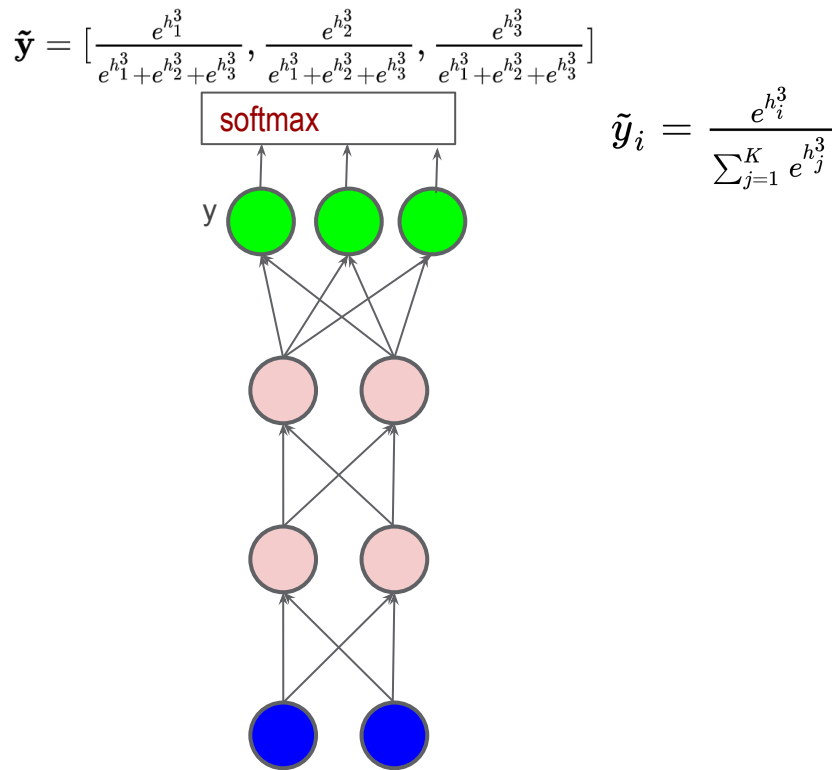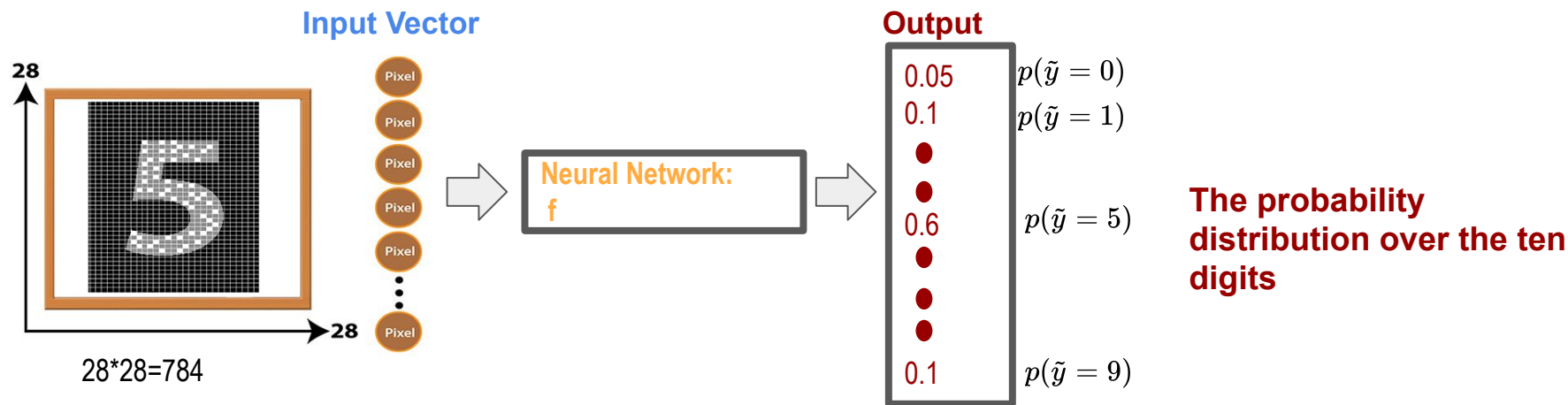| 1.Number of Layers
2.Number of neurons in each layer
3.Non-linear Activation function in each layer | **A function space containing various functions** | **A specific function mapping from input data to targets.** |

Output Layer

Regression

$$\tilde{y} = \mathbf{h}^3$$

h^3

Binary Classification

$$\tilde{y} = \frac{1}{1 + e^{-\mathbf{h}^3}}$$ sigmoid

h^3

Multi-label Classification

$$\tilde{\mathbf{y}} = [\frac{e^{h_1^3}}{e^{h_1^3} + e^{h_2^3} + e^{h_3^3}}, \frac{e^{h_2^3}}{e^{h_1^3} + e^{h_2^3} + e^{h_3^3}}, \frac{e^{h_3^3}}{e^{h_1^3} + e^{h_2^3} + e^{h_3^3}}]$$

softmax

$$\tilde{y}_i = \frac{e^{h_i^3}}{\sum_{j=1}^{K} e^{h_j^3}}$$

y

Example: MINST Dataset

Machine Learning Model

Output: 5

Input Vector

28

Pixel
Pixel
Pixel
Pixel
Pixel
Pixel

28*28=784

Neural Network: f

Output

0.05 $p(\tilde{y} = 0)$

0.1 $p(\tilde{y} = 1)$

0.6 $p(\tilde{y} = 5)$

0.1 $p(\tilde{y} = 9)$

The probability distribution over the ten digits

Example: MNIST Dataset

Input Vector

Output

28

28

28*28=784

Pixel
Pixel
Pixel
Pixel
Pixel
Pixel

Neural Network: f

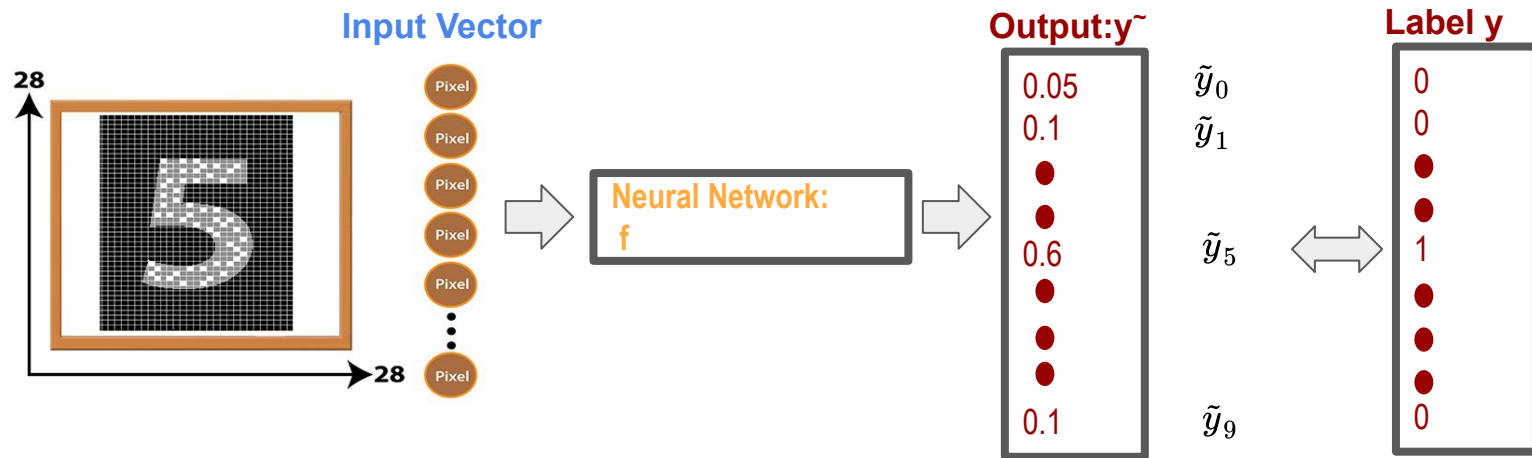|	
0.05	$p(\tilde{y} = 0)$
0.1	$p(\tilde{y} = 1)$
0.6	$p(\tilde{y} = 5)$
0.1	$p(\tilde{y} = 9)$
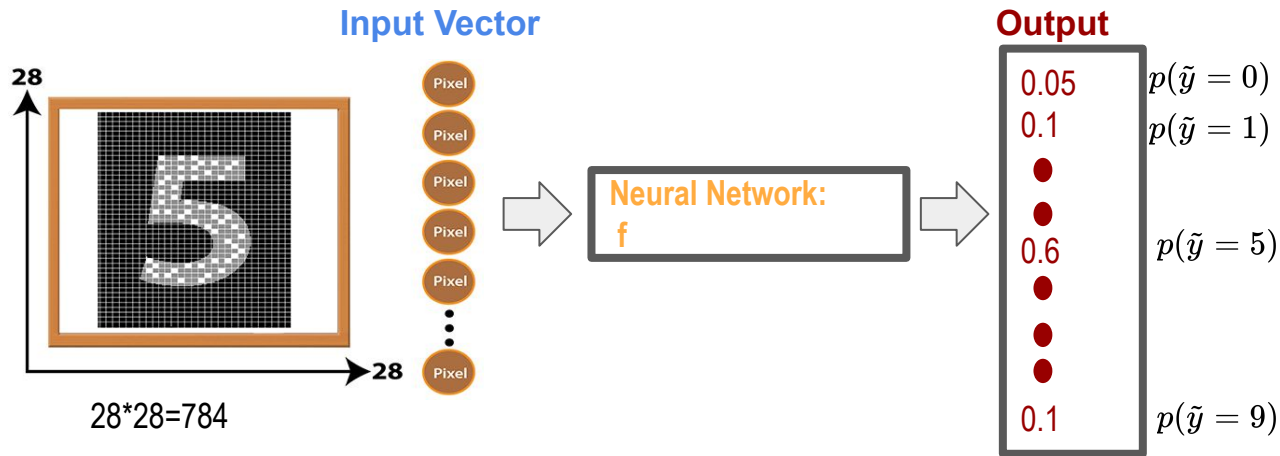
1. In this task, the neural network is a function mapping from the input 784-dim vector to the output 10-dim vector.
2. The neural network structure should be decided to make sure the best function exists in the function set.

3. Evaluation of Functions

Cross-Entropy Loss

Input Vector

28

Pixel
Pixel
Pixel
Pixel
Pixel
⋮
Pixel

28

Neural Network: f

Output:y~

0.05	$\tilde{y}_0$
0.1	$\tilde{y}_1$
●	
●	
0.6	$\tilde{y}_5$
●	
●	
●	
0.1	$\tilde{y}_9$

Label y

0
0
●
●
1
●
●
●
0

Given a set of parameters and one training sample,

$$loss(\tilde{\mathbf{y}}, \mathbf{y}) = -\sum_{i=0}^{9} y_i \, ln(\tilde{y}_i)$$

Total Loss

- Training dataset contains N training samples

- The total loss is: $J = \sum_{n=1}^{N} loss(\tilde{\mathbf{y}}_n, \mathbf{y}_n)$

- Find a function is the function set that minimizes the total loss J

- Find the network parameters θ that minimizes the total loss J. Modern

For loss function, training data are fixed and model parameters are unknown.
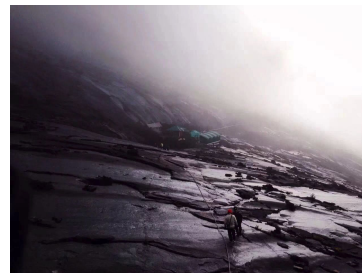
$$argmin_\theta J$$

4. Optimization

Gradient Descent

Gradient for the total loss function over parameters, which computed by Backpropagation algorithm

$$\theta_{t+1} = \theta_t - \alpha \bigtriangledown J(\theta_t)$$

New Parameters Guess
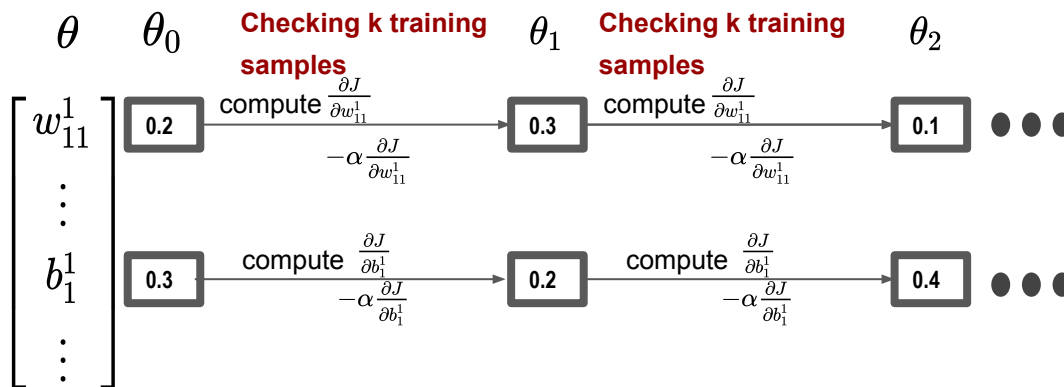
Current Parameters Guess

Learning Rate

Like hiking down a mountain
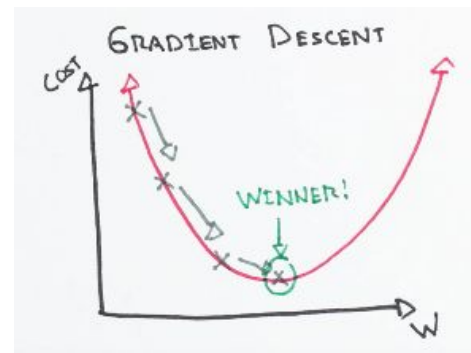
Credit: https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

Gradient Descent

θ θ_0 **Checking k training samples** θ_1 **Checking k training samples** θ_2

$$\begin{bmatrix} w^1_{11} \\ \vdots \\ b^1_1 \\ \vdots \end{bmatrix}$$

compute $\frac{\partial J}{\partial w^1_{11}}$

0.2 $\quad -\alpha \frac{\partial J}{\partial w^1_{11}}$ 0.3 compute $\frac{\partial J}{\partial w^1_{11}}$ $\quad -\alpha \frac{\partial J}{\partial w^1_{11}}$ 0.1 ● ● ●

0.3 compute $\frac{\partial J}{\partial b^1_1}$ $\quad -\alpha \frac{\partial J}{\partial b^1_1}$ 0.2 compute $\frac{\partial J}{\partial b^1_1}$ $\quad -\alpha \frac{\partial J}{\partial b^1_1}$ 0.4 ● ● ●

Backpropagation is used to compute gradients in an efficient way. $\dfrac{\partial J}{\partial \theta}$

Batch size: **k**

A dataset is [1,2,3,4,5,6] and the batch size is 2, one batch shuffle could be:
batch0=[2,1], batch1=[3,6], batch2=[4,5]

Backpropagation

Definition (from wiki):

By computing the gradient of the loss function with respect to each weight by the **chain rule**, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule

Predictions flows forward

Error flows backward

$h^l = \sigma(\mathbf{z}^l)$

$\mathbf{z}^l = \mathbf{W}^l h^{l-1} + \mathbf{b}^l$

$\mathbf{x} = \mathbf{h}^0$
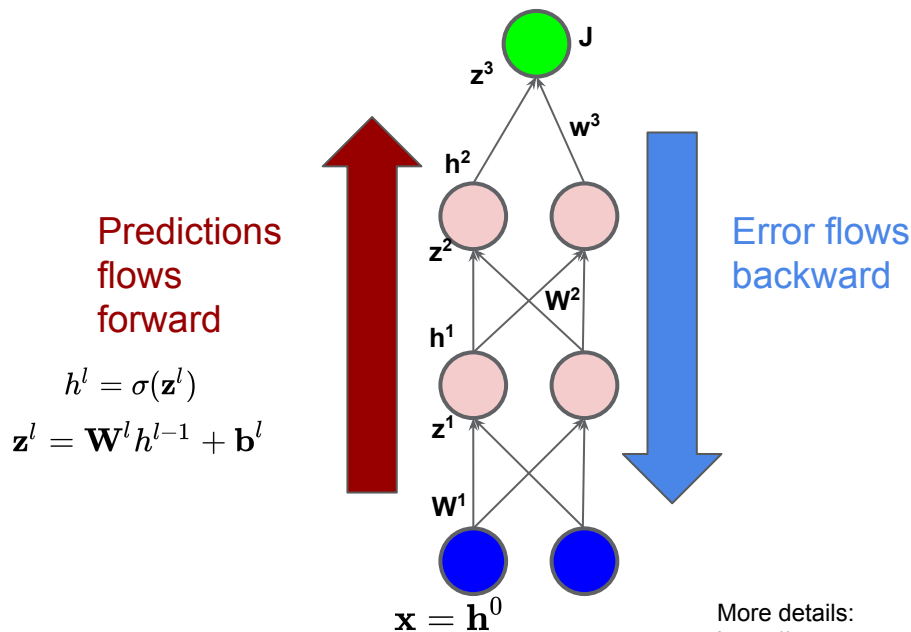
$$\frac{\partial J}{\partial \mathbf{w}^3} = \frac{\partial J}{\partial \mathbf{z}^3} \frac{\partial \mathbf{z}^3}{\partial \mathbf{w}^3}$$

$$\frac{\partial J}{\partial \mathbf{W}^2} = \frac{\partial J}{\partial \mathbf{z}^3} \frac{\partial \mathbf{z}^3}{\partial \mathbf{h}^2} \frac{\partial \mathbf{h}^2}{\partial \mathbf{z}^2} \frac{\partial \mathbf{z}^2}{\partial \mathbf{W}^2}$$

From **w**³

$$\frac{\partial J}{\partial \mathbf{W}^1} = \frac{\partial J}{\partial \mathbf{z}^3} \frac{\partial \mathbf{z}^3}{\partial \mathbf{h}^2} \frac{\partial \mathbf{h}^2}{\partial \mathbf{z}^2} \frac{\partial \mathbf{z}^2}{\partial \mathbf{h}^1} \frac{\partial \mathbf{h}^1}{\partial \mathbf{z}^1} \frac{\partial \mathbf{z}^1}{\partial \mathbf{W}^1}$$

From **w**³   From **W**²

More details:
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

40

Batch Size

Three approaches to select batch sizes:

1. Batch Gradient Descent   *batch size* = *Number of training data*

2. Mini-batch Gradient Descent   1<*batch size*< *number of training data*

3. Stochastic Gradient Descent   *batch size* = 1

Training Process

- Initialize neural network randomly

- For _ in range(number of epoch):

    - Shuffle all the training dataset into a list of batches

    - For _ in range(number of batches)

        - Get output with the input data in the batch

        - Compare outputs with ground truth in training data

        - Compute loss function with the batch data

        - Update weights with backpropagation and gradient descent algorithm

Iteratively perform

Learning Rate

Too low
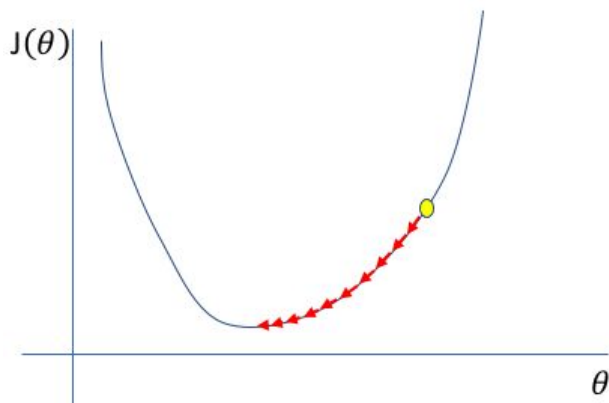
A small learning rate requires many updates before reaching the minimum point

Just right
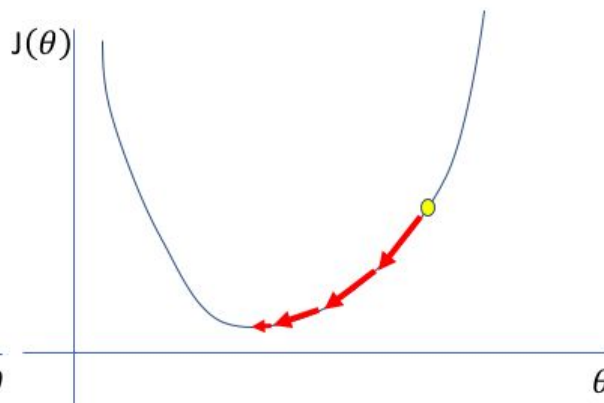
The optimal learning rate swiftly reaches the minimum point

Too high
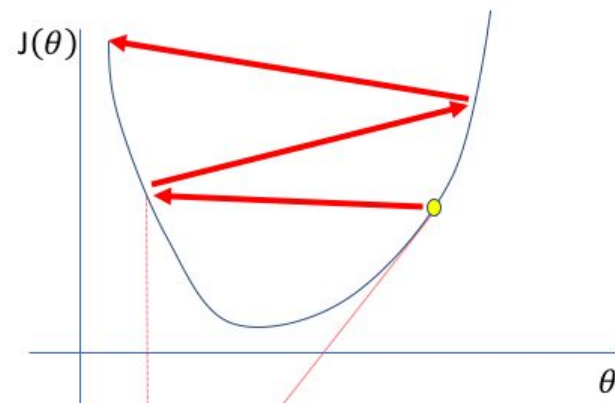
Too large of a learning rate causes drastic updates which lead to divergent behaviors

https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

43

Except SGD

SGD

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \bigtriangledown f(\mathbf{x}_n)$$

Different Variants

Momentum, Adam, AdaGrad, RMSProp

Auto-tune learning rates
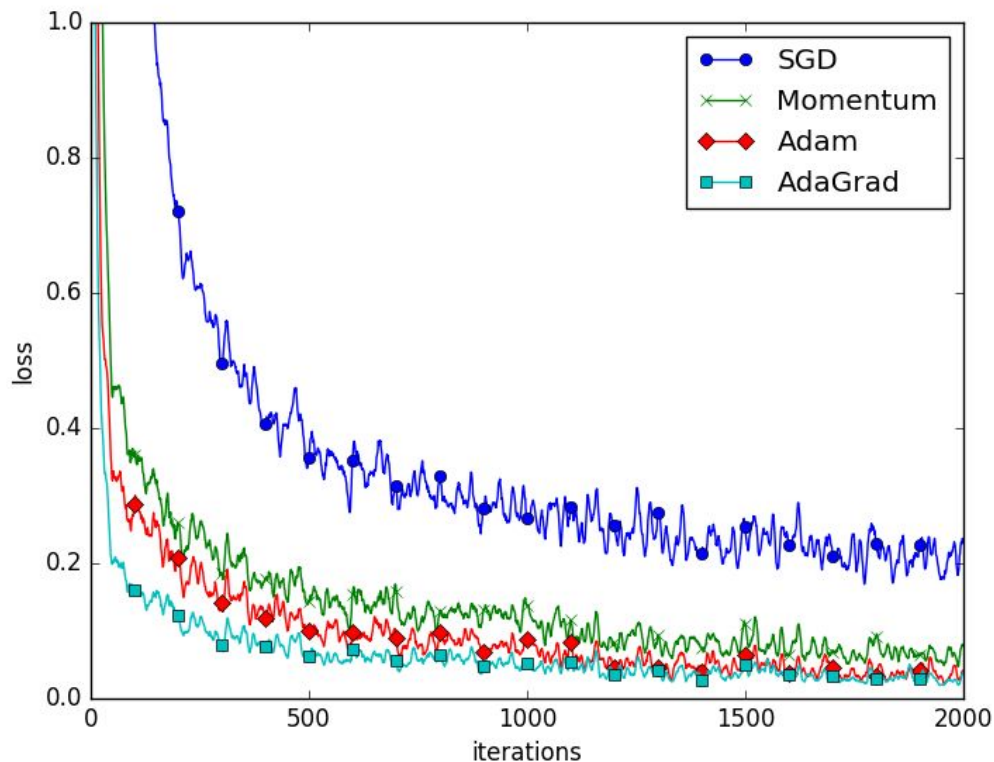
Neural Network Visualization

[Playground](#)

5. Deep Learning

Neural Network

- From Wiki:
  - NN is based on a collection of connected units of nodes called artificial neurons which loosely model the neurons in a biological brain.
- From another way:
  - NN is running several 'logistic regression' at the same time (expanding at width and depth dimensions).
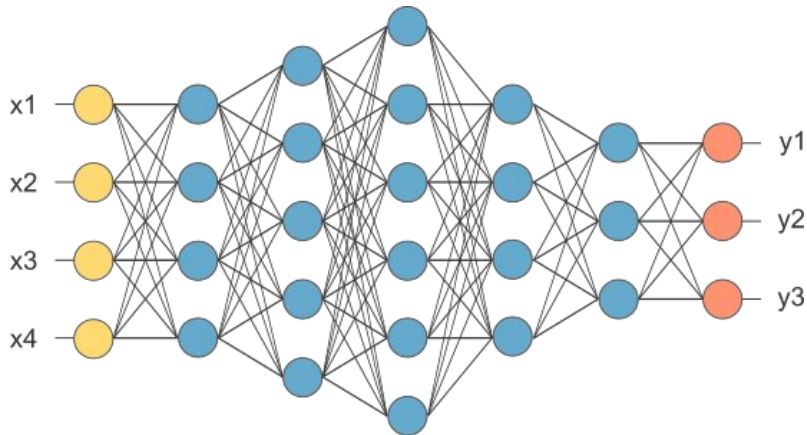
Neural Computation
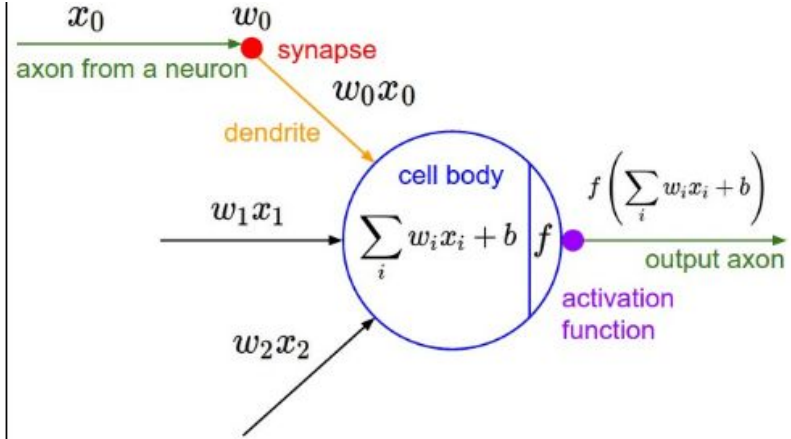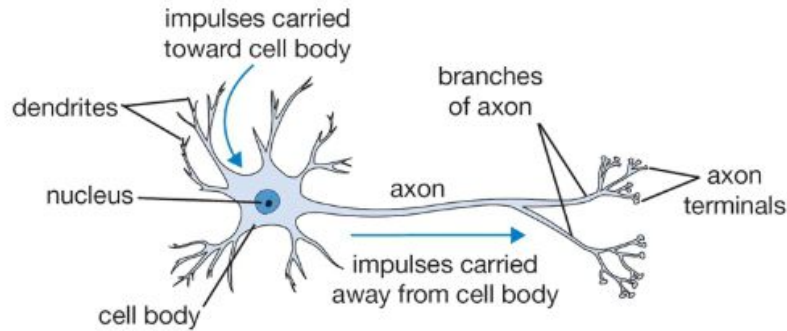
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

The fact that a neuron is essentially a logistic regression unit:
    1 performs a dot product with the input and its weights
    2 adds the bias and apply the non-linearity
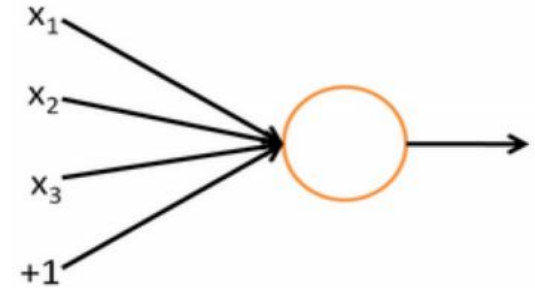
Shallow vs Deep

Simple Neural Network
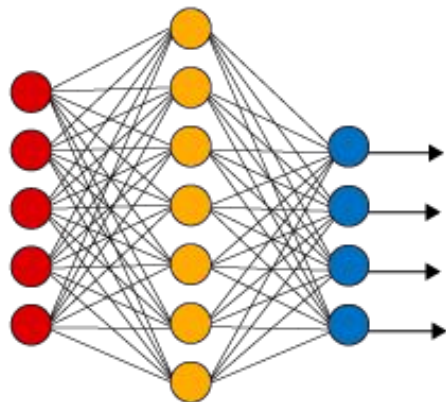
Deep Learning Neural Network
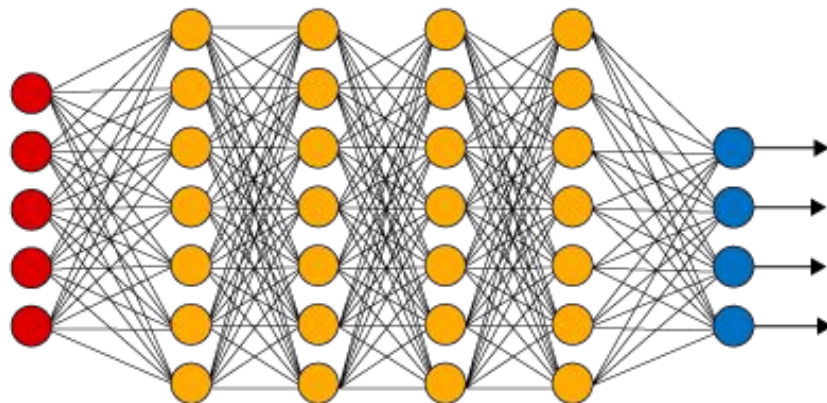
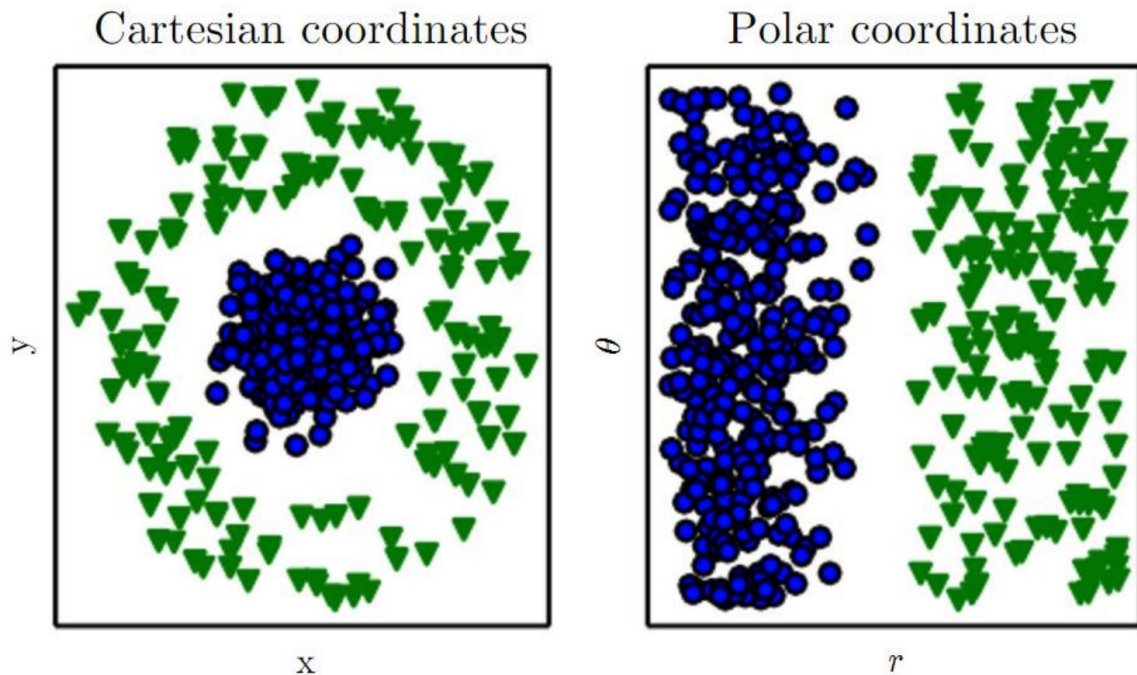● Input Layer   ● Hidden Layer   ● Output Layer

Representation Learning in DL
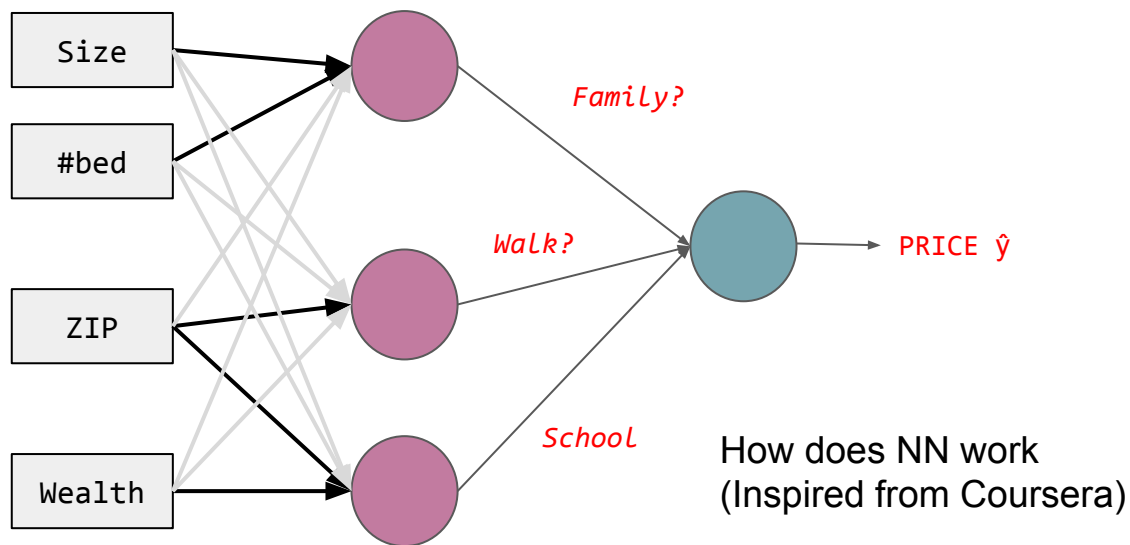
Representation Learning in DL
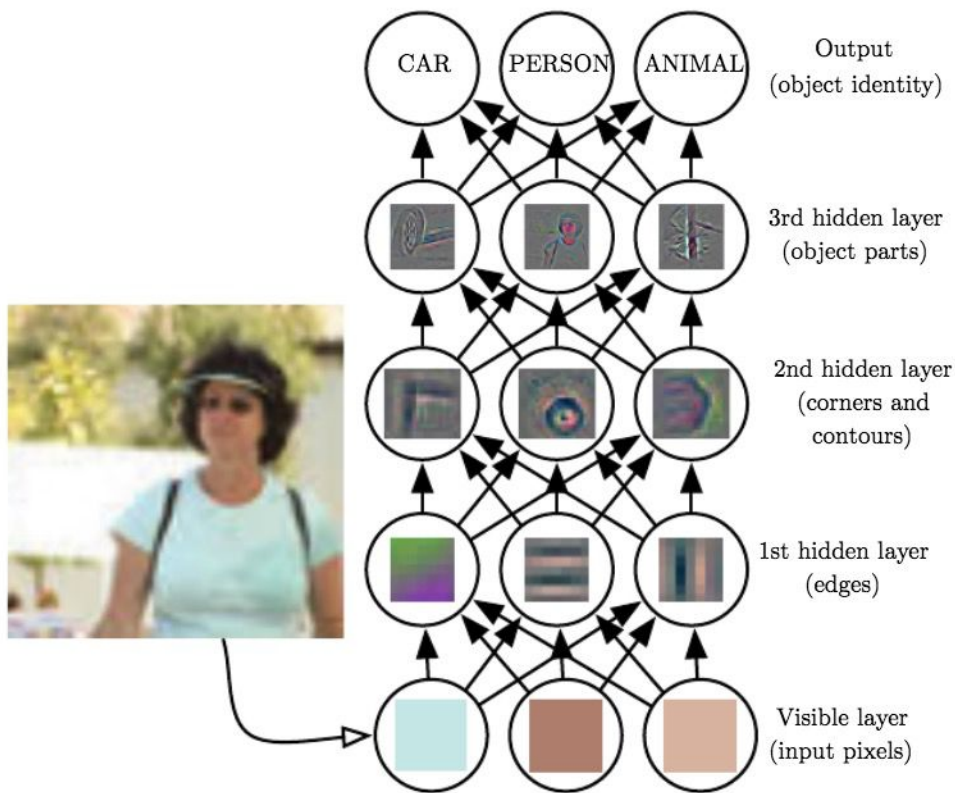
From Deep Learning (Goodfellow)
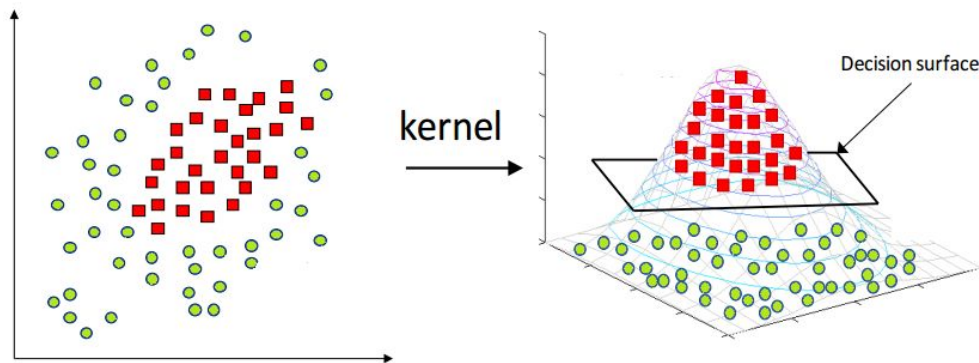
Representation Matters

Cartesian coordinates — Polar coordinates

Task: Draw a line to separate the **green triangles** and **blue circles**.

Cartesian coordinates   Polar coordinates
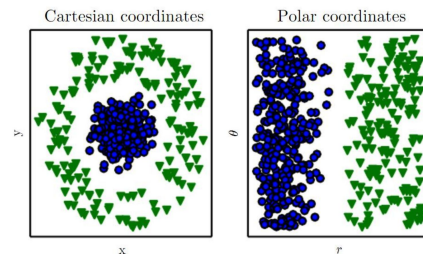
We want to project the data into the **new** feature/vector space that data is **linearly separated**
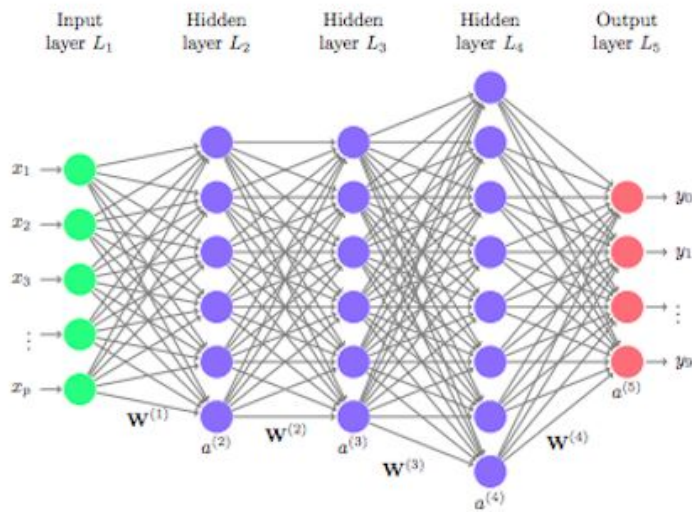
Kernel Tricks in SVM
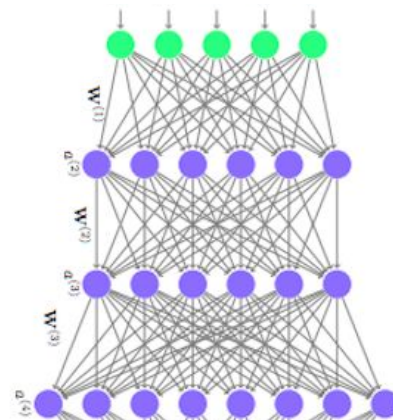

Low-dim, Original Space   High-dim, **Linearly Separated** Space
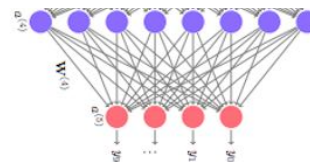
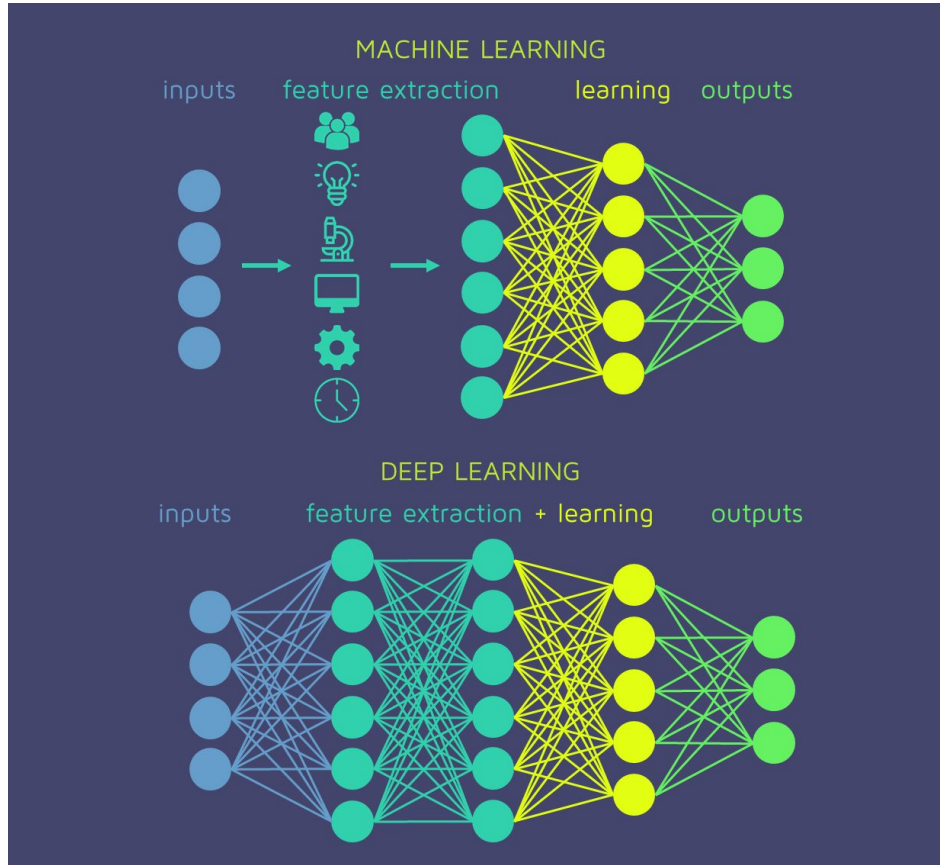# Hidden Representation in Deep Learning

Low-dim, Original Space

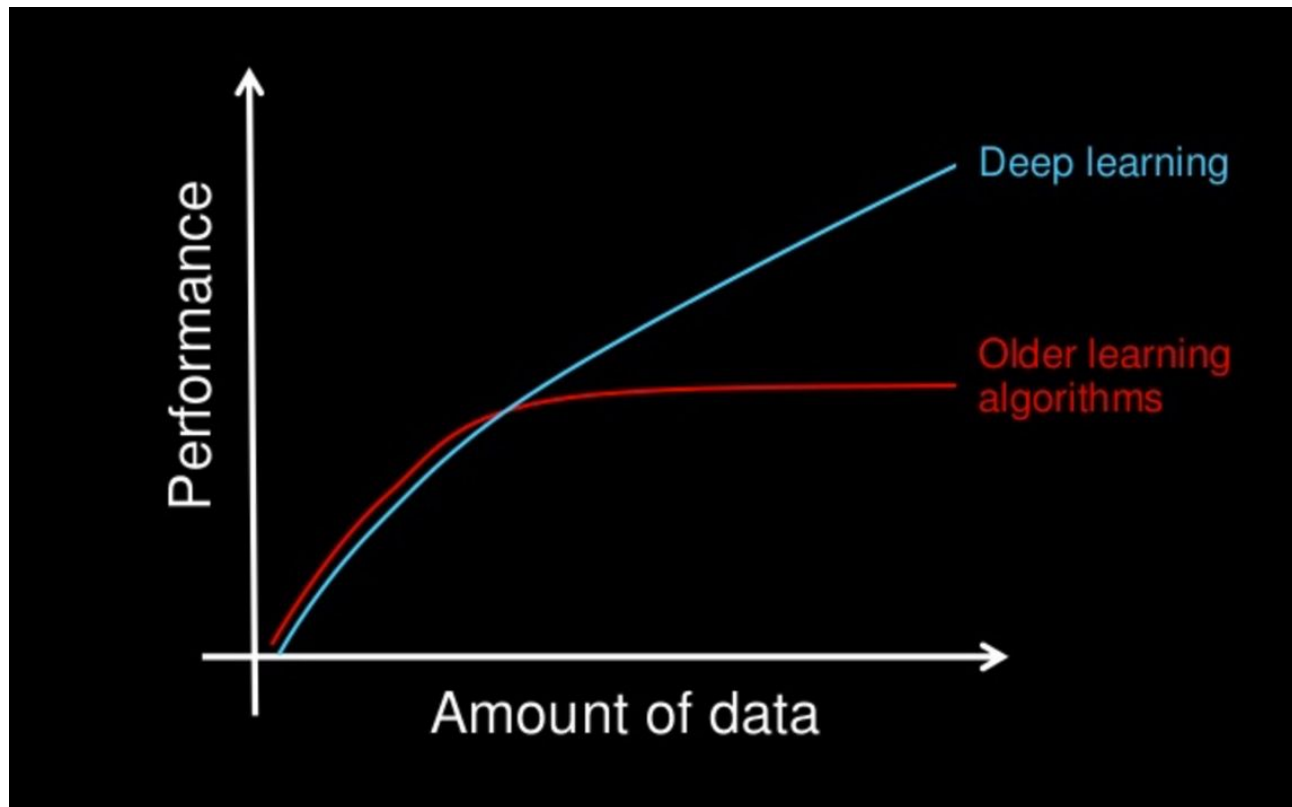High-dim, **Linearly Separated** Space

Softmax Classifier
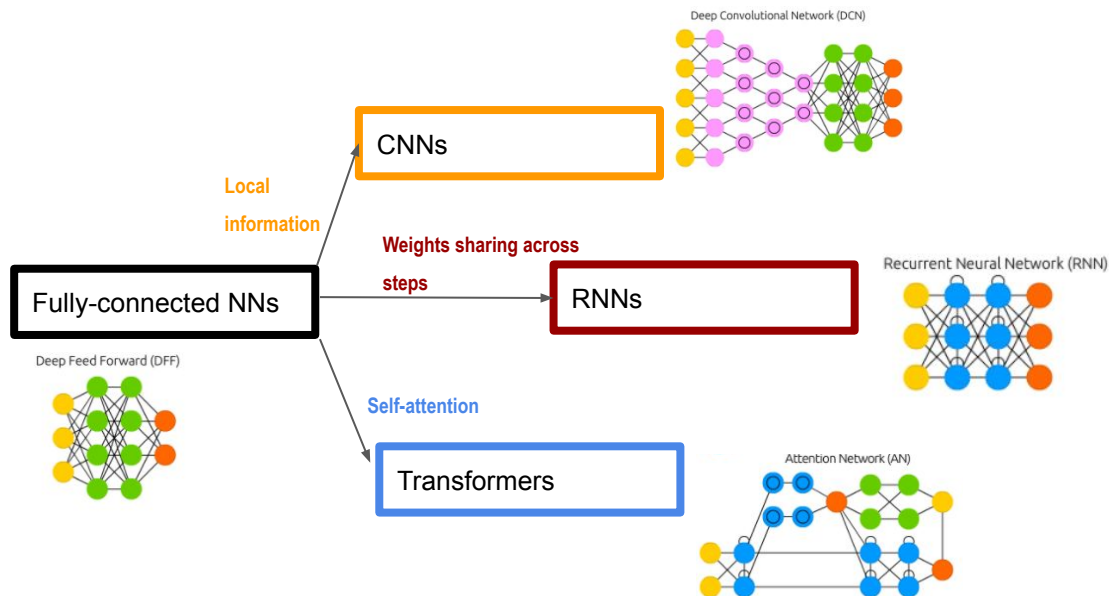(Linear Model)

End-to-End Learning

From Aporras

Why Deep Learning

From Andrew Ng

Deep Learning Structures

CNNs

Local information

Deep Convolutional Network (DCN)

Weights sharing across steps

Fully-connected NNs

RNNs

Recurrent Neural Network (RNN)

Deep Feed Forward (DFF)

Self-attention

Transformers

Attention Network (AN)

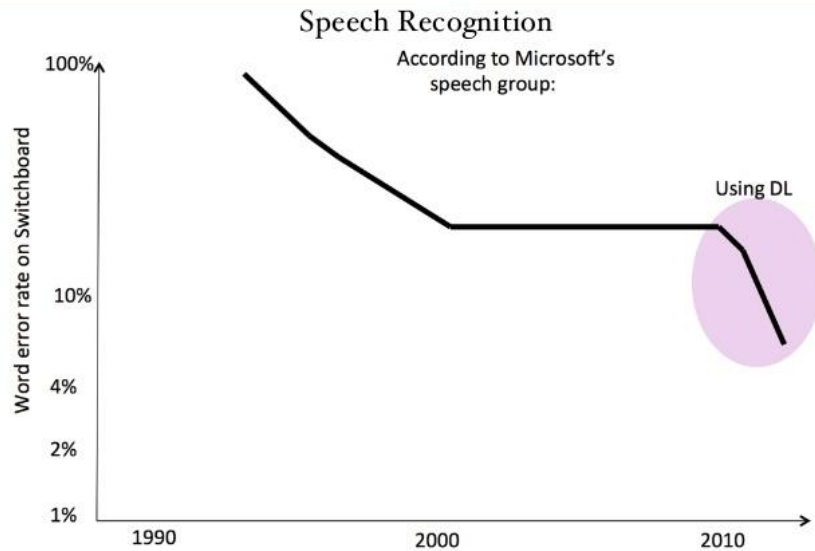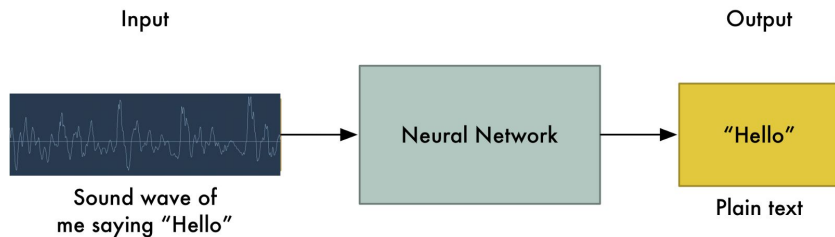https://www.asimovinstitute.org/author/fjodorvanveen/

Deep Learning

- Deep learning is a subfield of machine learning

- Most machine learning methods work well because of high-quality feature engineering/representation learning.

- Deep learning is an **end-to-end** structure, which supports automatic representation learning

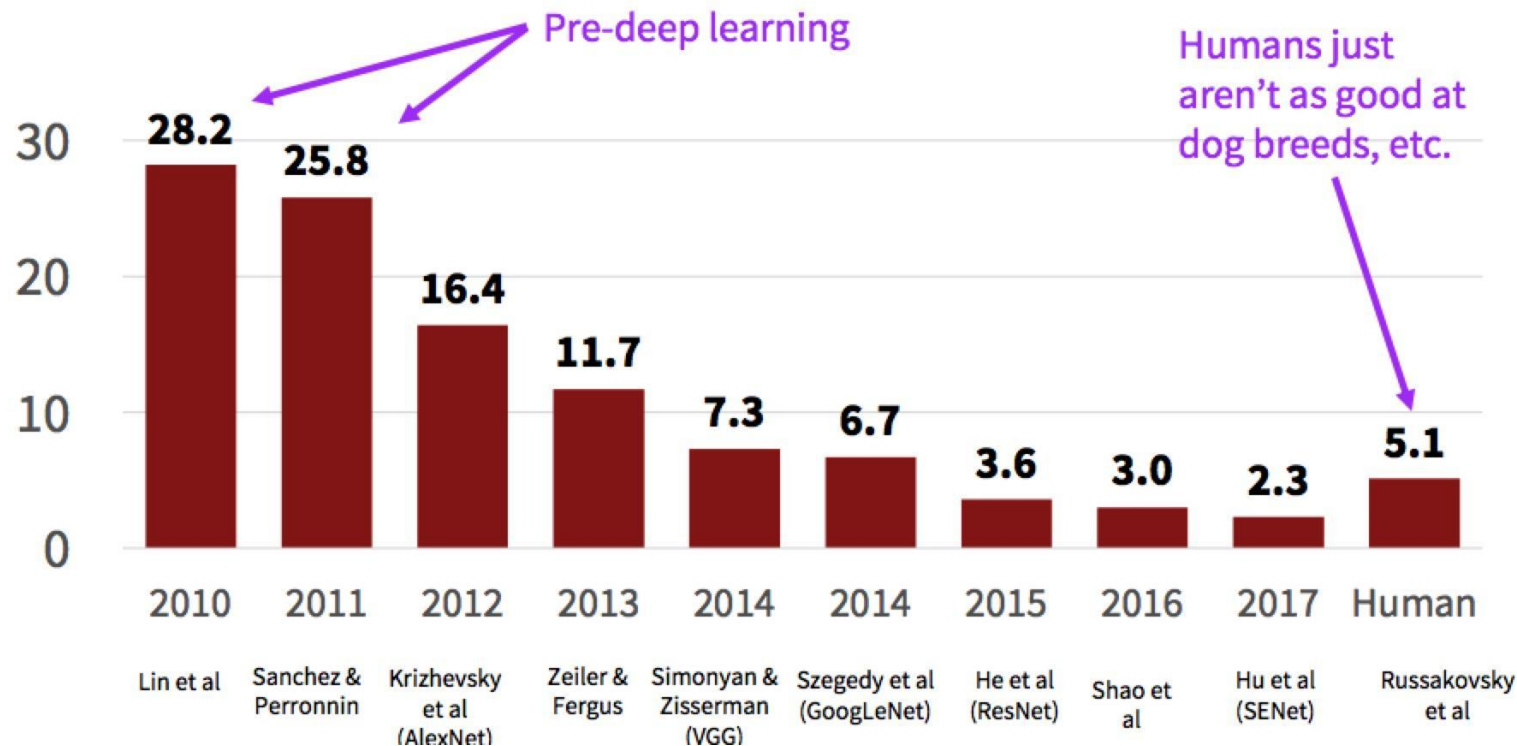- Different network structures: CNN, RNN, LSTM, GRU, Attention model, etc.

6. Applications of DL

Deep Learning for Speech

The first real-world tasks addressed by deep learning is speech recognition

Input

Output

Neural Network

"Hello"

Sound wave of
me saying "Hello"

Plain text

Speech Recognition
According to Microsoft's
speech group:

Using DL

Word error rate on Switchboard

100%

10%

4%

2%

1%

1990   2000   2010

Deep Learning for Computer Vision

Pre-deep learning

Humans just aren't as good at dog breeds, etc.

	2010	2011	2012	2013	2014	2014	2015	2016	2017	Human
Value	28.2	25.8	16.4	11.7	7.3	6.7	3.6	3.0	2.3	5.1
Author	Lin et al	Sanchez & Perronnin	Krizhevsky et al (AlexNet)	Zeiler & Fergus	Simonyan & Zisserman (VGG)	Szegedy et al (GoogLeNet)	He et al (ResNet)	Shao et al	Hu et al (SENet)	Russakovsky et al

Deep Learning for Arts

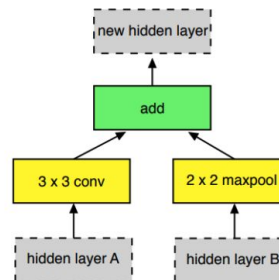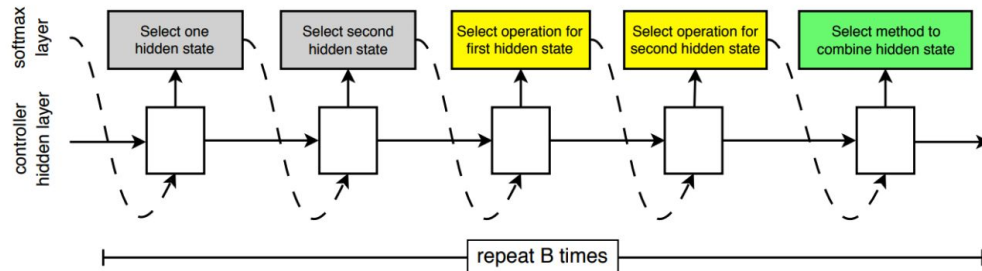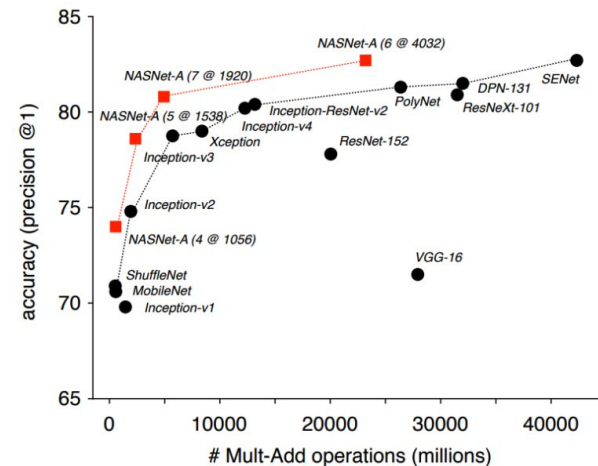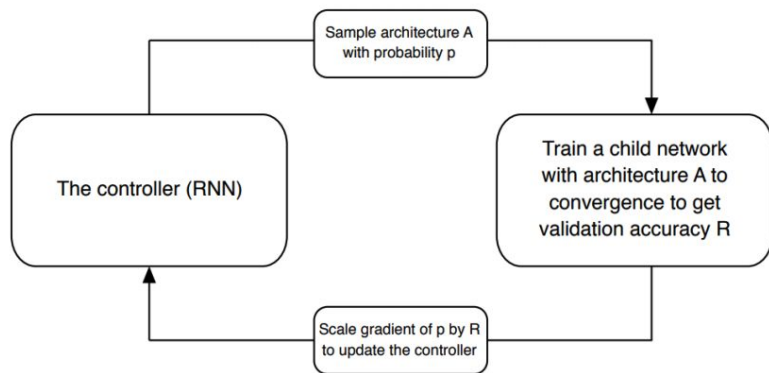Original photo   Reference photo   Result

Deep Learning for Data Generations

- Given training data, generate new samples from same distribution

AutoML and Neural Architecture Search

65

Happy Lunar New Year