Introduction of Deep Learning

Sanzhen Liu

PLPTH813

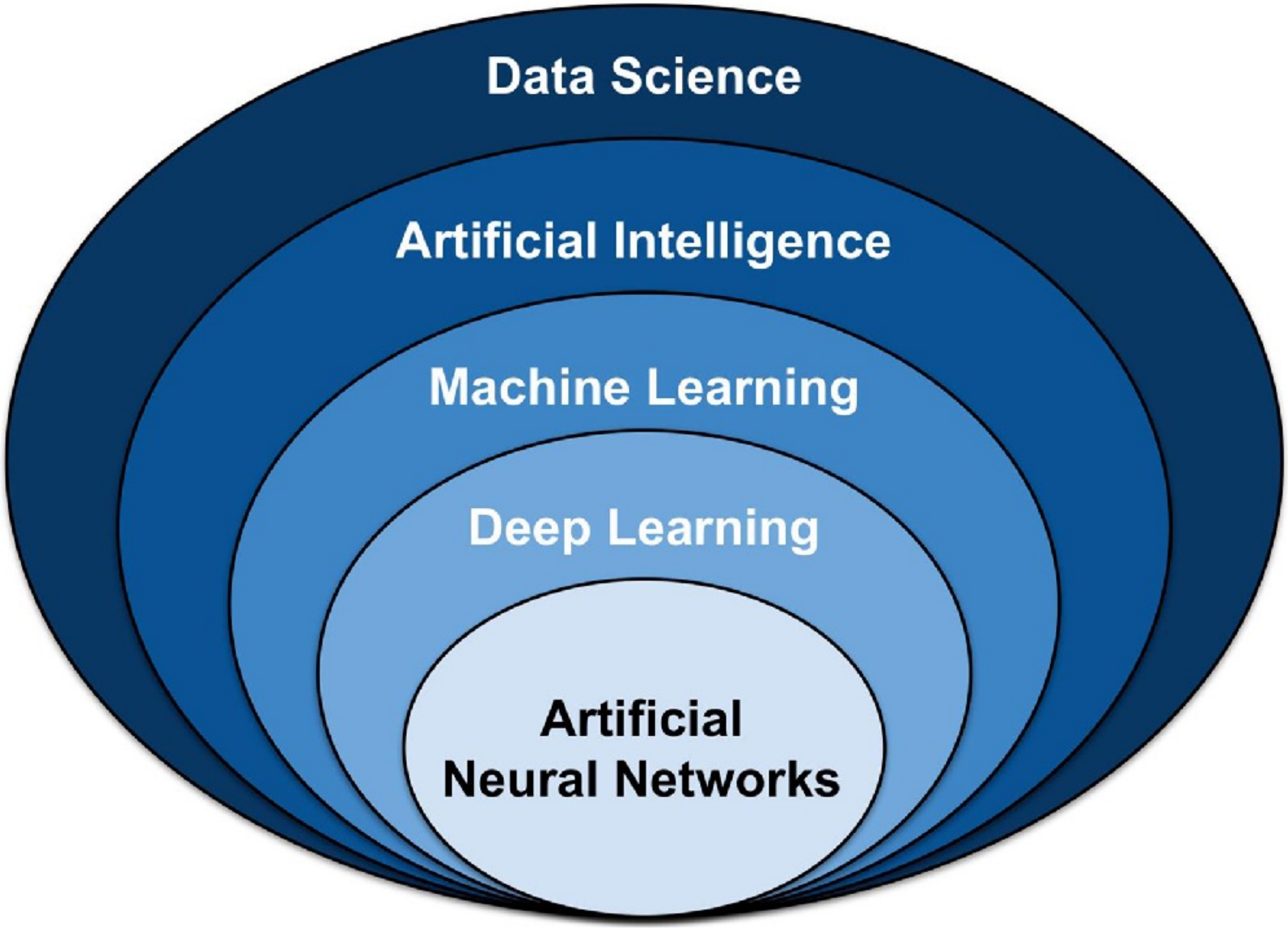
4/29/2021

Goals

To learn:

- What is machine learning or deep learning?
- How does neural network work?
- What do we need to know for applying machine learning in our studies?
- How to run a simple machine learning using R? (lab)

Machine learning



Translational Vision Science & Technology, 2020, 9:14.

Supervised and unsupervised (major methods)

- Unsupervised (to learn inherent patterns within data)

e.g., Clustering, principal component analysis

- Supervised: to predict response (regression) or classification of each data point by using a provided set of labeled training examples

Supervised: regression and classification

A **regression** model predicts continuous values. For example, regression models make predictions that answer questions like the following:

- What is the value of a house in California?
- What is the probability that a user will click on this ad?

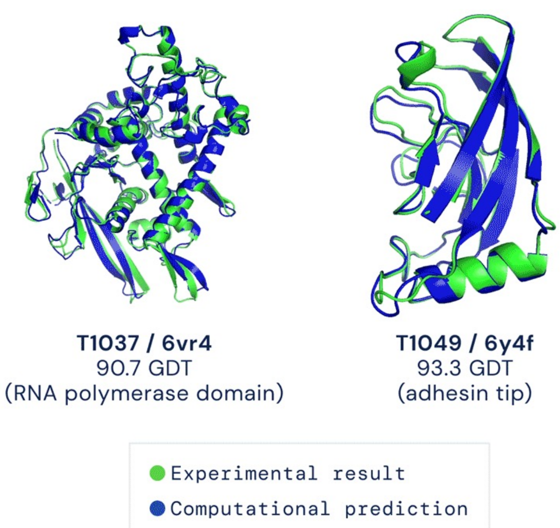
A **classification** model predicts discrete values. For example, classification models make predictions that answer questions like the following:

- Is a given email message spam or not spam?
- Is this an image of a dog, a cat, or a hamster?

Why machine learning

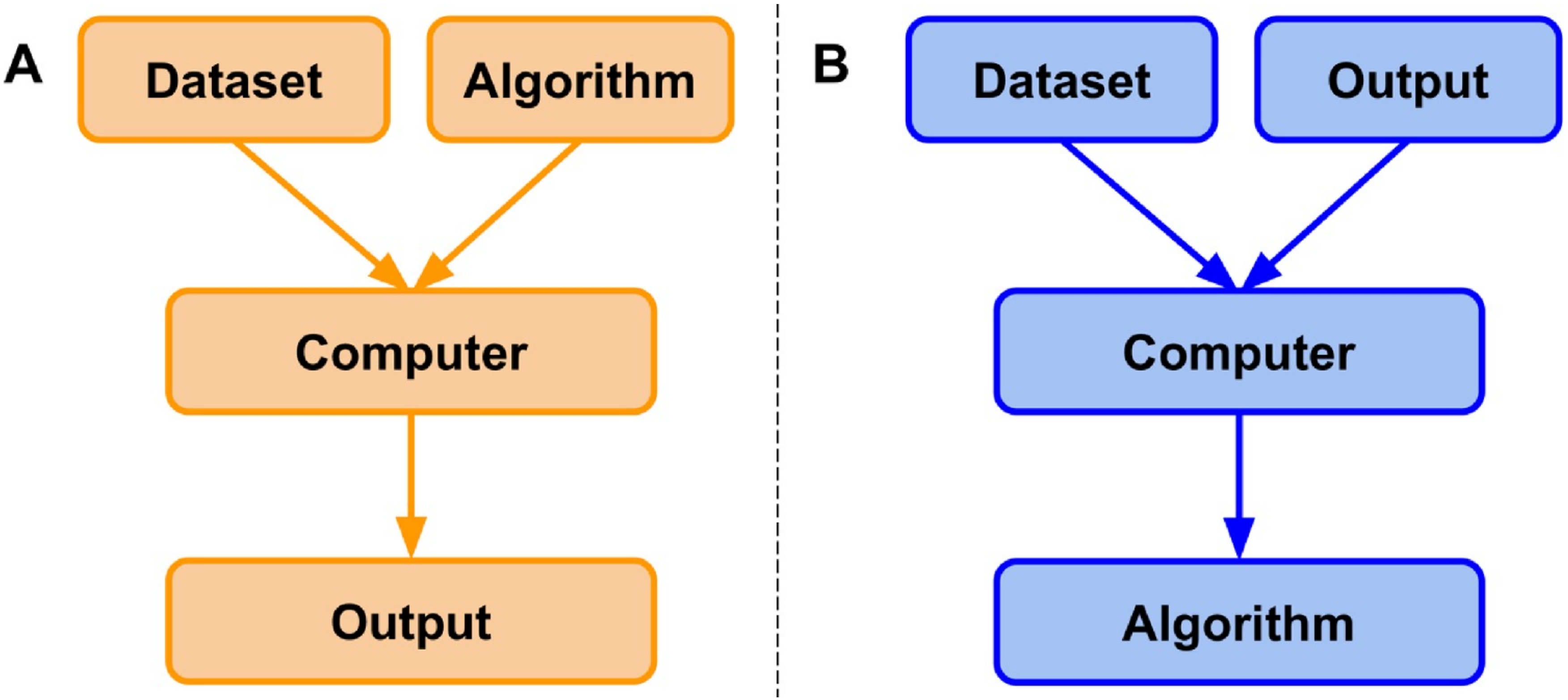
- Advance of algorithms and theories
- Increasing computational power
- Large data

autopilot



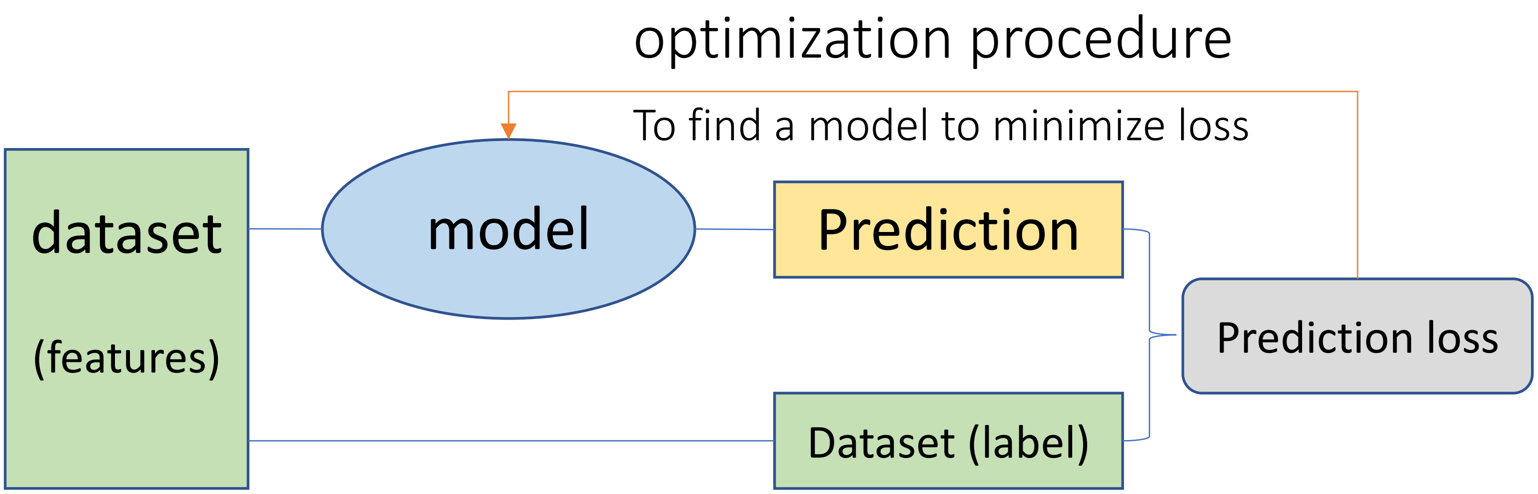
AlphaFold

Classical programming vs. machine learning paradigm



Translational Vision Science & Technology, 2020, 9:14.

General procedure



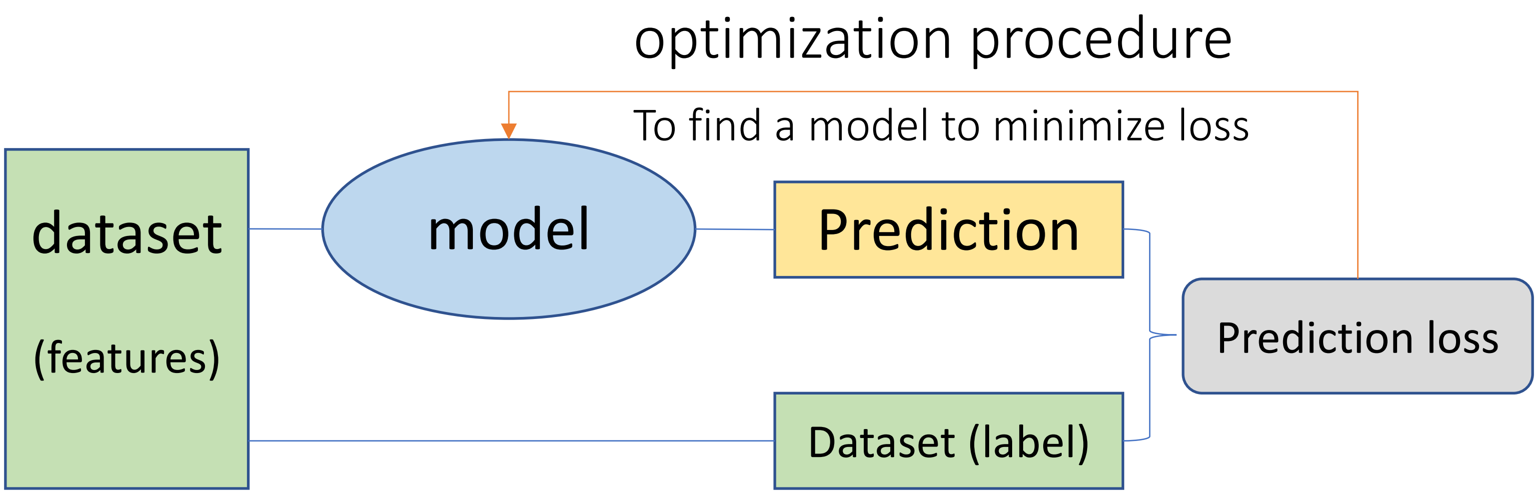
location, room, area, … house price

edu, org, spam_words, … spam (0 or 1)

Components

Common components of nearly all machine learning algorithms:

- Dataset (training data)
- cost function
- optimization procedure
- model



Training data

- Labels

A **label** is the thing we're predicting—the y variable in simple linear regression.

e.g., the price of wheat, the kind of animal shown in a picture

- Features

A **feature** is an input variable—the x variable in simple linear regression. A machine learning project could use millions of features, specified as:

$$x_1, x_2, \dots x_N$$

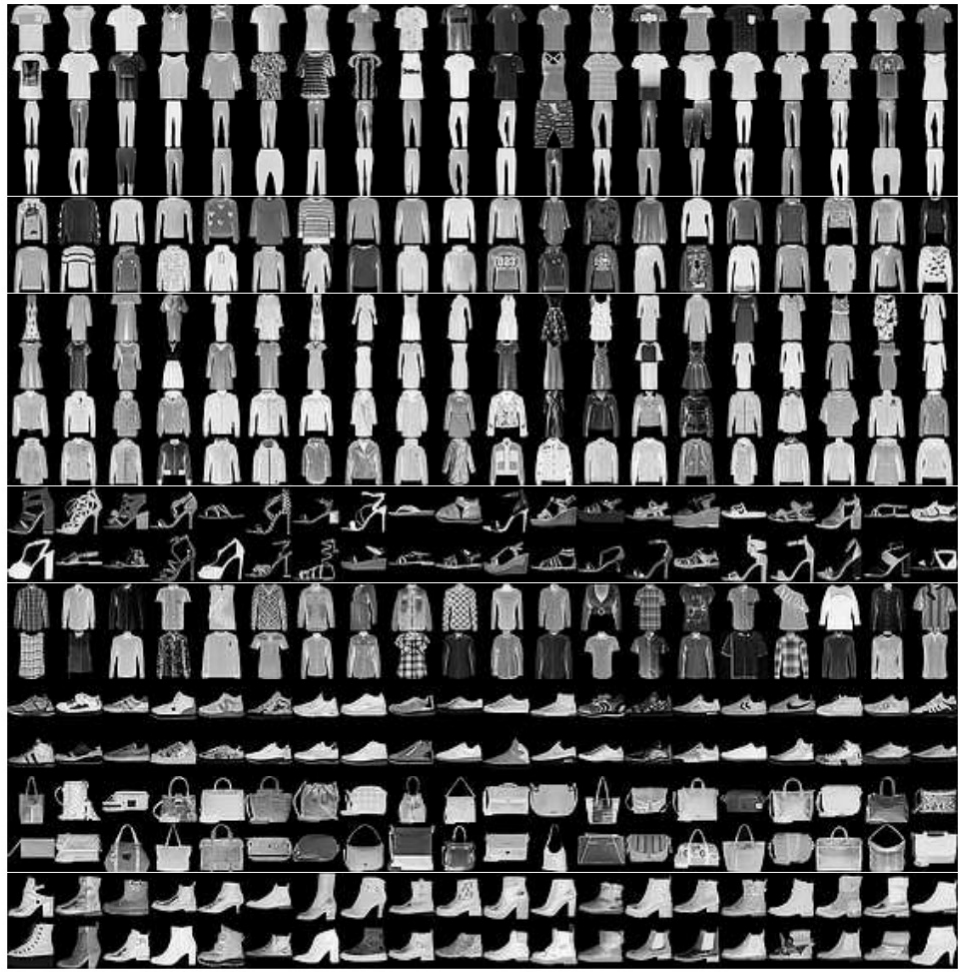
- Examples

An **example** is a particular entry of data $(\boldsymbol{x}, \boldsymbol{y})$

Training data - example

Table 2: Class names and example images in Fashion-MNIST dataset.

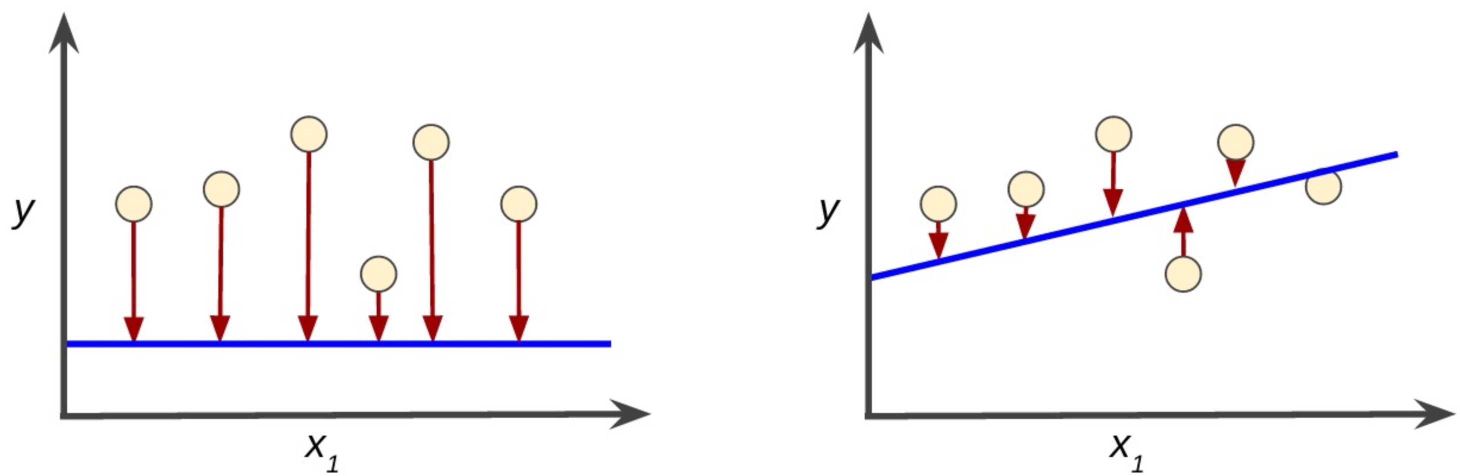
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	



https://arxiv.org/pdf/1708.07747.pdf

Cost function for a simple linear regression

Loss is the penalty for a prediction.

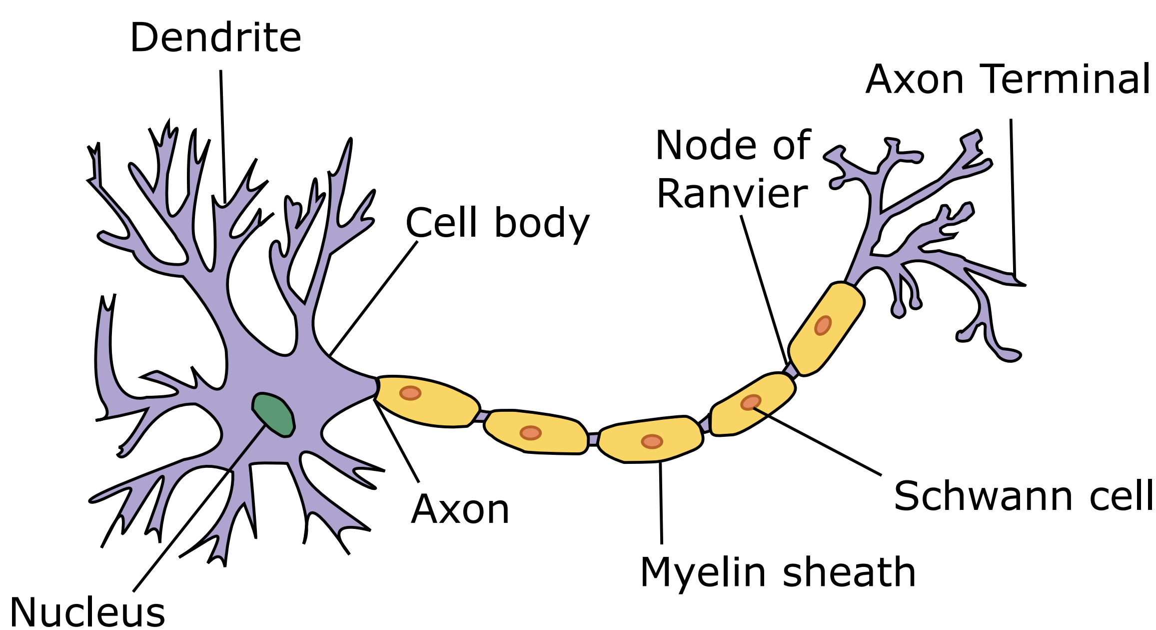


Loss function: $$MSE = \frac{1}{N}\sum(y - \text{prediction}(x))^2$$

Training a model is to find a set of weights (parameters) that have a *low* loss, on average, across all examples.

Neural networks: a main form of deep learning

- Deep learning refers to the recent advances in neural networks and the corresponding training platform
- Algorithms trying to mimic the brain
- Widely used in 80s and early 90s; popularity diminished in late 90s but resurged recently

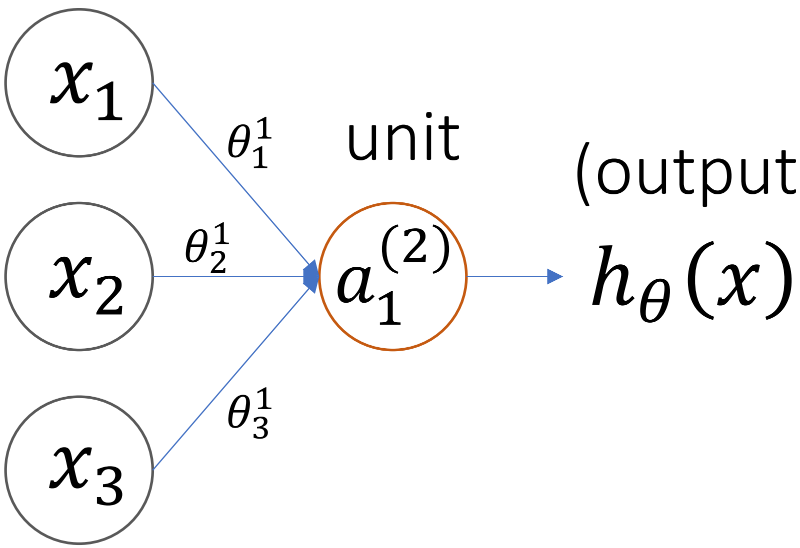


Neuron model

Features
(input)

θ: parameters or weights

$$z_1 = \theta^T x = \theta_1^1 x_1 + \theta_2^1 x_2 + \theta_3^1 \, x_3$$

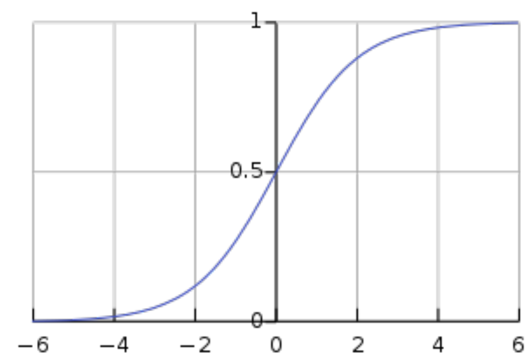


Activation function
E.g., Sigmoid (logistic)

$$a_1^{(2)} = h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

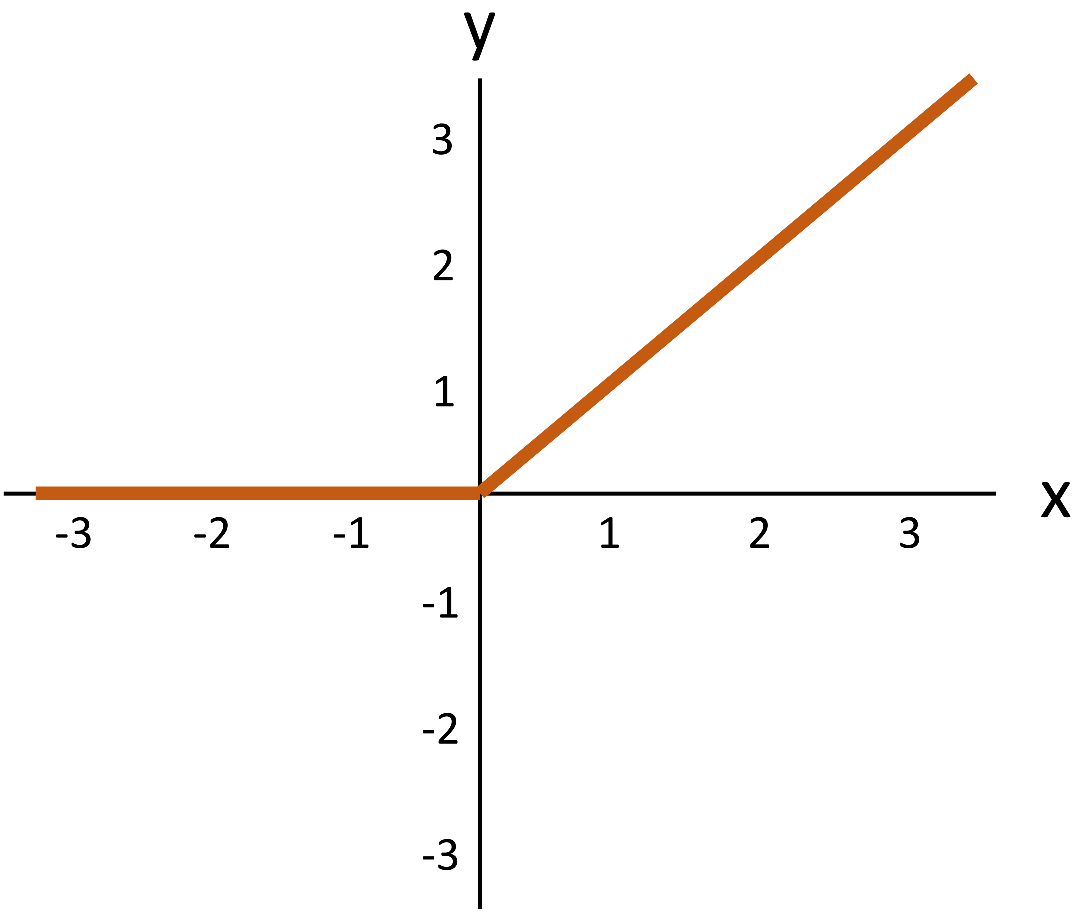
Layer 1
Input layer

Layer 2
Output layer



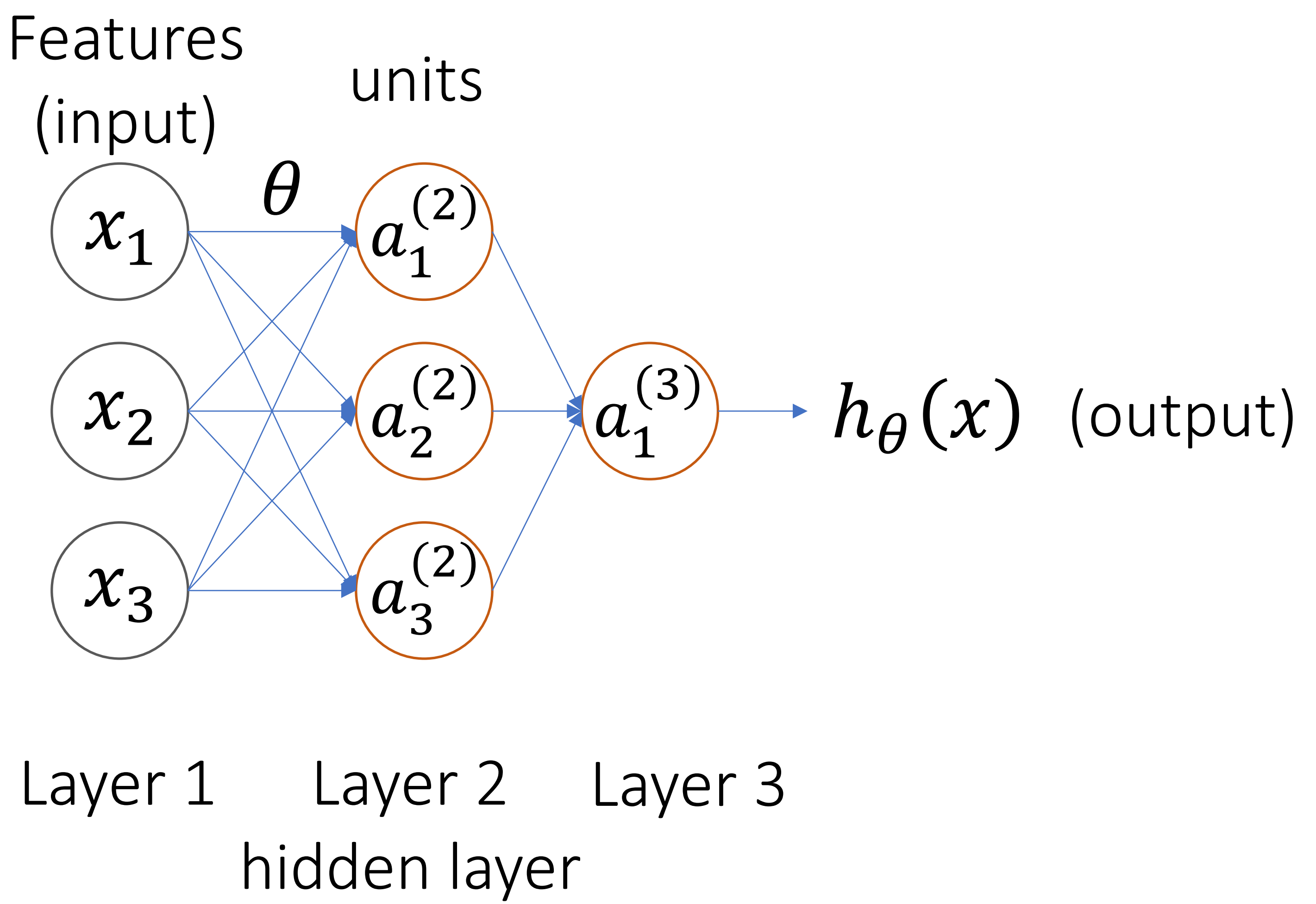
$[0, 1]$

Activation function:
ReLU (rectified linear activation function)



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

Neural Network



Activation function: softmax

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K.$$

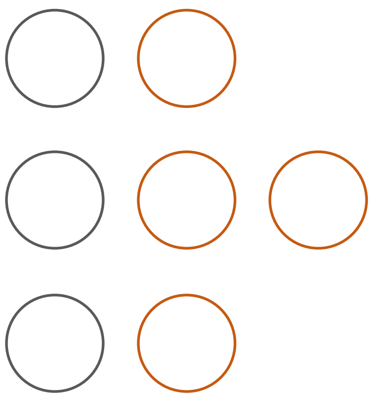
Prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1

After applying softmax, each component will be in (0,1), and all values will add up to 1, so that they can be interpreted as probabilities.

```
softmax <- function(x) {  exp(x) / sum(exp(x)) }
input <- c(0.02, 0.16, 0.10, 0.13, 0.99)
softmax(input)
```

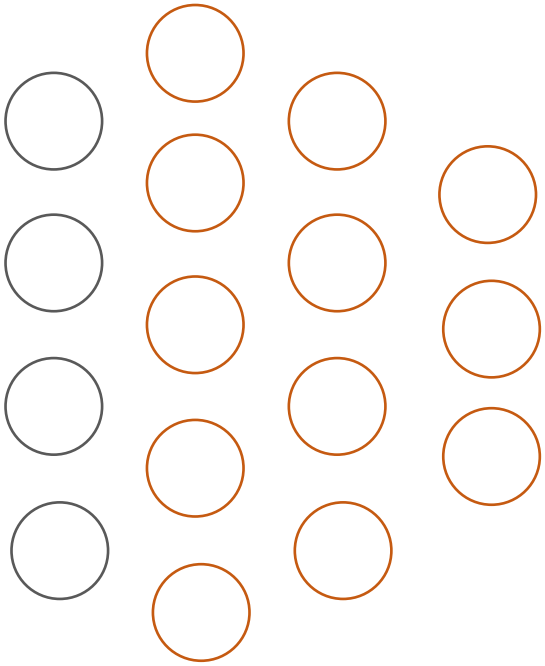
[1] 0.1431069   0.1646121   0.1550259   0.1597471   0.3775080

Neural network architecture



3 layers

2 units in 2^{nd} layer



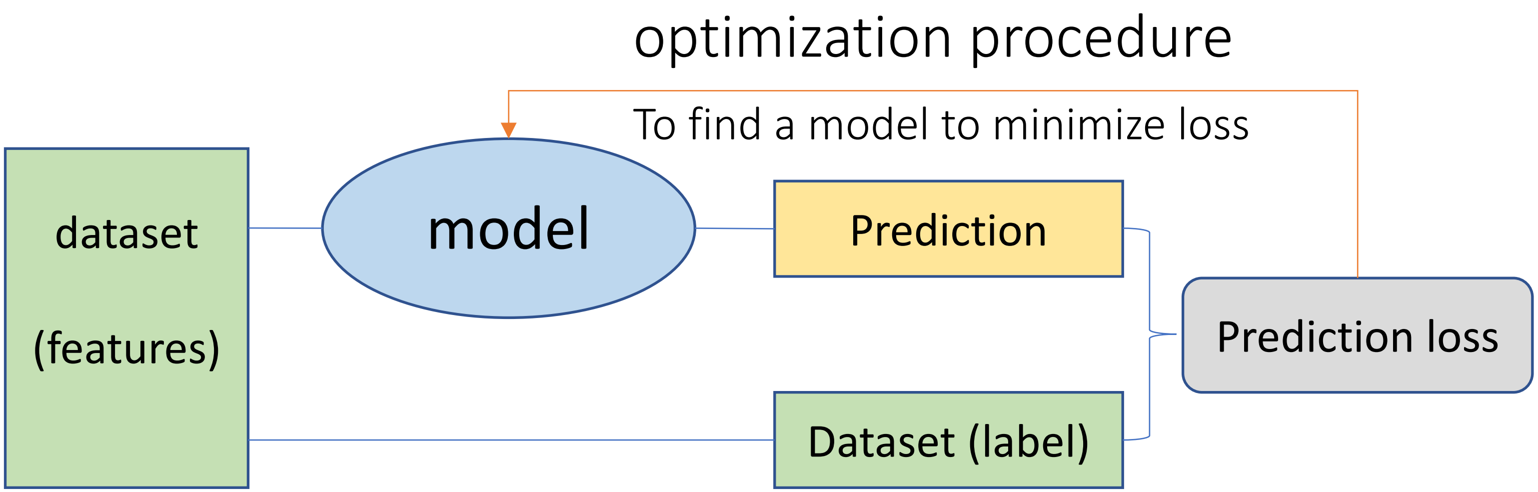
4 layers

5 units in 2^{nd} layer

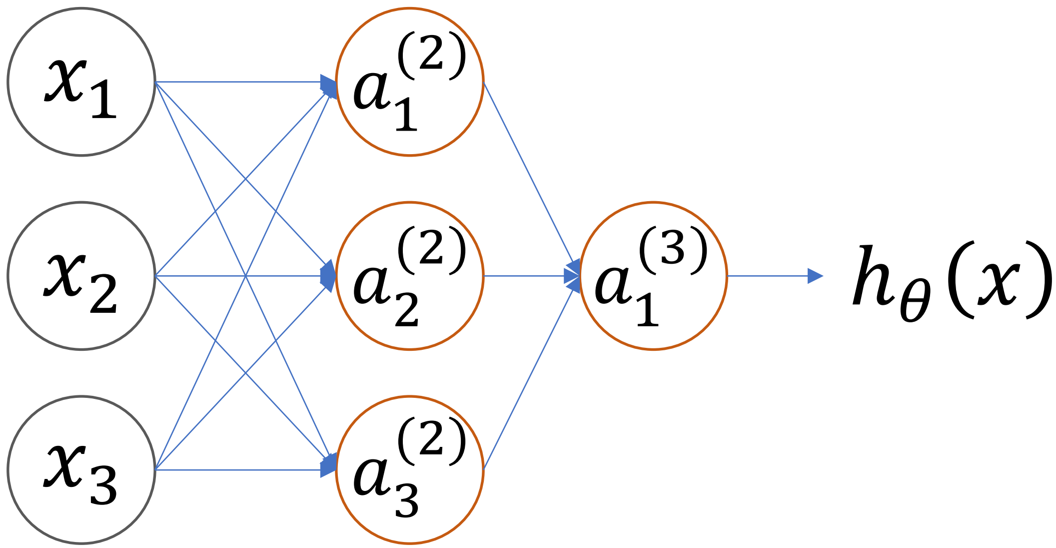
…

training process

- dataset
- cost function $\frac{1}{N}\sum(y - \text{prediction}(x, \theta))^2$
- optimization procedure To know: $\frac{\Delta loss(\theta)}{\Delta\theta}$, partial derivative
- model (parameters)?



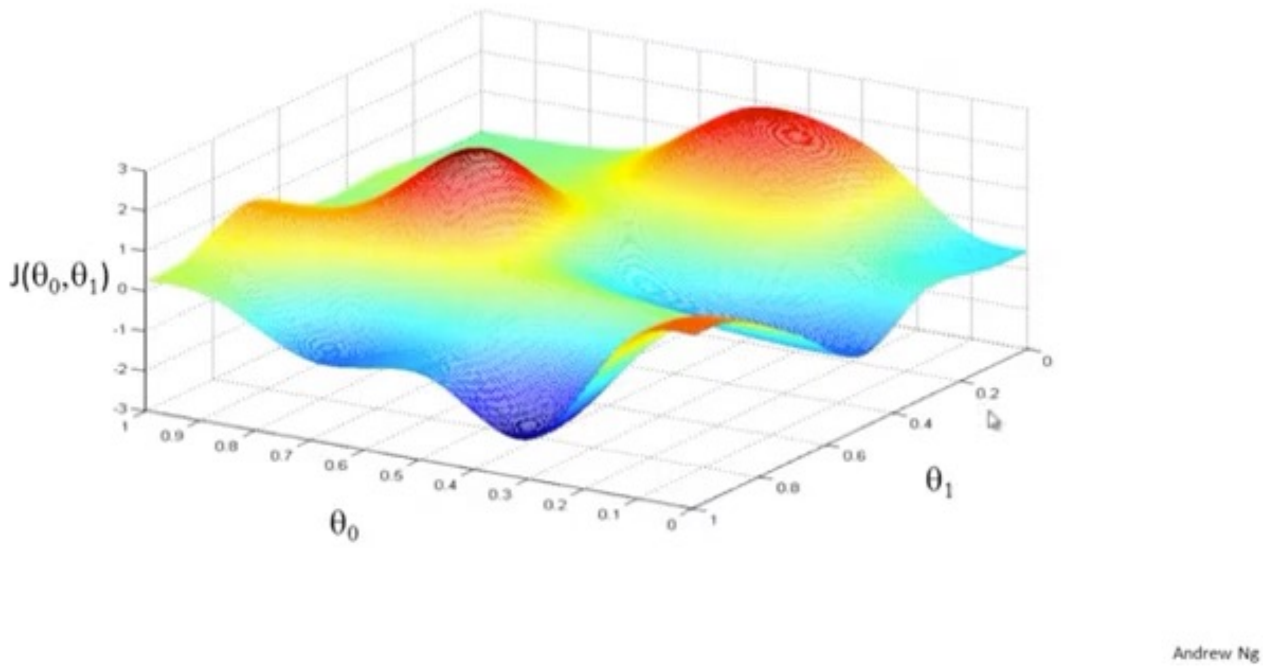
Forward propagation and backpropagation



Forward propagation:
To determine $h_\theta(x)$ with the input data x and parameters θ for each example (entry)

Backpropagation:
To compute "error" of each node (e.g., $a_1^{(3)}$) and then determine <u>*partial derivatives*</u> of loss function (loss slope for each θ or $\frac{\Delta loss(\theta)}{\Delta\theta}$)

Gradient decent to optimize parameters (θ)



To find an optimized model is to identify θ that can minimize the prediction loss;

Therefore, the key is to understand how the loss changes with the change of each θ (partial derivative)

Procedure of neural network (one typical method)

- Randomly initialization of parameters (θ, weights)
- For any input from training data (e.g., (x_i, y_i))
  1. *Forward propagation* to determine $\hat{y}_i$ (prediction(x, θ))
  2. *Backpropagation* to compute the function of *partial derivatives* of loss function
- *Gradient descent* (or other optimization methods) to optimize θ to minimize the loss

* my understanding: backpropagation determines how the change of each θ changes the loss

How can we know if a model trained is a good predictor?

- Training set

\- to learn the model parameters

- Validation set

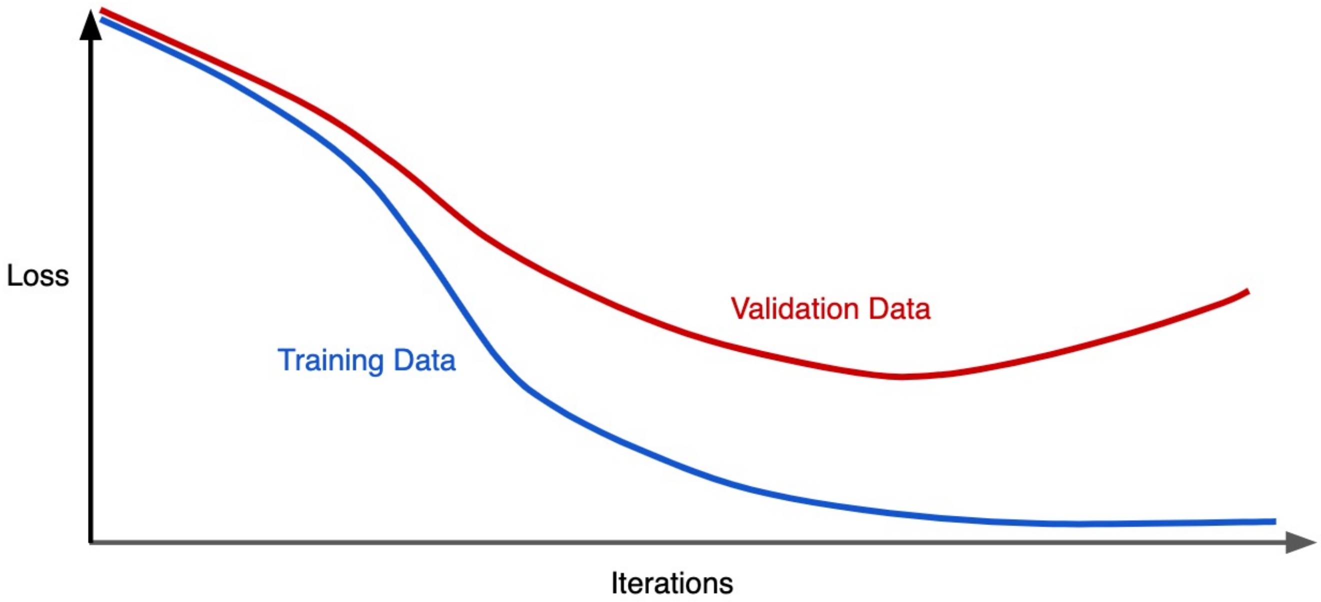
\- to select the best model

- Test set

\- to estimate the generalization

underfitting and overfitting

- *Underfitting* occurs when the model obtains an insufficiently low error value on the training set.
- *Overfitting* occurs when the gap between the training error and test/validation error is too large.



Make the training error small.

Make the gap between training and validation error small.

regularization

Regularization refers to a set of different techniques that lower the complexity of a neural network model during training, and thus prevent the overfitting.

Large weights (θ) in a neural network are a sign of a complex model that possibly overfits the training data.

Original loss function + regularization term

$$\frac{1}{N}\sum(y - \text{prediction}(x,\theta))^2 + \lambda||\theta||^2$$

$$L_2 \text{ regularization term} = \lambda||\theta||^2$$

λ: weight of regulation
$||\theta||^2$: sum of square

Dropout: a simple and effective regularization method

A model is adjusted by dropping out nodes during training. The node is temporarily removed from the network, along with all its incoming and outgoing connections.

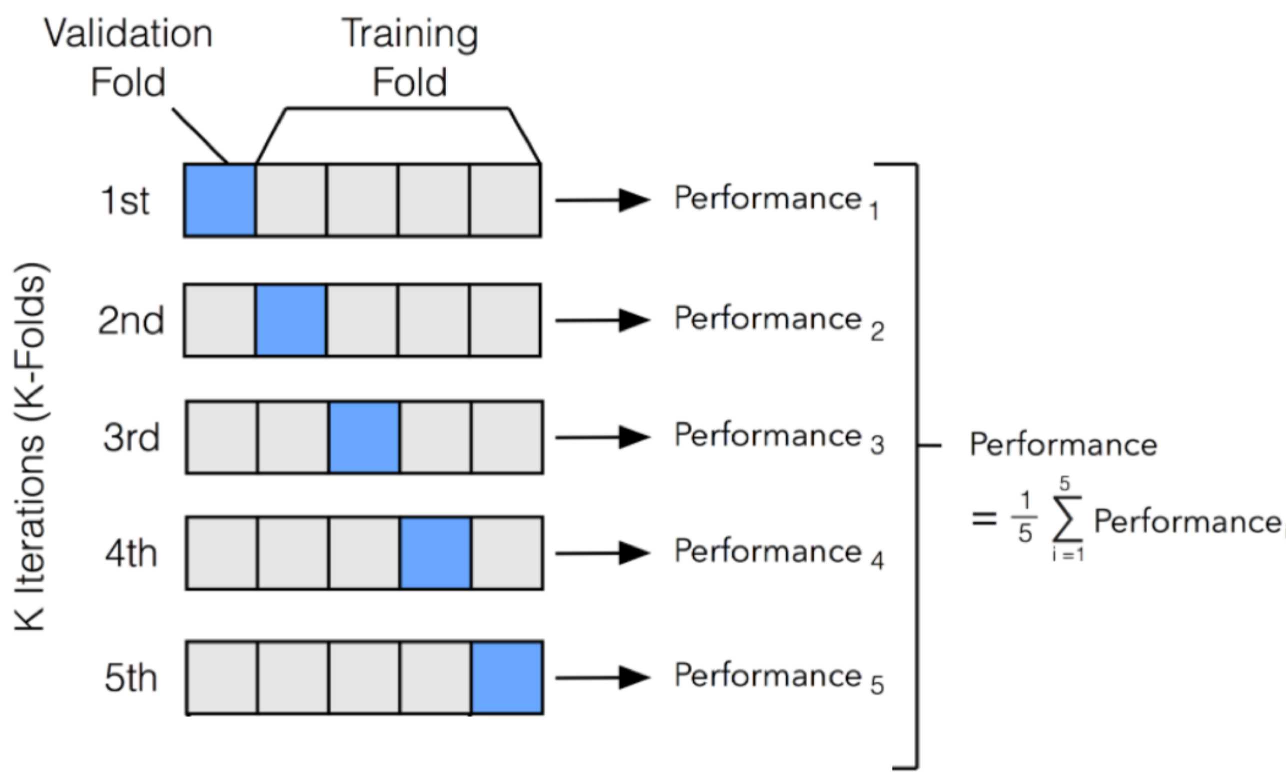
Dropout is a computationally cheap and remarkably effective regularization method to reduce overfitting and improve generalization in deep neural networks of all kinds.

https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

K-fold cross validation

the k-fold cross-validation partitions a dataset into K nonoverlapping subsets. The model performance may then be estimated by taking the average performance across K trials.

5-fold



software

- Tensorflow: an platform developed by the Google Brain team for machine learning.

- Keras: a deep learning software written in Python, running on top of the machine learning platform TensorFlow.

Deep learning in genomics

1. Large training datasets (e.g., thousands of examples), curated to remove confounders, are typically required.
2. Most genomic data do not require very deep networks.
3. Researchers must be wary of high accuracy due to data imbalance or bias that makes classification too easy.
4. A good practice is to compare against simpler machine learning models on the same dataset.
5. Deep learning can achieve high accuracy, but the interpretation of results is more challenging than for standard statistical models.

References

- Zou et al., A primer on deep learning in genomics, 2019, Nature Genetics: doi.org/10.1038/s41588-018-0295-5
- Choi et al., Introduction to Machine Learning, Neural Networks, and Deep Learning, Translational Vision Science & Technology, 2020, 9:14. doi:https://doi.org/10.1167/tvst.9.2.14
- Machine learning, Coursera, taught by Andrew Ng
- Youtube: StatQuest
- Google course: https://developers.google.com/machine-learning/crash-course/ml-intro

Architectures for neuron connection

- Feed-forward neural network
- Convolutional neural network (CNN)
- Recurrent neural network (RNN)