DEEP LEARNING

Lecture 2: Basics of Machine Learning

Dr. Yang Lu

Department of Computer Science and Technology

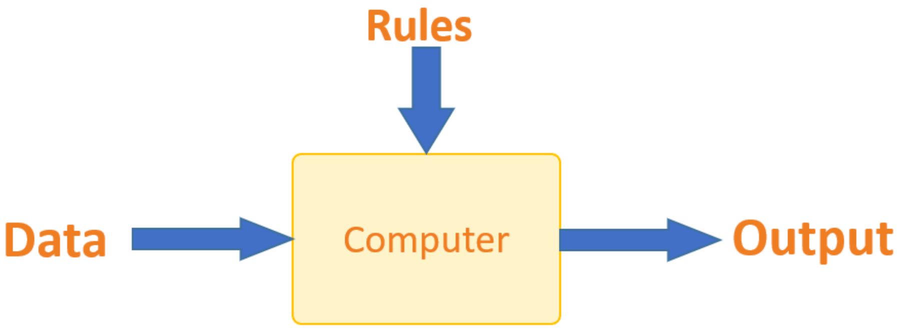
luyang@xmu.edu.cn

What is Machine Learning?

- Machine Learning is a system that can learn from example to produce accurate results through self-improvement and without being explicitly coded by programmer.
- The goal of machine learning: do prediction by learning from data.
  - The prediction output is then used to makes actionable insights.

Machine Learning vs. Traditional Programming

- In traditional programming, a programmer code all the rules in consultation with an expert in the area.
- Each rule is based on a logical foundation. The machine will execute an output following the logical statement.



Rules

Data

Computer

Output

Traditional Programming

- Disadvantages:
  - When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.
  - The expert's knowledge may not be comprehensive enough to provide accurate rules.

Image source: https://www.guru99.com/machine-learning-tutorial.html

Example of Traditional Programming

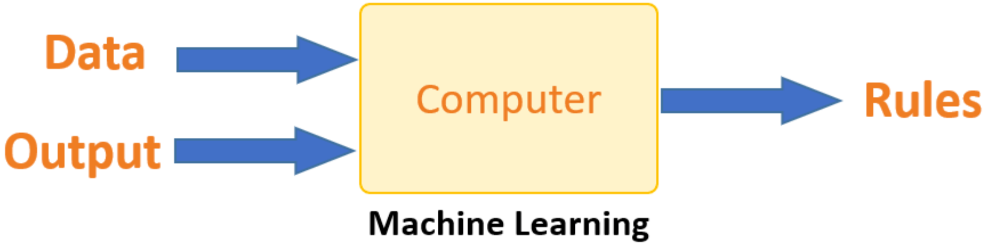
- Take recommandation as an example.
- We want to predict if a customer is willing to buy a lipstick. We build the following rules:

```
Check if the customer is female.
    Yes, check if the customer’s age is above 18.
        Yes, return true.
        No, return false.
    No, return false.
```

- Based on the rules, we have the inference:
  - (Female, 20) -> True;
  - (Male, 25) -> False.

Machine Learning vs. Traditional Programming

- The machine learns how the input and output data are correlated and it writes a rule.
- The programmers write an algorithm to generate rules rather than write rules.
- The algorithms may learn implicit rules that experts are not able to know.



Data

Output

Computer

Rules

Machine Learning

Image source: https://www.guru99.com/machine-learning-tutorial.html

Example of Machine Learning

Known data:

(Female, 20, True)

(Male, 20, False)

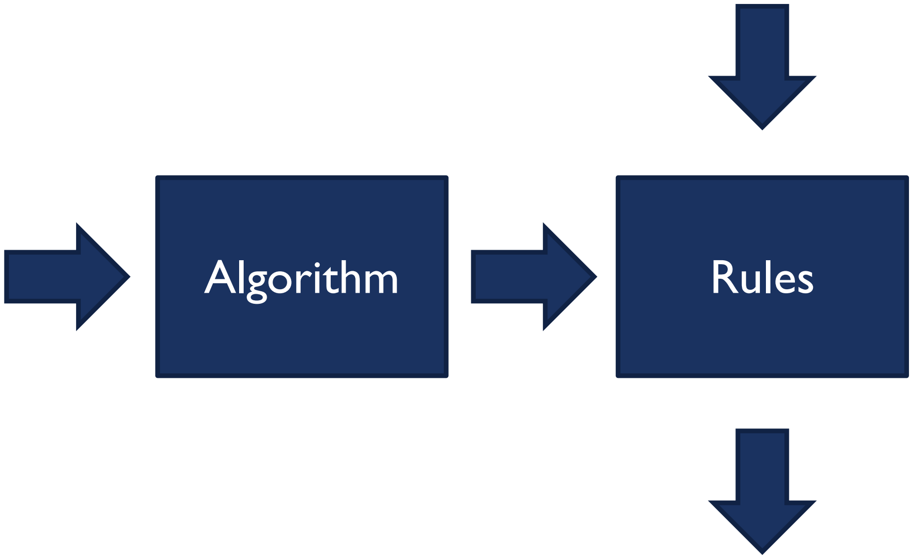
(Female, 15, True)

(Female, 50, False)

(Male, 25, True)

(Male, 18, False)

New data: (Female, 27)



Algorithm

Rules

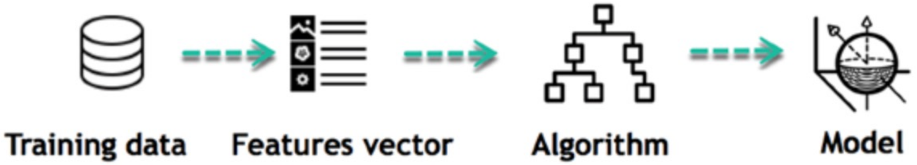
True

Machine Learning Terminology

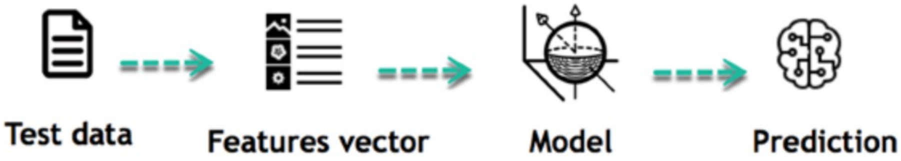
- The list of attributes used to solve a problem is called a feature vector.
    - E.g. gender, age, occupation, income...
- The target we want to predict is called label.
    - E.g. Willing to buy or not (True / False).
- The rules used to predict is called model.
- The program that is used to generate the model is called algorithm.

Machine Learning Terminology

- The process to generate a model a called training or learning, and the set of data used in this process is called training data.


Training data
Features vector
Algorithm
Model

- The process to use a trained model to predict is called testing or inference, and the data used in this process is called test data.


Test data
Features vector
Model
Prediction

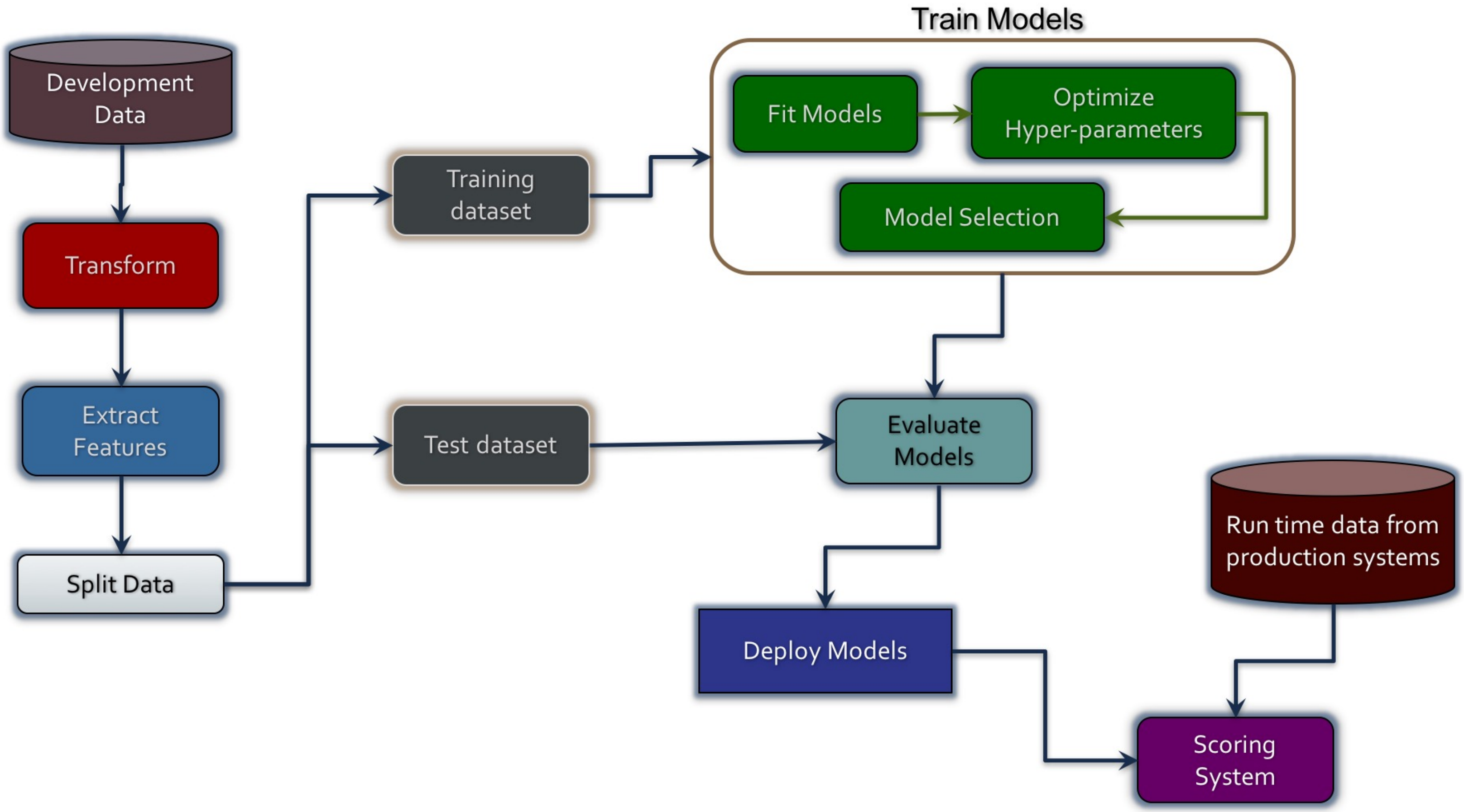
- A single data instance is called a sample or an example.

Image source: https://www.guru99.com/machine-learning-tutorial.html

How does Machine learning work?

- The way the machine learns is similar to the human being.
- Human brain:
  - Humans learn from experience.
  - The more we know, the more easily we can predict.
  - By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation.
- Machine algorithm:
  - To make an accurate prediction, the machine sees an example.
  - When we give the machine a similar example, it can figure out the outcome.
  - However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

A Typical Machine Learning Process


Train Models
Development Data
Transform
Extract Features
Split Data
Training dataset
Test dataset
Fit Models
Optimize Hyper-parameters
Model Selection
Evaluate Models
Run time data from production systems
Deploy Models
Scoring System

Types of Machine Learning Problems

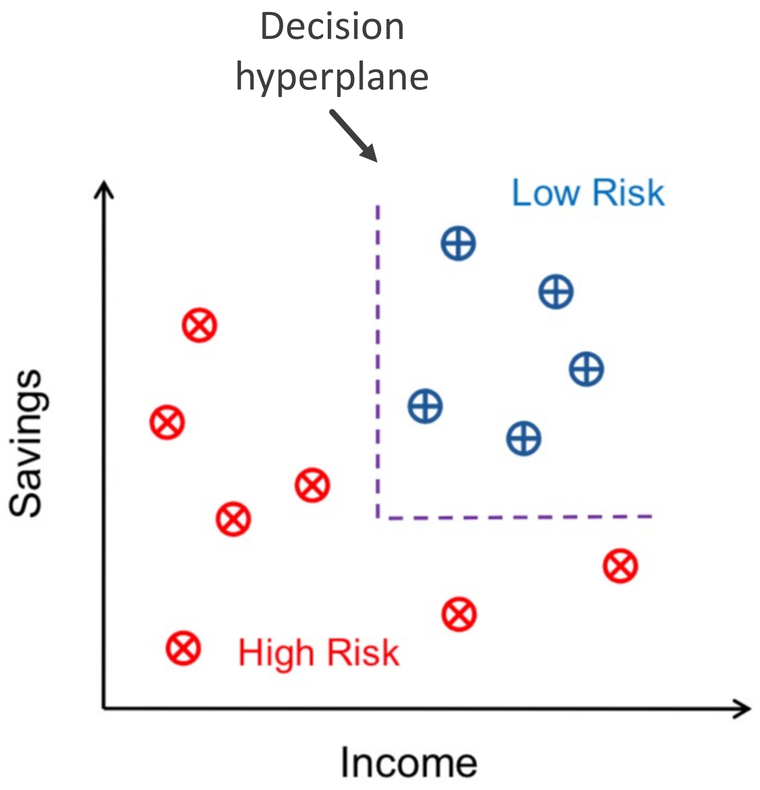
- Supervised learning: applications where input vectors $\boldsymbol{x}$, and their label (target) vectors y, are known during training.
- Unsupervised learning: discover the structure or representation of input vectors $\boldsymbol{x}$, absent of knowledge of their label information during training.
- Reinforcement learning: finding suitable actions in certain scenarios to maximize the given reward. It discovers policies through trial and error.

Supervised Learning

- We have the label information such that we are able to know if a sample is correctly predicted or not during training.
  - E.g. the market knows if a customer buys the lipstick or not and use it in training.
- The label in the training data is also called ground truth.
- There are two categories of supervised learning:
  - Classification
  - Regression

Classification

- The label of classification problem is discreate.
- The discreate value that the label can take is called class.
  - If the label only has two classes, it is called binary classification.
  - If the label only has more than two classes, it is called multiclass classification.



Binary classification with 2-d feature space

Classification Examples

- Binary classification:
  - Face verification (True, False)
  - Sentiment analysis (positive, negative)
  - Spam filtering (True, False)
  - Cancer diagonosis (benign, malignant)
- Multiclass classification:
  - Face identification (Alice, Bob, Charles, …)
  - Object recognition (flower, ball, cup, dog, …)
  - Weather prediction (sunny, rainy, foggy, …)
  - Behavior recognition (walking, running, dancing, …)

Regression

- The label of classification problem is continuous.
- Example:
  - Stock price prediction.
  - Temperature prediction.
  - Salary estimation.
  - House price prediction.

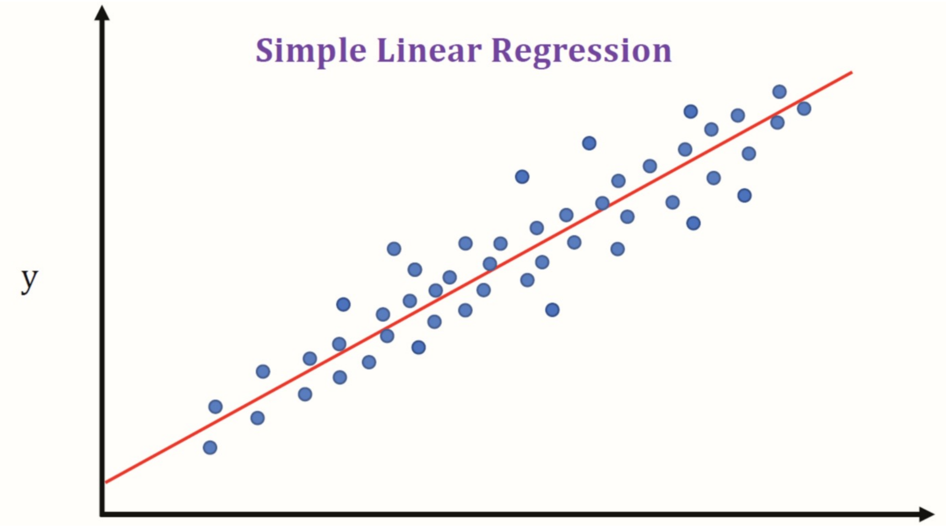
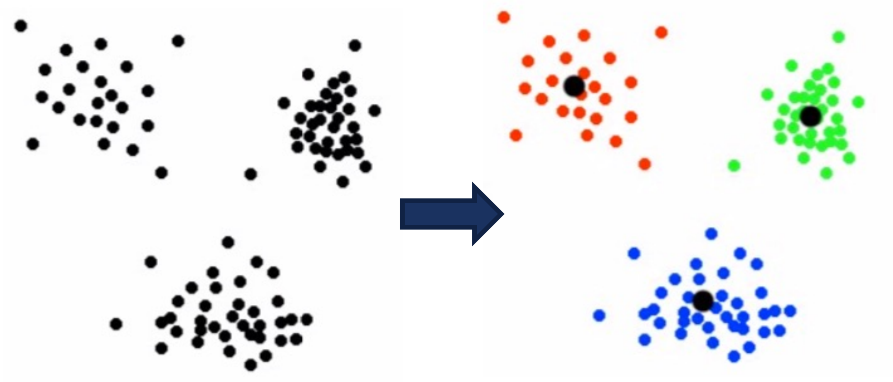


Image source: https://miro.medium.com/max/1400/1*Cw5ZSYDkIFpmhBwr-hN84A.png

Unsupervised Learning

- In unsupervised learning, a model is trained to learn previously undetected patterns in a data set with no pre-existing labels.
  - You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you.
- Example:
  - Social network user grouping.
  - Image segmentation.
  - Anomaly detection.



Clustering

Traditional Machine Learning Methods

- Supervised learning
  - Logistic regression, SVM, neural networks, decision tree, LDA, naïve Bayes, ensemble methods, …
- Unsupervised learning
  - k-means, mixture models, DBSCAN, autoencoder, PCA, …

Reinforcement Learning

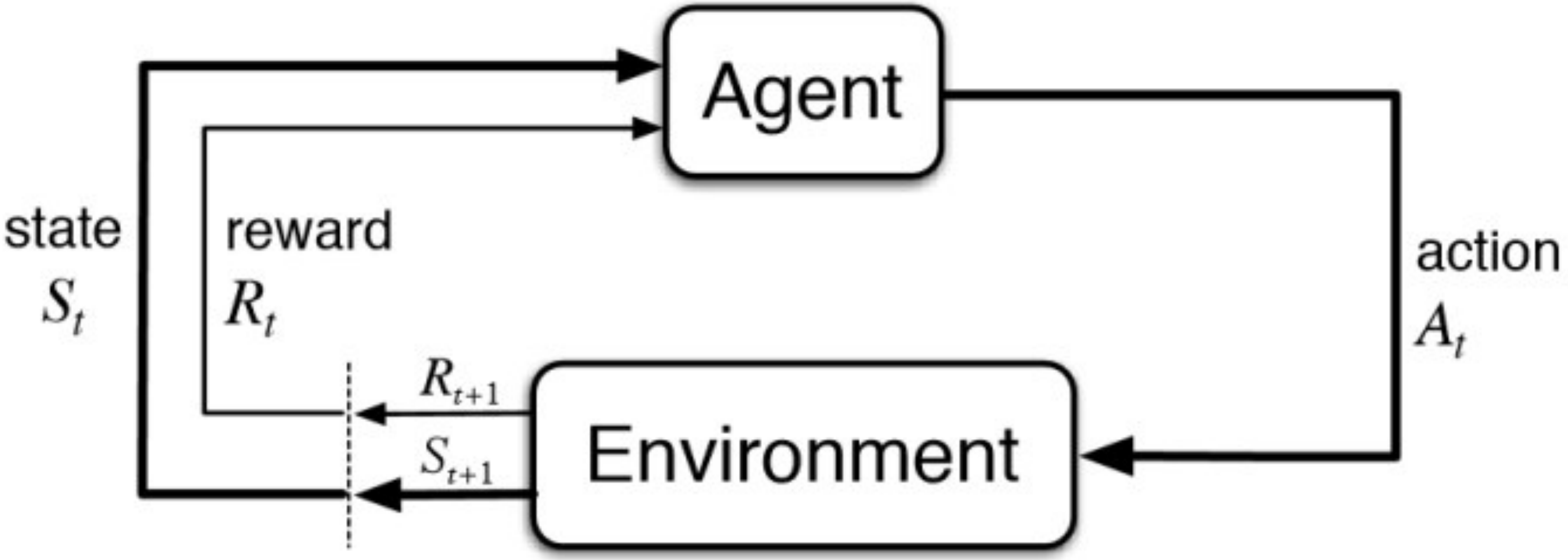

Agent
state
S_t
reward
R_t
R_{t+1}
S_{t+1}
Environment
action
A_t

Image source: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

Reinforcement Learning

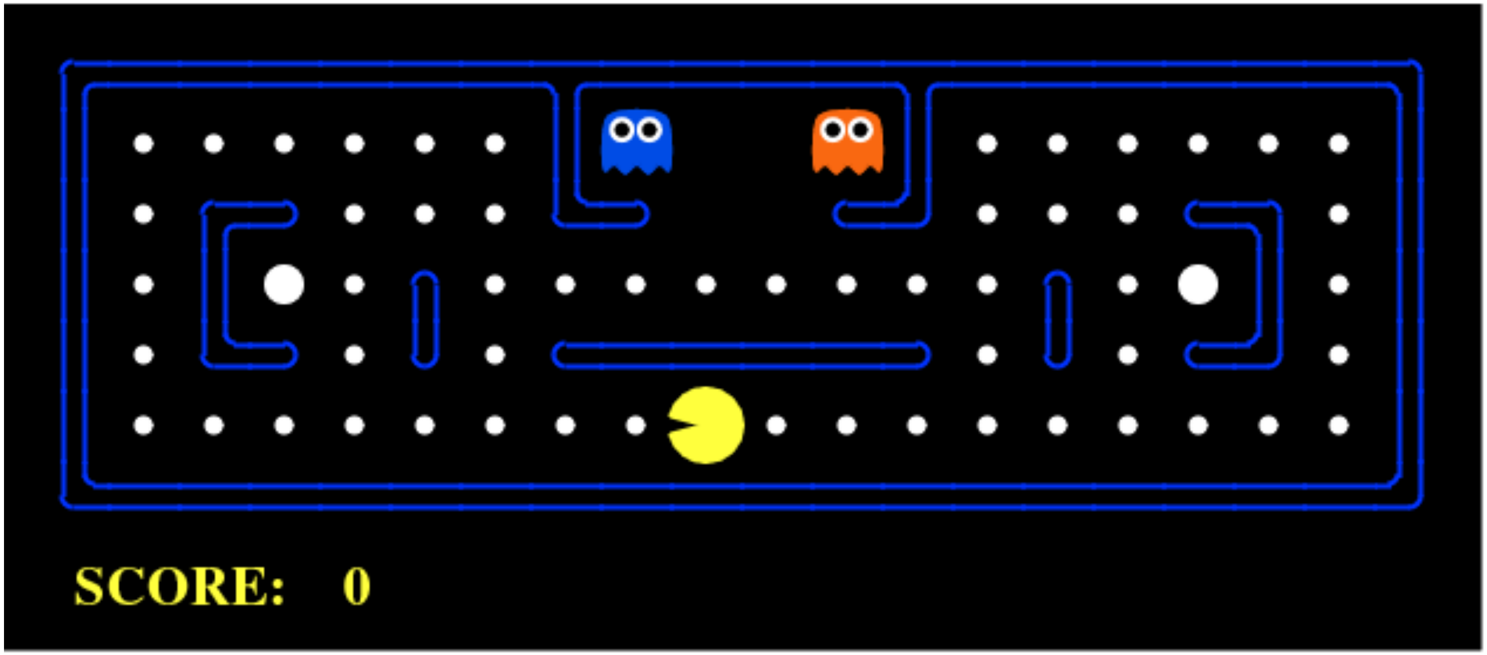


Image source: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

LINEAR MODELS

Data Representation

- For a given dataset, we usually use x to represent the features and y to represent the label. For the ith sample:

$$\boldsymbol{x}_i = \left[x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, \dots, x_d^{(i)}\right]^T \in \mathbb{R}^d$$

$$y_i \in \mathbb{R}$$

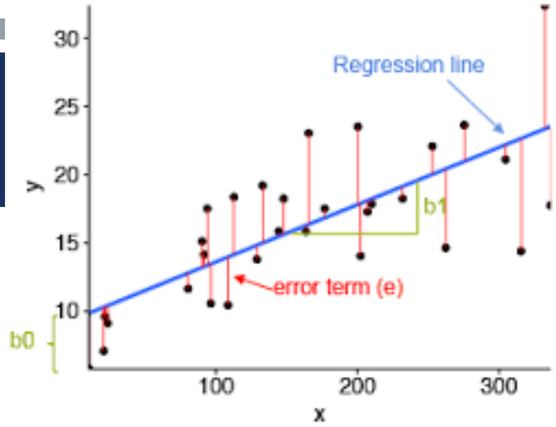
- A dataset can be represented as:

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{\mathrm{d}\times n}$$

$$\boldsymbol{y} = [y_1, y_2, y_3, \dots, y_n] \in \mathbb{R}^n$$

- $\mathbb{R}$ is the domain of real number, d is the feature dimension and n is the number of samples.
- We use bold font (e.g. $\boldsymbol{x}$) to represent vector, and uppercase letter (e.g. $\boldsymbol{X}$) to represent matrix.

Linear Regression



- Linear regression model can be represented by

$$
\begin{aligned}
f(\boldsymbol{x}) &= \boldsymbol{w}^T\boldsymbol{x} + b \\
&= w_1x_1 + w_2x_2 + \cdots + w_dx_d + b
\end{aligned}
$$

- $\boldsymbol{w}$ is called the model weights or coefficients, and b is called the bias or intercept. Together they are called the model parameters.
- The goal of linear regression is to find $\boldsymbol{w}$ and b such that the following loss function (aka cost function) is minimized:

$$
J = \frac{1}{n}\sum_{i=1}^{n}\left(f(\boldsymbol{x}_i) - y_i\right)^2
$$

- This cost function is also known as the Mean Squared Error (MSE) function.

Image source: https://miro.medium.com/max/1400/1*Cw5ZSYDkIFpmhBwr-hN84A.png

Gradient Descent

- The gradient vector is orthogonal to the tangent of a plane towards the greater value.
- Thus, the direction of negative gradient heads to the local minimum.
- We can update our model parameter by iteratively adding the negative gradient.

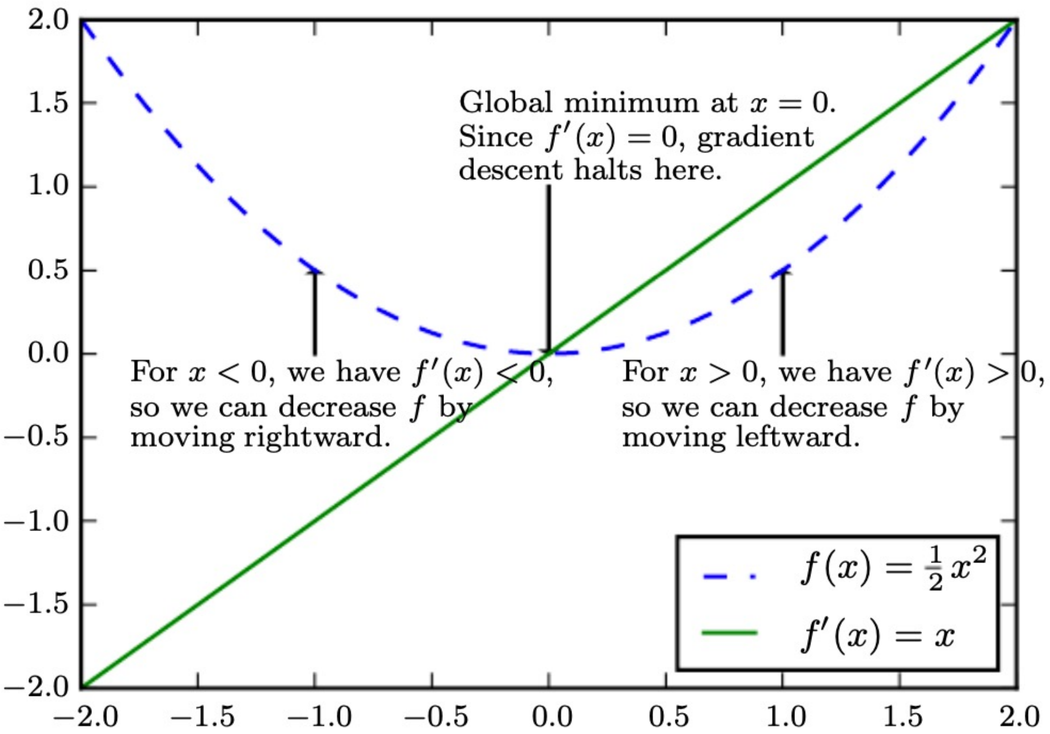


Image source: Figure 4.1, Goodfellow, Bengio, and Courville, Deep Learning, Cambridge: MIT press, 2016.

Gradient Descent

- To solve this minimization problem, we calculate its partial derivatives:

$$\frac{\partial J}{\partial w_j} = \frac{2}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)x_j^{(i)}$$

$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)$$

- Putting partial derivatives together in a vector is the gradient $\nabla J(\boldsymbol{w})$.
- Thus, the model weights can be iteratively updated by:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla J(\boldsymbol{w})$$

$$b \leftarrow b - \eta \frac{\partial J}{\partial b}$$

The cost function is convex such that gradient descent is able to find the global minimum

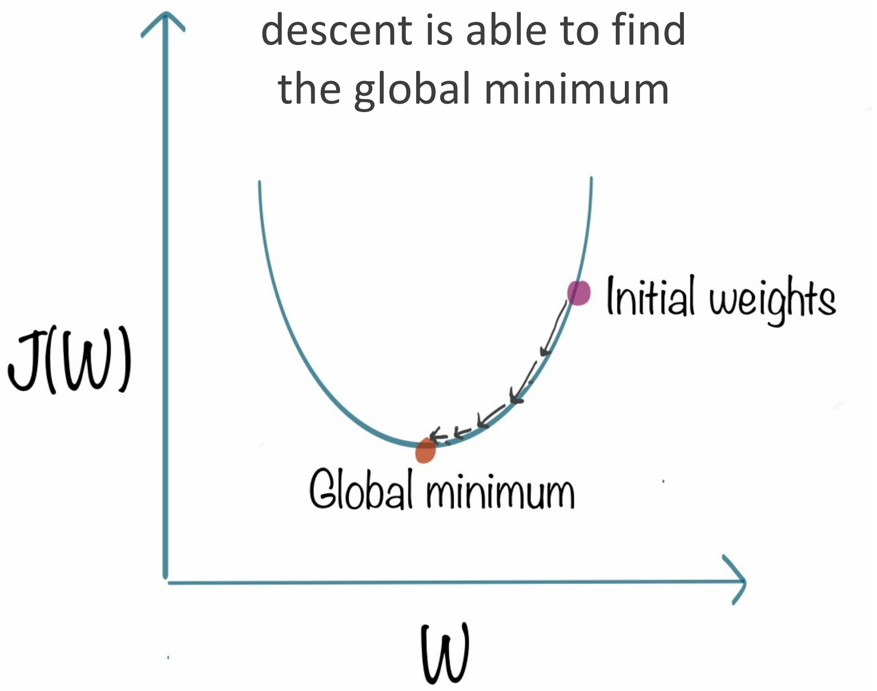
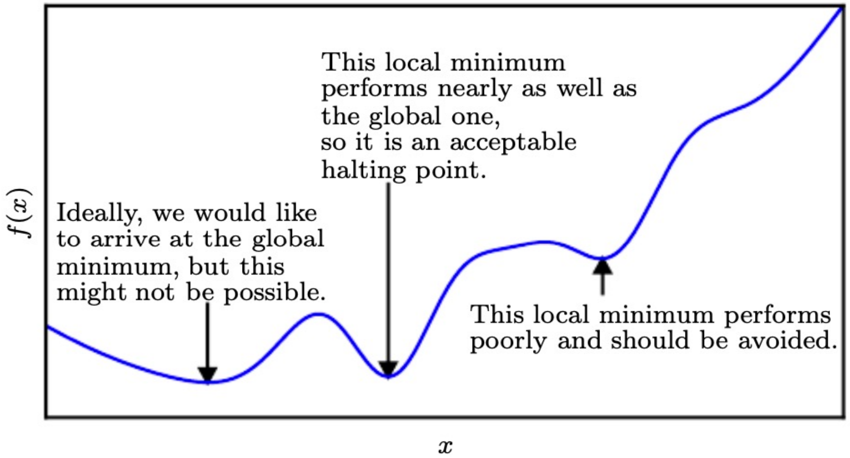


Image source: https://www.kdnuggets.com/2018/06/intuitive-introduction-gradient-descent.html

Gradient Descent

- When the cost function is not convex, optimization algorithms may fail to find a global minimum.



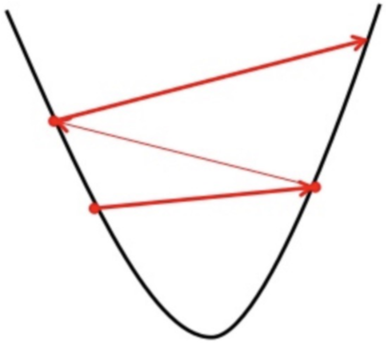
- In the context of deep learning, we generally accept such solutions even though they are not truly minimal, so long as they correspond to significantly low values of the cost function.

Image source: Figure 4.3, Goodfellow, Bengio, and Courville, Deep Learning, Cambridge: MIT press, 2016.

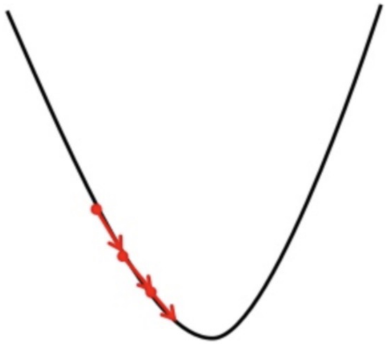
Learning Rate

- In the above updating formula, The size of these steps η is called the learning rate.
  - With a big learning rate, we can go with large step, but we risk overshooting the lowest point and resulting in non-convergence.
  - With a small learning rate, we can confidently move in the right direction, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.
- One strategy is to decrease the learning rate gradually on iteration.

Big learning rate



Small learning rate



Logistic Regression

- How can we use linear regression to do classification?
- The range of linear regression model is $(-\infty, +\infty)$.
- Can we map it into the range $[0, 1]$?

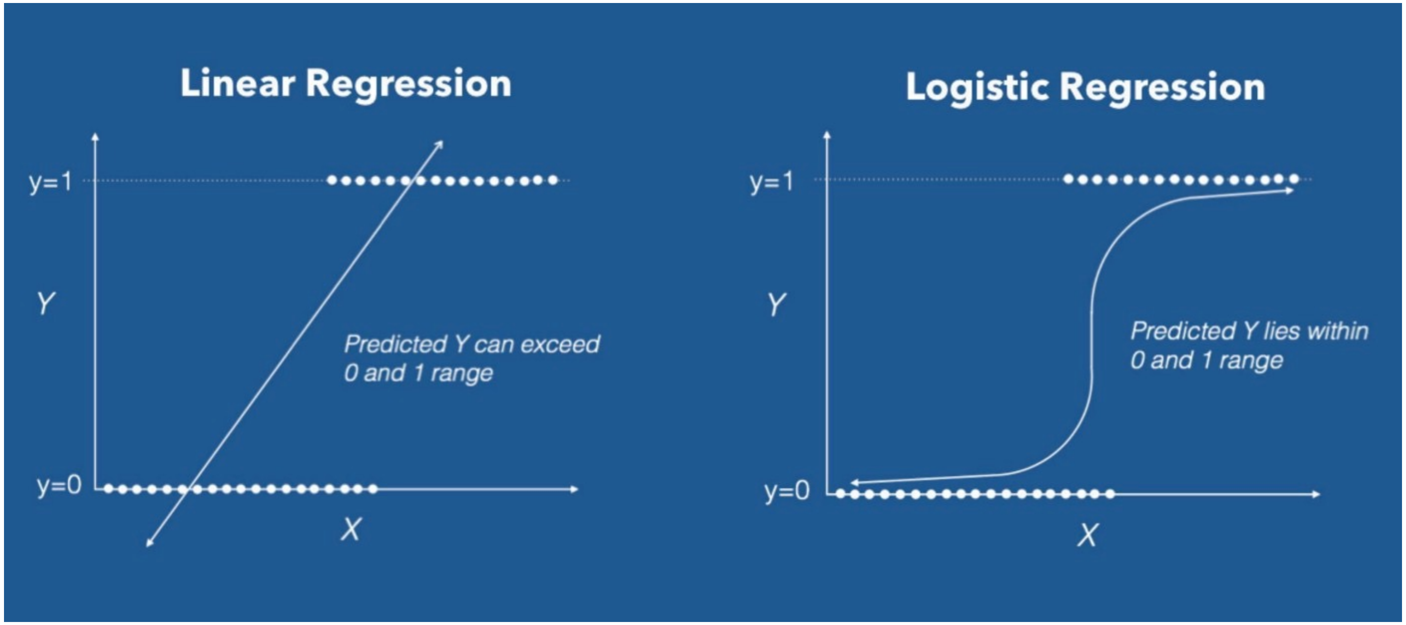


Image source: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Sigmoid Function

- We can make a new model by using the sigmoid function which maps $(-\infty, +\infty)$ to $(0, 1)$:

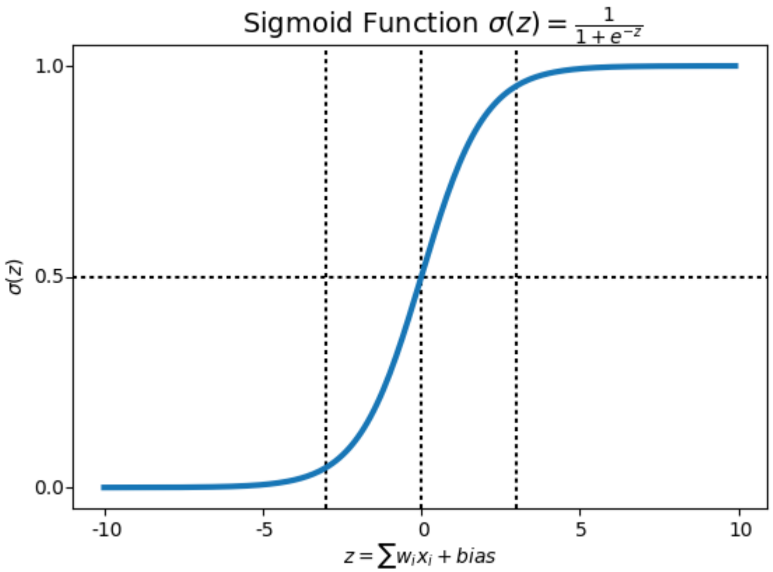
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

while $z = \boldsymbol{w}^T \boldsymbol{x} + b$.

- The sigmoid function can be used to represent the probability of each class

$$P(y = 1|z) = \sigma(z)$$
$$P(y = 0|z) = 1 - \sigma(z)$$

- Now, if $\sigma(z)$ is in $[0, 1]$.
  - If $\sigma(z) < 0.5$, we classify $\boldsymbol{x}$ as 0.
  - If $\sigma(z) \geq 0.5$, we classify $\boldsymbol{x}$ as 1.



The sigmoid function is also called logistic function

Image source: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

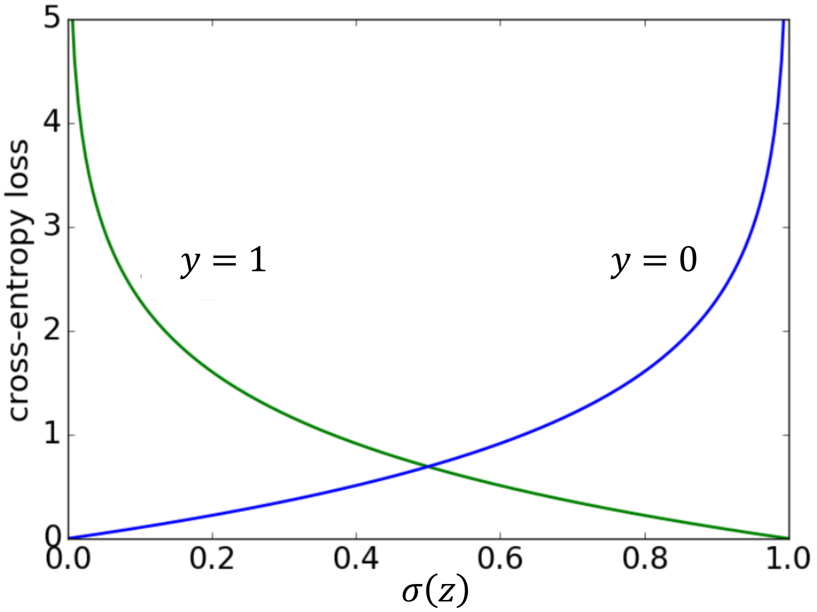
Cross-Entropy

- MSE is no longer suitable for measuring the error for a classification problem.
- Instead, we use cross-entropy loss (aka log loss or negative log-likelihood) as the cost function:

$$J = \frac{1}{n}\sum_{i=1}^{n} J(z_i)$$

$$J(z_i) = \begin{cases} -\log\sigma(z_i) & \text{if } y_i = 1 \\ -\log\big(1-\sigma(z_i)\big) & \text{if } y_i = 0 \end{cases}$$

$$= -y_i \log\sigma(z_i) - (1-y_i)\log\big(1-\sigma(z_i)\big)$$



Derivative of the Cross-Entropy Cost Function

- Calculate partial derivatives:

$$\frac{\partial J}{\partial \sigma} = \frac{\partial(-y\log\sigma - (1-y)\log(1-\sigma))}{\partial \sigma} = -\frac{y}{\sigma} + \frac{1-y}{1-\sigma} = \frac{\sigma - y}{\sigma(1-\sigma)}$$

$$\frac{\partial \sigma}{\partial z} = \frac{\partial \frac{1}{1+e^{-z}}}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2} = \sigma(1-\sigma).$$

- By the chain rule, we have:

$$\frac{\partial J}{\partial z} = \frac{\partial J}{\partial \sigma}\frac{\partial \sigma}{\partial z} = \frac{\sigma - y}{\sigma(1-\sigma)}\sigma(1-\sigma) = \sigma - y.$$

- Then, we can easily get $\partial J/\partial w_i$ and $\partial J/\partial b$ by using chain rule again with $\partial z/\partial w_i$ and $\partial z/\partial b$.

Iteration with Gradient Descent

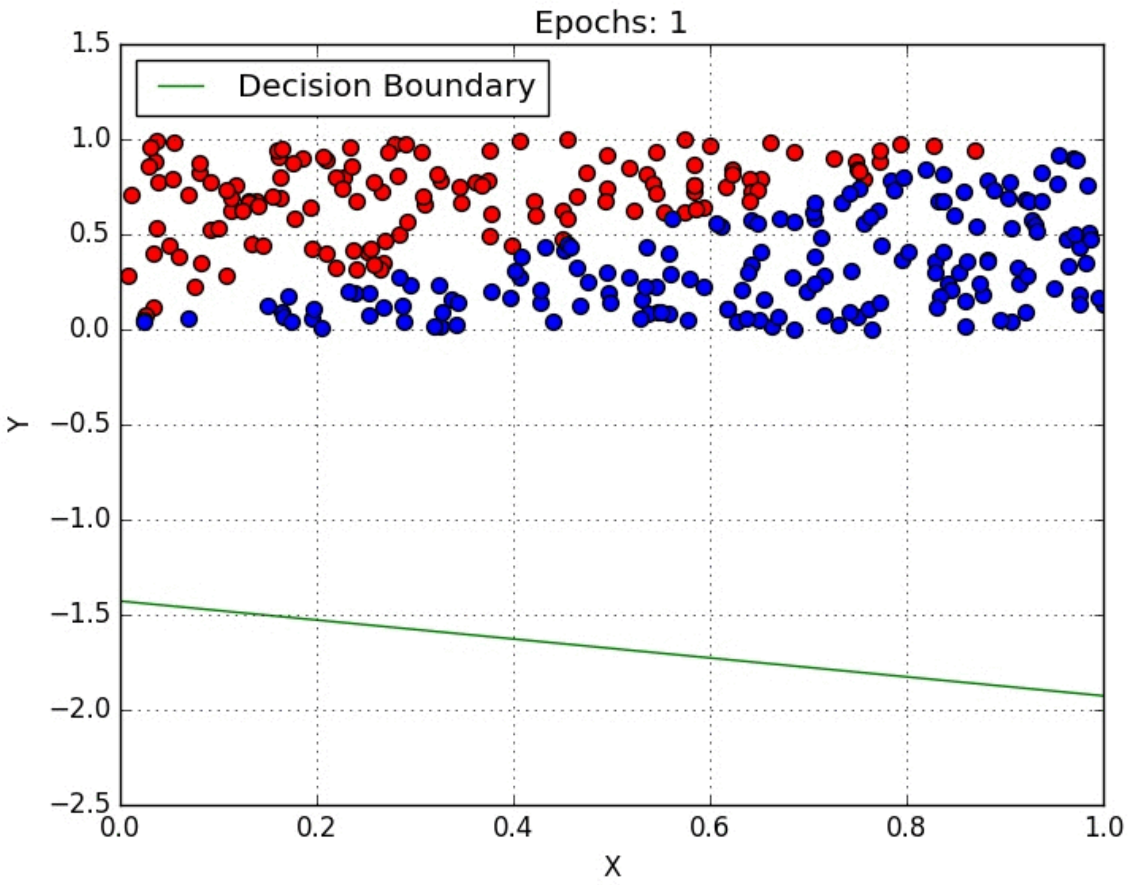


Image source: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

NEURAL NETWORKS

Limitation of Linear Models

- Linear models, such as logistic regression and linear regression, are appealing because they may be fit efficiently and reliably.
- Linear models also have the obvious defect that the model capacity is limited to linear functions, so the model cannot understand the interaction between any two input variables.
  - For example, the model $f(x) = w_1x_1 + w_2x_2 + b$ can never fit the function $f(x) = (x_1 + x_2)^2$.

XOR Problem

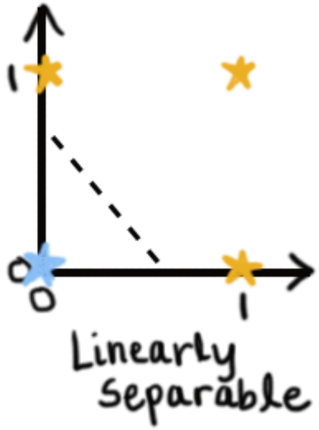
- Exclusive or (XOR) operation:

$$p \oplus q = (p \vee q) \wedge \neg(p \wedge q)$$

where

$$XOR(0,0) = 0 \quad XOR(1,1) = 0$$
$$XOR(1,0) = 1 \quad XOR(0,1) = 1$$

- Using a linear model (a line in 2d or a plane in 3d) can never correctly classify the XOR problem.



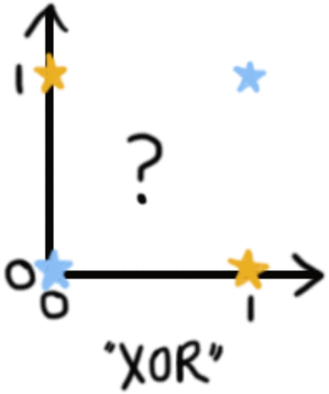
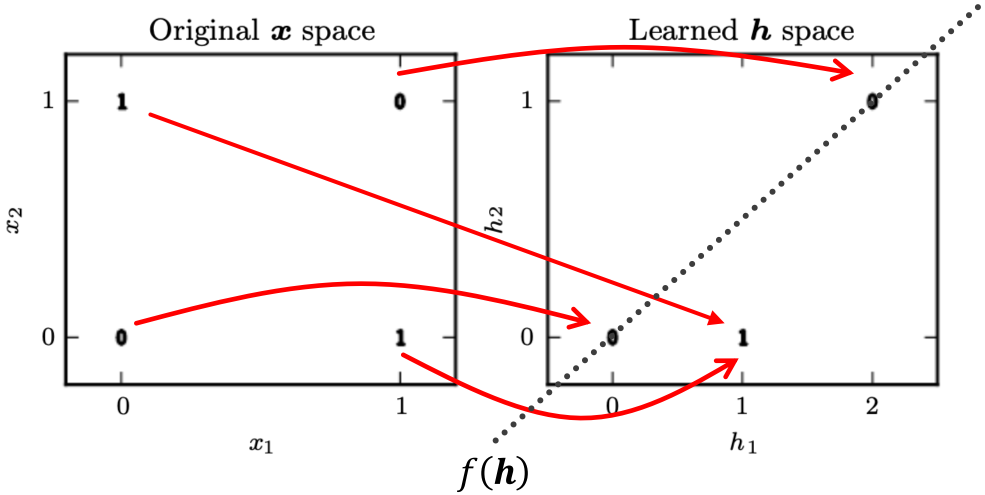


Image source: https://blog.insightdatascience.com/a-quick-history-of-neural-nets-from-inglorious-to-incredible-46e115c38b95

XOR Problem

- One way to solve this problem is to use a model that learns a different feature space in which a linear model is able to represent the solution.
- Do the following mapping:
  - (0,0) -> (0,0)
  - (0,1) -> (1,0)
  - (1,0) -> (1,0)
  - (1,1) -> (2,1)
- Then use the linear model:

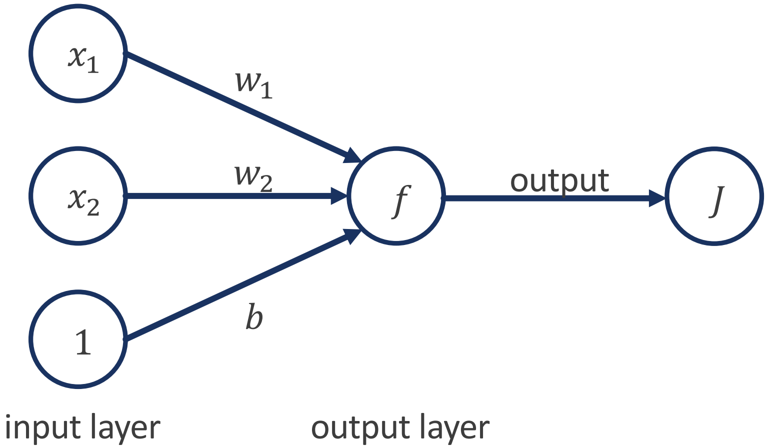
$$f(\boldsymbol{h}) = h_1 - 2h_2$$



How to achieve that? Can we add more layers?

Perceptron Model

- The previous linear model is also called perceptron model.
- This model has an input layer and an output layer.

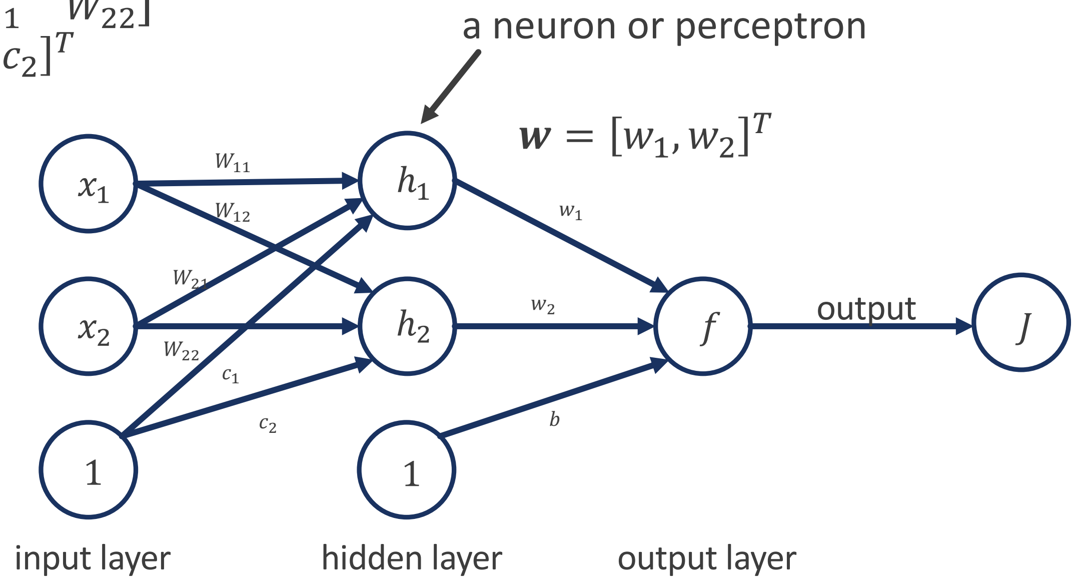

x_1
w_1
x_2
w_2
f
output
J
1
b
input layer
output layer

Feedforward Neural Networks

- The hidden layer is used as the input of output layer.

$$\boldsymbol{W} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$$

$$\boldsymbol{c} = [c_1, c_2]^T$$



a neuron or perceptron

$$\boldsymbol{w} = [w_1, w_2]^T$$

x_1
x_2
1
W_{11}
W_{12}
W_{21}
W_{22}
c_1
c_2
h_1
h_2
1
w_1
w_2
b
f
output
J
input layer
hidden layer
output layer

A multi-layer perceptron (MLP)

Feedforward Neural Networks

- However, this model is still linear because

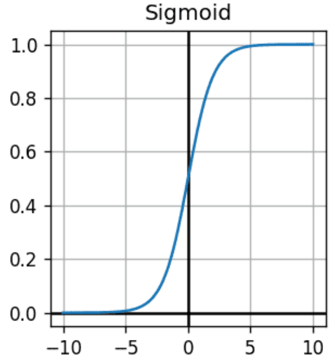
$$
\begin{aligned}
f(\boldsymbol{x}) &= \boldsymbol{w}^T\left(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{c}\right) + b \\
&= \boldsymbol{w}^T\boldsymbol{h} + b \\
&= w_1 h_1 + w_2 h_2 + b \\
&= w_1(W_{11}x_1 + W_{21}x_2 + c_1) \\
&\quad + w_2(W_{12}x_1 + W_{22}x_2 + c_2) + b \\
&= (\dots)x_1 + (\dots)x_2 + (\dots)
\end{aligned}
$$

Non-Linearity

- To break the linearity, we add an function called activation function to make it non-linear. We can define:

$$\boldsymbol{h} = g(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})$$

- Commonly adopted activation functions:
  - Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$.
  - Rectified Linear Unit (ReLU): $g(z) = \max\{0, z\}$.
- ReLU is the default activation function recommended for use with most feedforward neural networks.



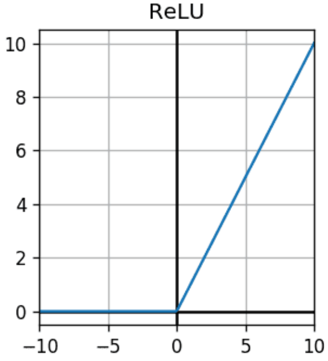
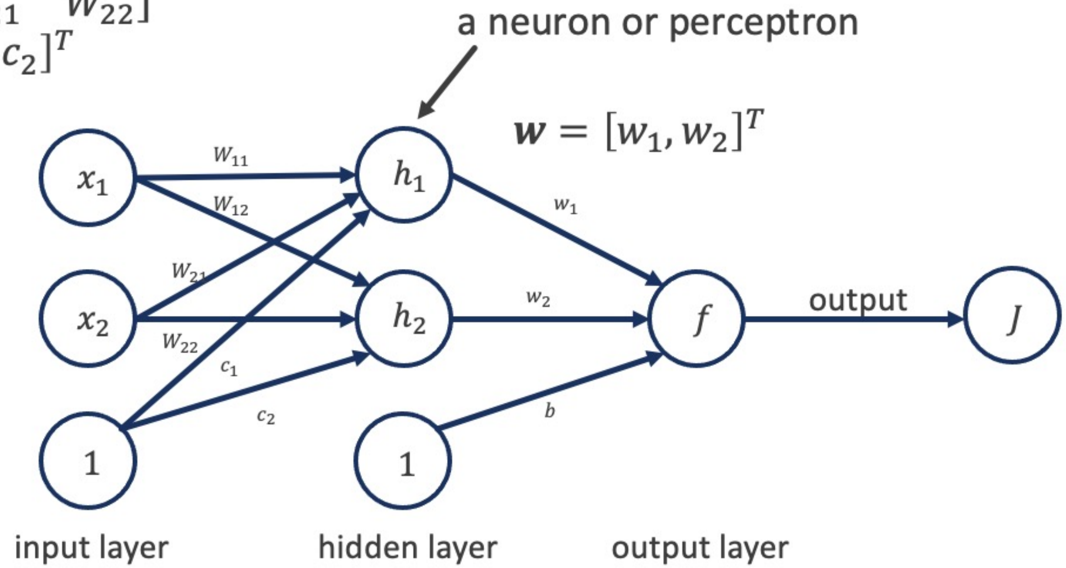


Image source: https://dhruvs.space/posts/ml-basics-issue-4/

Non-Linearity

$$\boldsymbol{W} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$$

$$\boldsymbol{c} = [c_1, c_2]^T$$


a neuron or perceptron
$\boldsymbol{w} = [w_1, w_2]^T$
x_1
x_2
1
W_{11}
W_{12}
W_{21}
W_{22}
c_1
c_2
h_1
h_2
1
w_1
w_2
b
f
output
J
input layer
hidden layer
output layer

- Thus, we have:

$$\begin{aligned} h_1 &= g(W_{11}x_1 + W_{21}x_2 + c_1) \\ h_2 &= g(W_{12}x_1 + W_{22}x_2 + c_2) \\ f(\boldsymbol{x}) &= w_1 h_1 + w_2 h_2 + b \\ &= \boldsymbol{w}^T g\left(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{c}\right) + b \end{aligned}$$

- If ReLU is adopted as the activation function, we have:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \max\{0, \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{c}\} + b$$

Non-Linearity

- We can now specify a solution to the XOR problem.
- After learning, we may obtain the model parameters:

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \qquad \boldsymbol{c} = [0, -1]^T, \qquad \boldsymbol{w} = [1, -2]^T, \qquad b = 0.$$

- Let $\boldsymbol{X}$ be the design matrix for batch input containing all four points in the binary input space. Each column is a sample.

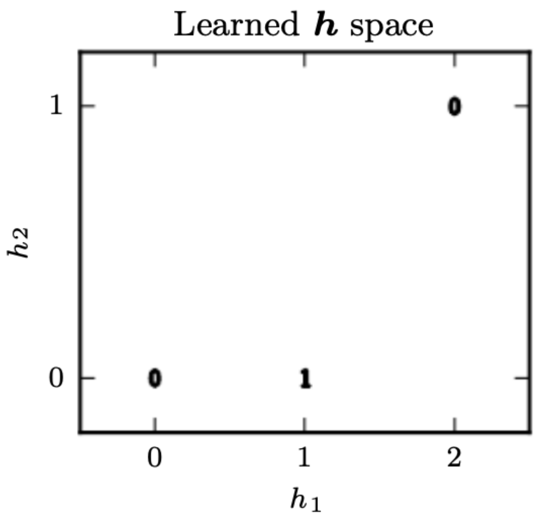
$$\boldsymbol{X} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

- We have the following calculation:

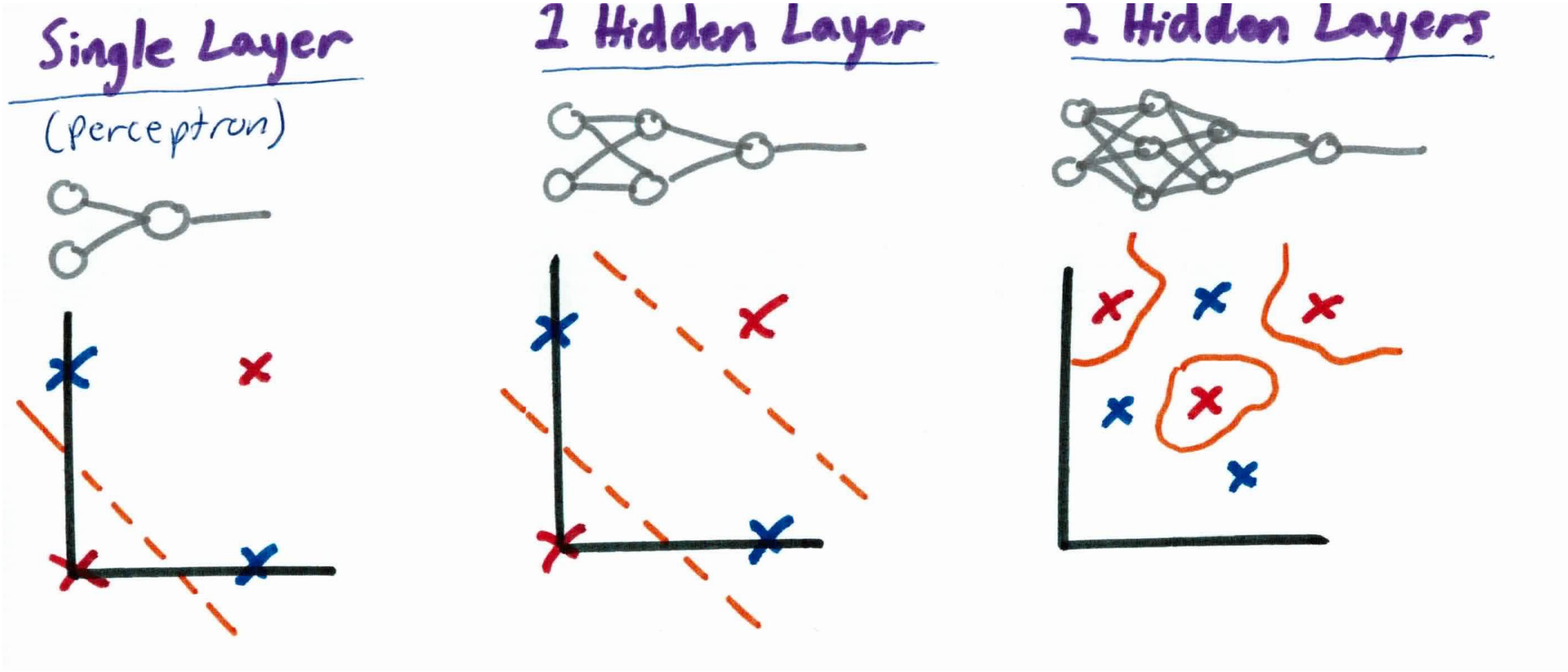
$$\boldsymbol{W}^T\boldsymbol{X} + \boldsymbol{c} = \begin{bmatrix} 0 & 1 & 1 & 2 \\ -1 & 0 & 0 & 1 \end{bmatrix},$$

$$\max\{0, \boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{c}\} = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{w}^T \max\{0, \boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{c}\} + b = [0 \quad 1 \quad 1 \quad 0].$$



Non-Linearity

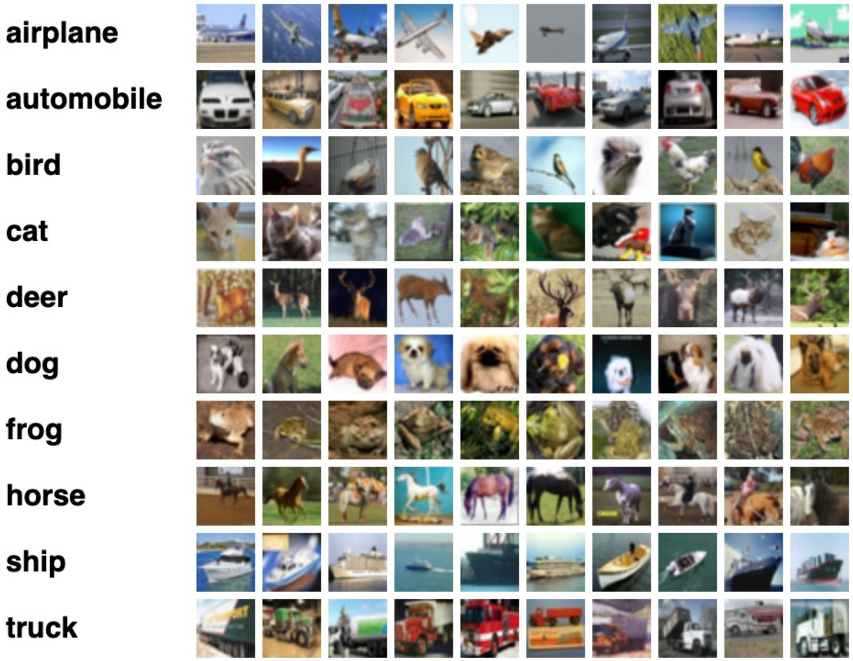

Single Layer
(perceptron)
1 Hidden Layer
2 Hidden Layers

Multi-layer perceptron (MLP)

Image source: https://kseow.com/nn

Multiclass Classification by Neural Networks

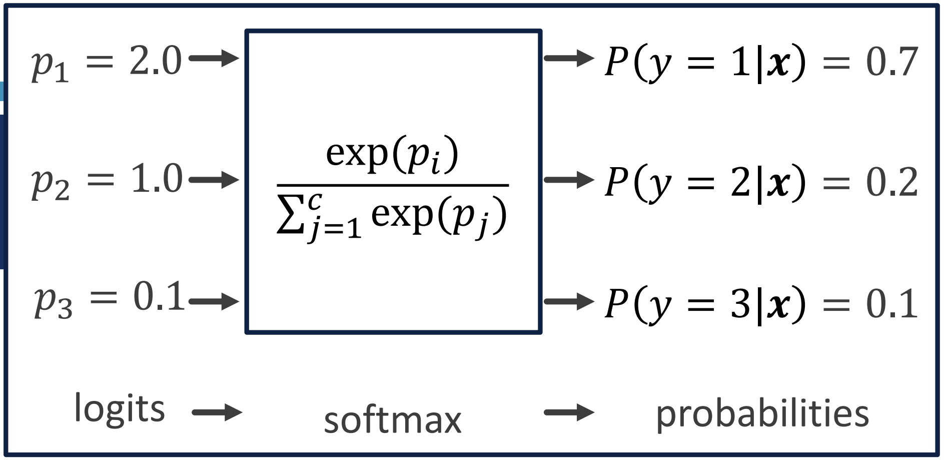
- For a binary classification problem, only one neuron in the output layer is enough.
  - It generates the probability of 0/1.
- For multiclass classification, we may have multiple neurons in the output layer. Each of them generates a score of one class.
  - Then we take the one with maximum score as the predicted class.



CIFAR10 dataset

Image source: https://www.cs.toronto.edu/~kriz/cifar.html

Softmax Function



$p_1 = 2.0 \rightarrow$

$p_2 = 1.0 \rightarrow$

$p_3 = 0.1 \rightarrow$

$\frac{\exp(p_i)}{\sum_{j=1}^{c} \exp(p_j)}$

$\rightarrow P(y = 1|\boldsymbol{x}) = 0.7$

$\rightarrow P(y = 2|\boldsymbol{x}) = 0.2$

$\rightarrow P(y = 3|\boldsymbol{x}) = 0.1$

logits → softmax → probabilities

- The softmax function is calculated by:

$$P(y = i|\boldsymbol{x}) = \frac{\exp(p_i)}{\sum_{j=1}^{c} \exp(p_j)}$$

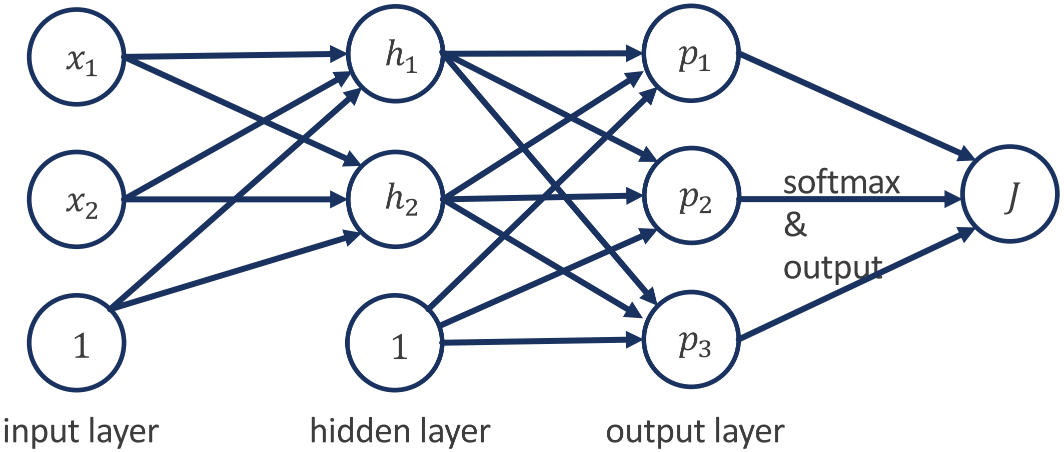
- p_i is the score (logit) of the ith class.
- The cross-entropy loss for multiclass classification:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{m} -[\![y_i = j]\!] \log P(y_i = j|\boldsymbol{x}_i)$$

where $[\![\cdot]\!]$ is the indicator function.

- When there are only two classes, softmax function reduces to sigmoid function.

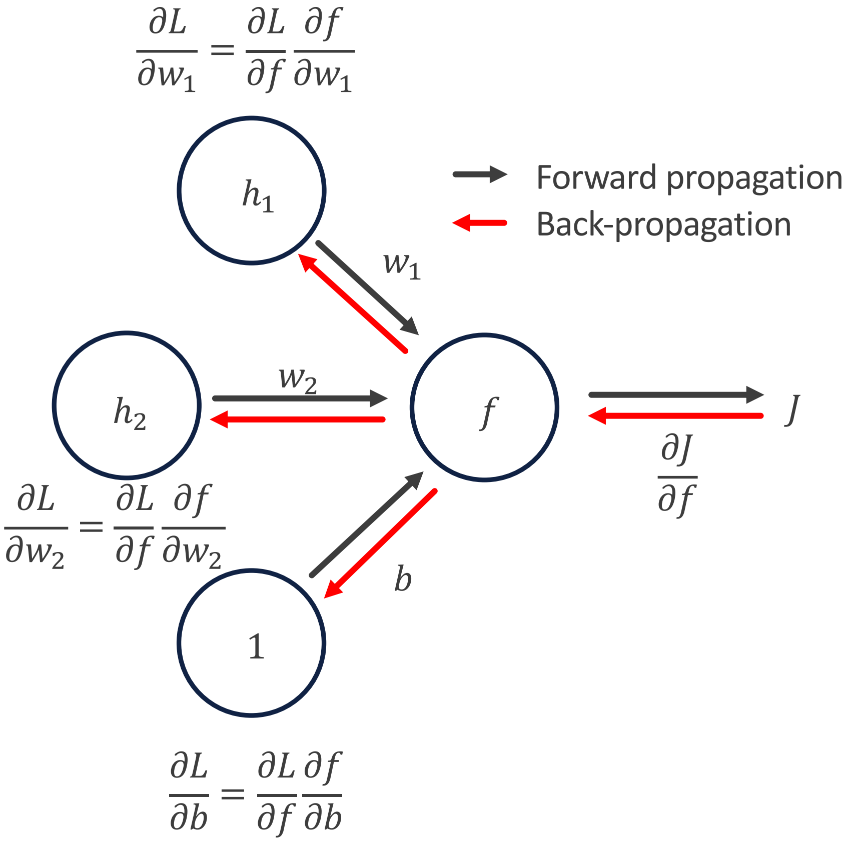
Multiclass Classification by Neural Networks


x_1
h_1
p_1
x_2
h_2
p_2
softmax
&
output
J
1
1
p_3
input layer
hidden layer
output layer

A two-layer neural network for 3-class classification

Back-propagation

- When we use a feedforward neural network to accept an input $\boldsymbol{x}$ and produce the output $f(\boldsymbol{x})$ and the cost J, information flows forward through the network. This is called forward propagation.
- The back-propagation (BP) algorithm often simply called backprop, allows the information from the cost to then flow backwards through the network, in order to compute the gradient.



$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_1}$
h_1
Forward propagation
Back-propagation
w_1
h_2
w_2
f
J
$\frac{\partial J}{\partial f}$
$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_2}$
b
1
$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial b}$

Image reference: https://www.kaggle.com/romaintha/an-introduction-to-backpropagation

Backpropagation

- Making use of the chain rule, we can express the gradient of J with respect to the weights and biases.
- For a multilayer perceptron model:.
  - Let $W_{ij}^{(l)}$ the weight connecting the ith neuron in the $(l-1)$th layer with jth neuron in the lth layer, $h_j^{(l)} = g(z_j^{(l)}), \boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)^T}\boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)}$.

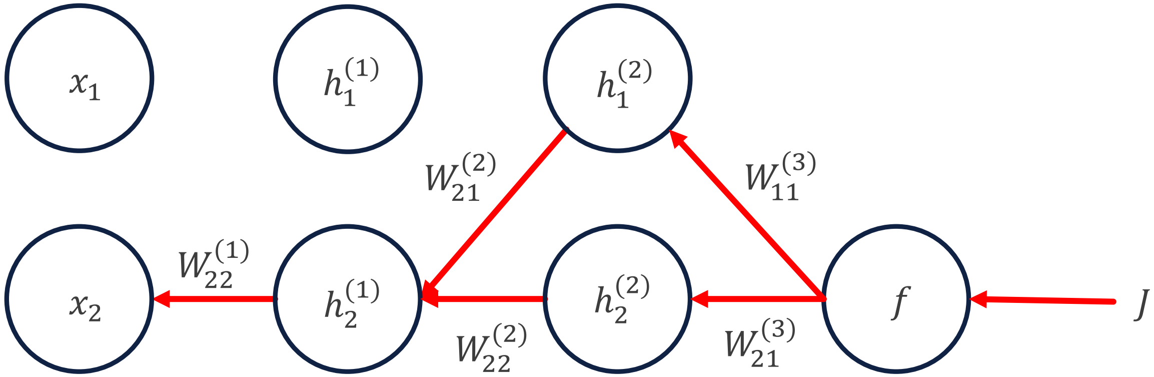
$$\frac{\partial J}{\partial W_{ij}^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial W_{ij}^{(l)}} = \frac{\partial J}{\partial h_j^{(l)}}\frac{\partial h_j^{(l)}}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial W_{ij}^{(l)}}$$

$$= \left(\boldsymbol{W}_{*j}^{(l+1)^T}\frac{\partial J}{\partial \boldsymbol{z}^{(l+1)}}\right) g'(z_j^{(l)}) h_i^{(l-1)}$$

recursively apply chain rule

Back-propagation

- Example of calculating $W_{22}^{(1)}$ by backprop.



$$\frac{\partial L}{\partial W_{22}^{(1)}} = \frac{\partial J}{\partial z_2^{(1)}} \frac{\partial z_2^{(1)}}{\partial W_{22}^{(1)}}$$

$$= \frac{\partial J}{\partial h_2^{(1)}} \frac{\partial h_2^{(1)}}{\partial z_2^{(1)}} \frac{\partial z_2^{(1)}}{\partial W_{22}^{(1)}}$$

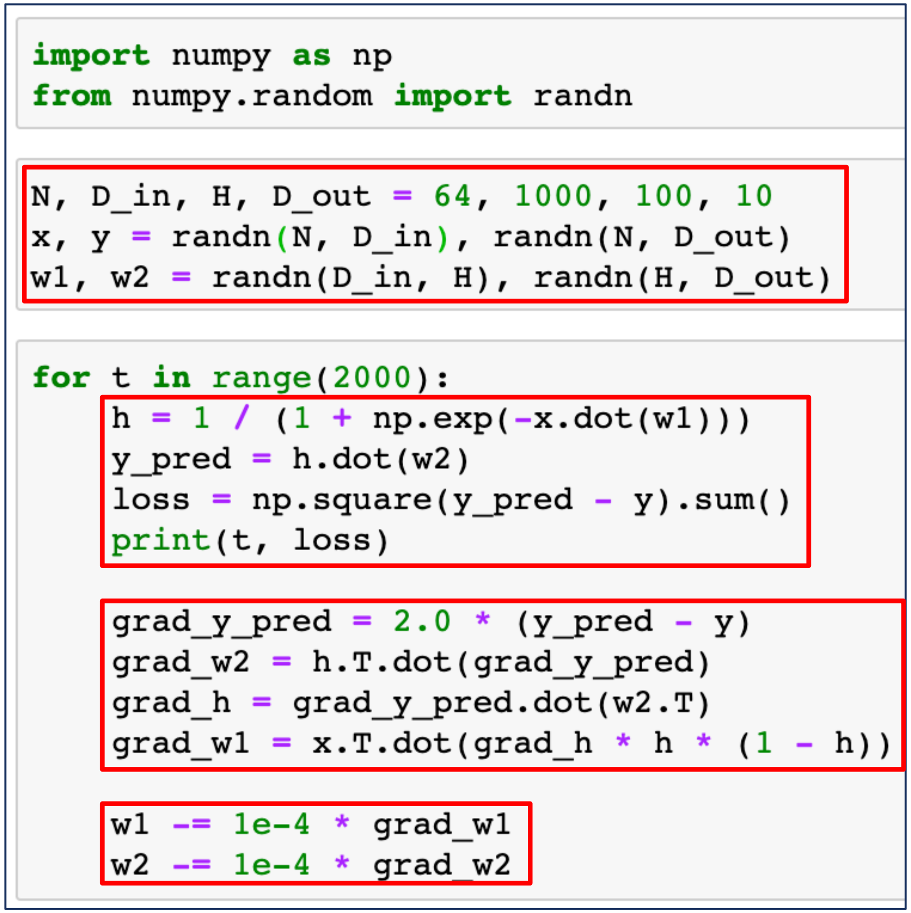
$$= \left(W_{21}^{(2)} \frac{\partial J}{\partial z_1^{(2)}} + W_{22}^{(2)} \frac{\partial J}{\partial z_2^{(2)}} \right) g'(z_2^{(1)}) x_2$$

$$\frac{\partial J}{\partial z_1^{(2)}} = \frac{\partial J}{\partial f} \frac{\partial f}{\partial h_1^{(2)}} \frac{\partial h_1^{(2)}}{\partial z_1^{(2)}} = \frac{\partial J}{\partial f} W_{11}^{(3)} g'\left(z_1^{(2)}\right)$$

$$\frac{\partial J}{\partial z_2^{(2)}} = \frac{\partial J}{\partial f} \frac{\partial f}{\partial h_2^{(2)}} \frac{\partial h_2^{(2)}}{\partial z_2^{(2)}} = \frac{\partial J}{\partial f} W_{21}^{(3)} g'(z_2^{(2)})$$

Image source: https://www.kaggle.com/romaintha/an-introduction-to-backpropagation

Full Implementation of Training a 2-layer Neural Network



```
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
    h = 1 / (1 + np.exp(-x.dot(w1)))
    y_pred = h.dot(w2)
    loss = np.square(y_pred - y).sum()
    print(t, loss)

    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h.T.dot(grad_y_pred)
    grad_h = grad_y_pred.dot(w2.T)
    grad_w1 = x.T.dot(grad_h * h * (1 - h))

    w1 -= 1e-4 * grad_w1
    w2 -= 1e-4 * grad_w2
```

Define the network

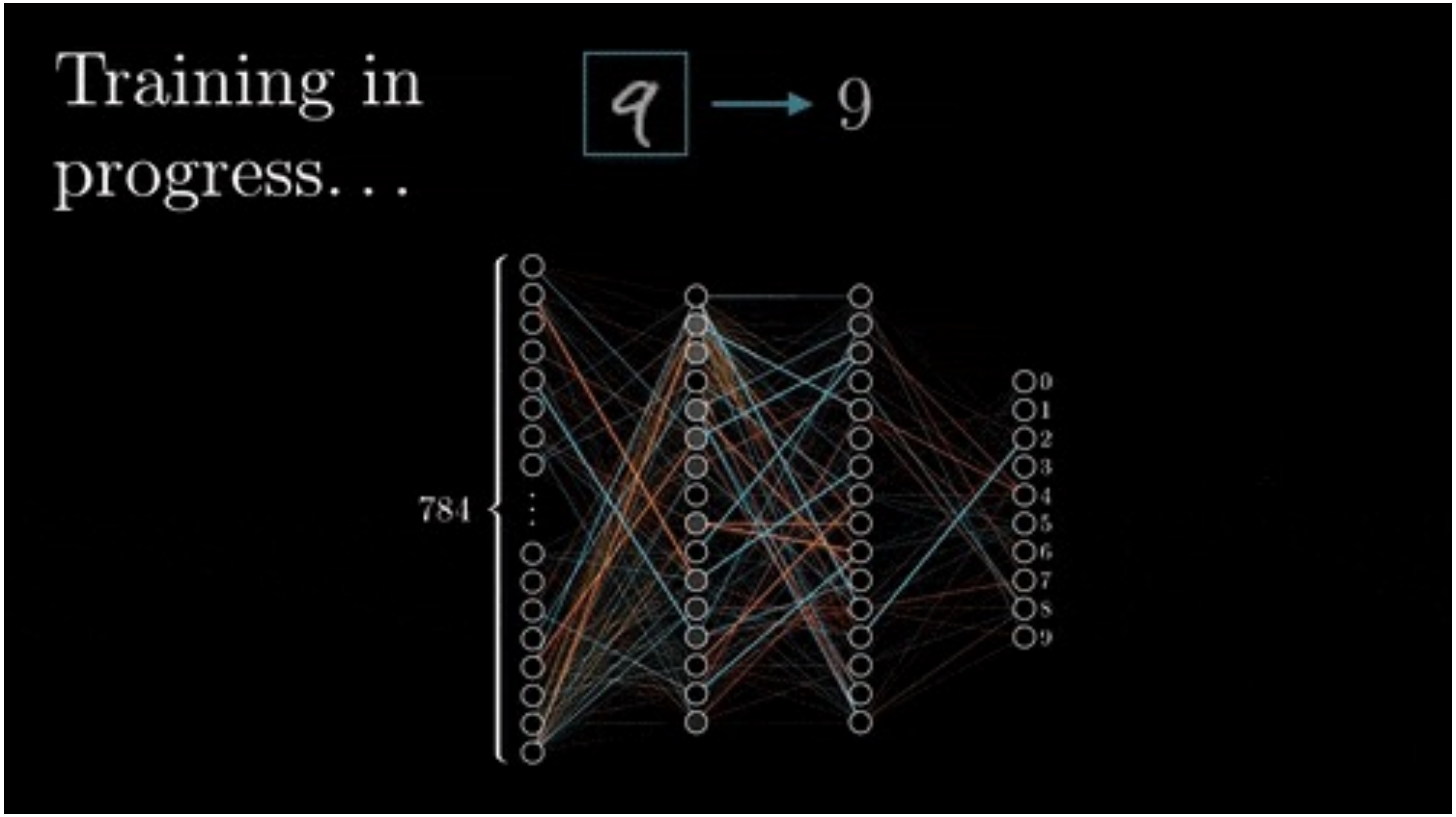
Forward pass

Calculate the gradient

Gradient descent

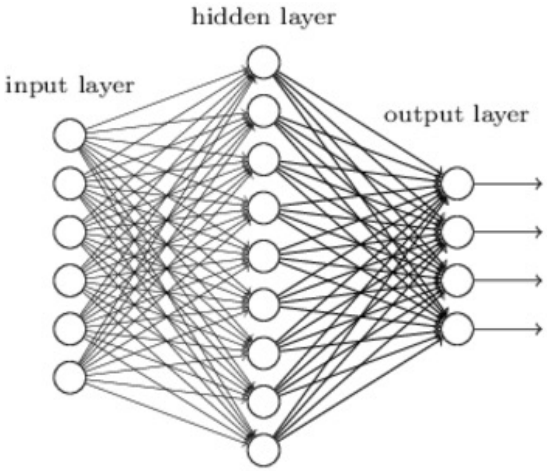
Code source: lecture 4, cs231n

Back-propagation


Training in progress. . .
9
9
784
0
1
2
3
4
5
6
7
8
9

From Neural Network to Deep Learning

"Non-deep" feedforward neural network


hidden layer
input layer
output layer

Deep neural network

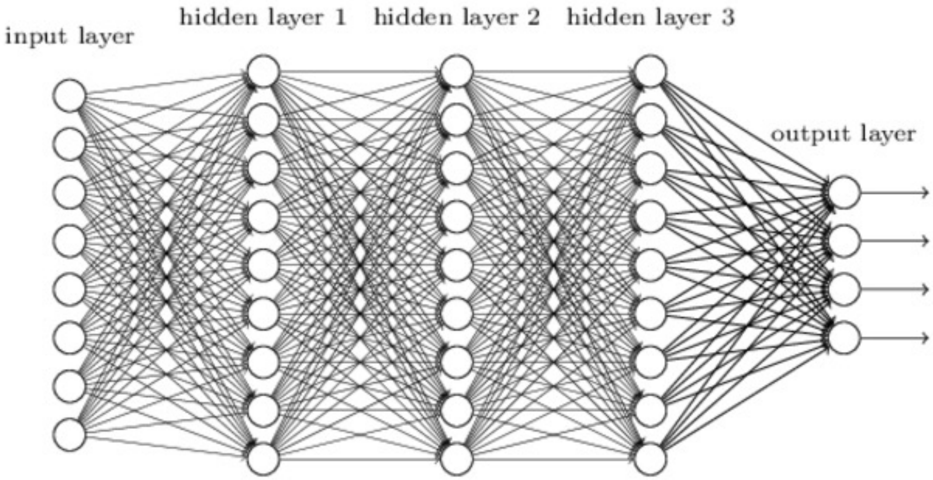

input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

Image source: https://www.quora.com/What-is-the-difference-between-deep-and-shallow-neural-networks

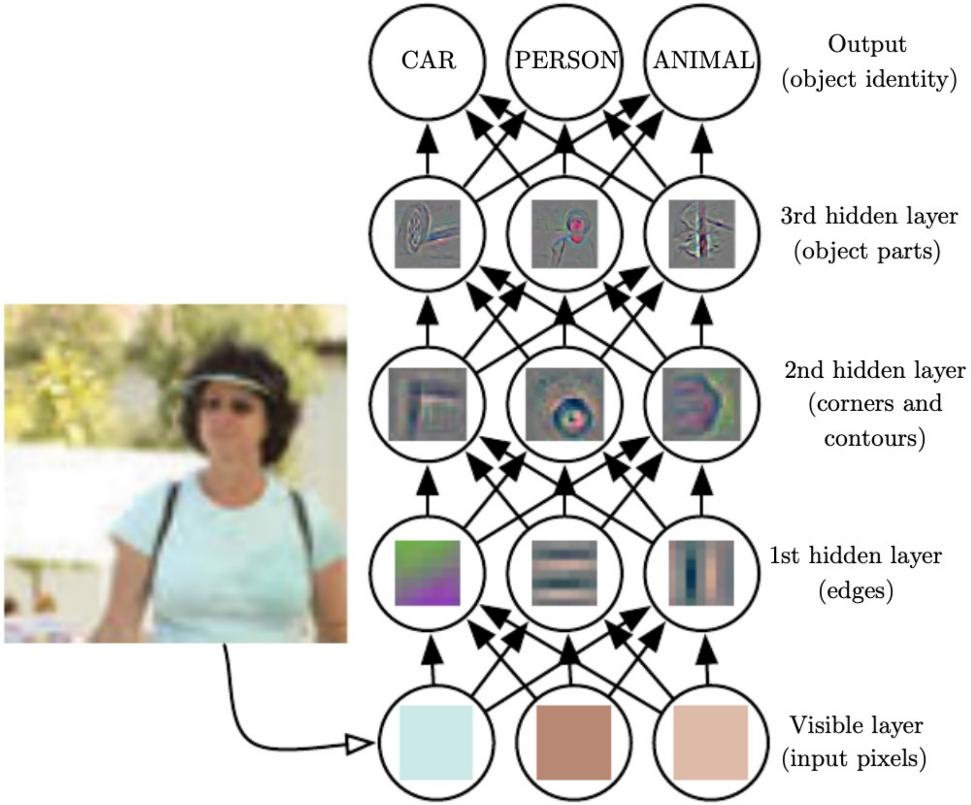
From Neural Network to Deep Learning

- Non-deep learning commonly refers to traditional machine learning.
- It follows a typical two-stage learning pattern.
  - Feature extraction.
  - Model building.
- Feature extraction highly depends on human knowledge, which determines the upper bound of a model can predict.

From Neural Network to Deep Learning

- Deep learning enables feature learning or representation learning.
- Deep learning usually follows end-to-end learning pattern.
  - The features and high level representations are learned from the original raw data.
  - The learned features and representations are stored in the hidden layers.
- The whole process is in one framework.

From Neural Network to Deep Learning


CAR
PERSON
ANIMAL
Output
(object identity)
3rd hidden layer
(object parts)
2nd hidden layer
(corners and
contours)
1st hidden layer
(edges)
Visible layer
(input pixels)

Visualizations of feature represented by each hidden unit in a deep neural network

Image source: Figure 1.2, Goodfellow, Bengio, and Courville, Deep Learning, Cambridge: MIT press, 2016.

MODEL SELECTION

Hyperparameter Tuning

- Almost every model has some hyperparameters.
  - C and λ in SVM.
  - Number of layers and neurons on each layer in neural networks.
  - Number of trees and size of feature subset in random forest.
- Notice the difference between hyperparameters and parameters!
  - Sometimes hyperparameters are also called parameters, e.g. regularization parameter.
- For different datasets, the best hyperparameters are different.
  - We need to select them by trial and error.
- This process is called hyperparameter tuning or model selection.

Parameter Tuning

- However, we can’t select hyperparameters based on the performance on the test data.
  - It is like cheating if you know questions and answers of the final exam and then adjust your study plan.
- Generally, we have two strategies:
  - K-fold cross validation.
  - Fixed validation set.

K-Fold Cross Validation

- We cut the training data into K folds, where
  - K-1 folds are used for training.
  - 1 fold is used for validation.
  - Repeat K times with different combinations and select the parameter with the highest average performance.
- For example, 3-fold cross validation will separate the training data into 5 folds.
  - Train on fold [1, 2] and test on [3].
  - Train on fold [1, 3] and test on [2].
  - Train on fold [2, 3] and test on [1].
- After selecting the best hyperparameters, train the model again with all training data.

Fixed Validation Set

- If the dataset is large enough, we can fix the training/validation/test data.
  - 8-1-1 split is usually adopted.
- In this way, we can compare training/validation/test error, because the size of training data is same.
  - The data size used for training in cross validation is actually shrinked.

Grid Search

- If you have only one hyperparameter, simply try all the values you want.
  - Learning rate in logistic regression: [0.0001, 0.001, 0.01, 0.05, 0.1, 0.5].
- If you have multiple hyperparameters, we should try all the combinations of them. This is called grid search.
  - Learning rate in logistic regression: [0.0001, 0.001, 0.01, 0.05, 0.1, 0.5].
  - Regularization parameter: [0.001, 0.01, 0.1, 0, 1, 10, 100]
  - Totally $6\times 7 = 42$ combinations should be tried.

MODEL EVALUATION

Evaluation Metric

- Evaluating your machine learning algorithm is an essential part of any project.
- For different applications, we may adopt different evaluation metrics.
- Most of the times we use classification accuracy to measure the performance of our model, however it is not enough to truly judge our model.

Classification Accuracy

- Classification accuracy is what we usually mean, when we use the term accuracy.
- It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}$$

- However, sometimes accuracy cannot well reflect the performance.
  - Consider training a disease diagnose system that there are 98% samples of healthy people and 2% samples of patients in our training set.
  - Then our model can easily get **98% training accuracy** by simply predicting every training sample as healthy.

Confusion Matrix

- Confusion matrix outputs and describes the complete performance of the model.
- Confusion matrix forms the basis for the other types of metrics.

		Ground Truth	
Prediction		Positive (sick)	Negative (healthy)
	Positive (sick)	True Positive (TP)	False Positive (FP)
	Negative (healthy)	False Negative (FN)	True Negative (TN)

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision and Recall

- Precision refers to how much of your positive predictions are correct. It doesn’t care if you cover all of the positive samples.

$$Precision = \frac{TP}{\boxed{TP + FP}}.$$

Total positive prediction

  - For the previous example, if you only predict one sample as sick and it is correct, the precision will be 1.
- Recall (aka sensitivity, TPR) refers to how much of your positive samples are correctly classified. It doesn’t care how many positive predictions you make.

$$Recall = \frac{TP}{\boxed{TP + FN}}.$$

Total positive samples

  - For the previous example, if you predict all of the samples as sick, the recall will be 1.

F1 Score

- Therefore, a good classifier should take a balance between precision and recall.
- We use F1 score to measure this trade-off. It is the harmonic mean of precision and recall.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

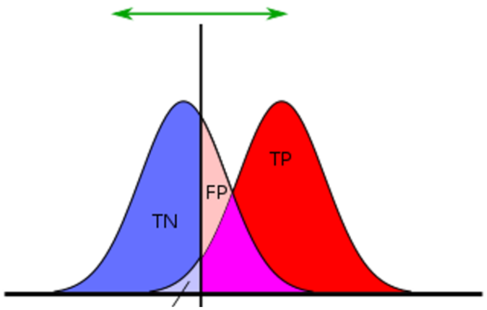
- F1 score will be high only if both precision and recall be high.
  - One of them is close to 0 will make F1 score close to 0.

Receiver Operating Characteristic

- Another trade-off can be made between TPR (recall) and FPR.

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN}.$$

- By moving the threshold for binary classification, the change of TPR and FPR can be drawn in a figure. It is called Receiver Operating Characteristic (ROC).
- Usually, we calculate the Area Under Curve (AUC) of ROC to compare.



TP	FP
FN	TN

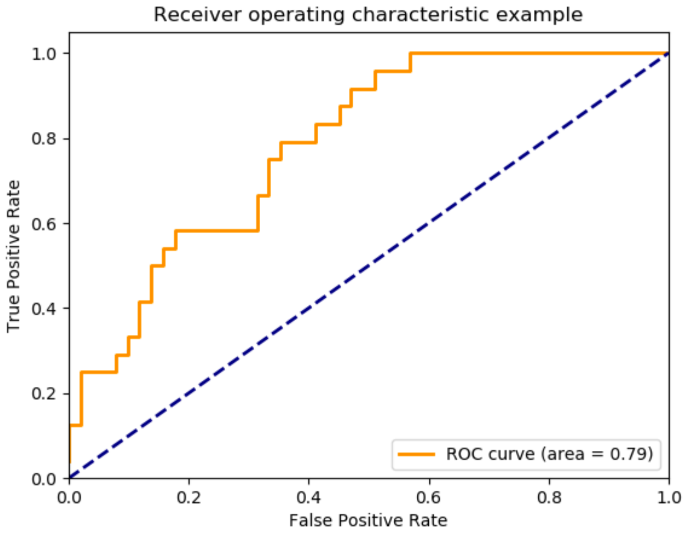


Image source: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Conclusion

After this lecture, you should know:

- What is machine learning.
- What is a typical machine learning process.
- How do linear models work.
- How does gradient descent work.
- How do neural networks work.
- How does backprop work.

Suggested Reading

- Deep learning textbook chapter 5-6.
- Detailed derivation of backprop: http://neuralnetworksanddeeplearning.com/chap2.html

Thank you!

- Any question?
- Don’t hesitate to send email to me for asking questions and discussion. ☺