Introduction to Machine Learning

Injung Kim
Handong Global University

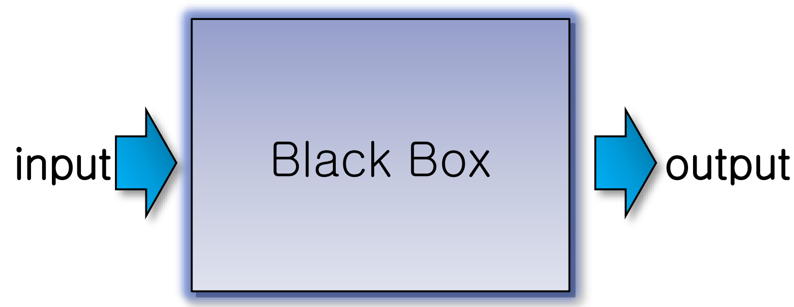
Agenda

- Introduction
- Machine Learning Models
- Types of Machine Learning
- Machine Systems
- k-Nearest Neighbor
- Q&A

Artificial Intelligence

- Artificial intelligence (AI) is the intelligence exhibited by machines or software
- Automation of tasks that require intelligence
  - Recognition, prediction, learning, natural language processing, planning, reasoning, etc.

Strong AI, AGI

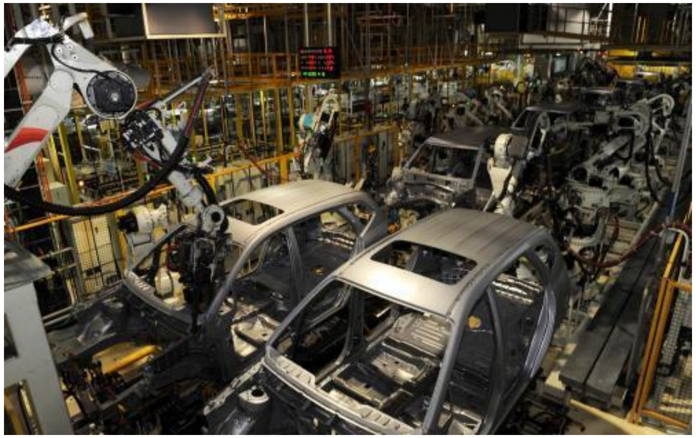


Week AI, Narrow AI

AI vs. Conventional SW

- Conventional SW
  - Perform tasks by following predefined algorithm

- Artificial intelligence targets
  - Complex tasks hard to solve by a fixed procedure
  - Tasks under changing environment
  - Decisions under uncertainty or ambiguity

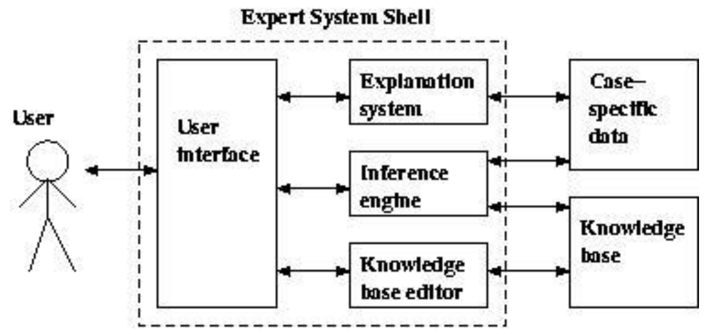


vs.

Two Major Approaches to AI

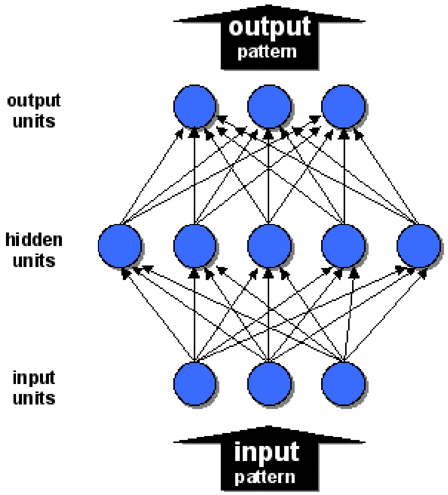
- Knowledge-based approach
  - Rules derived from designer's knowledge
  - Symbolic AI

  Ex) IBM Watson



- Data-driven approach (machine learning)
  - Learn from data
  - Connectionist AI

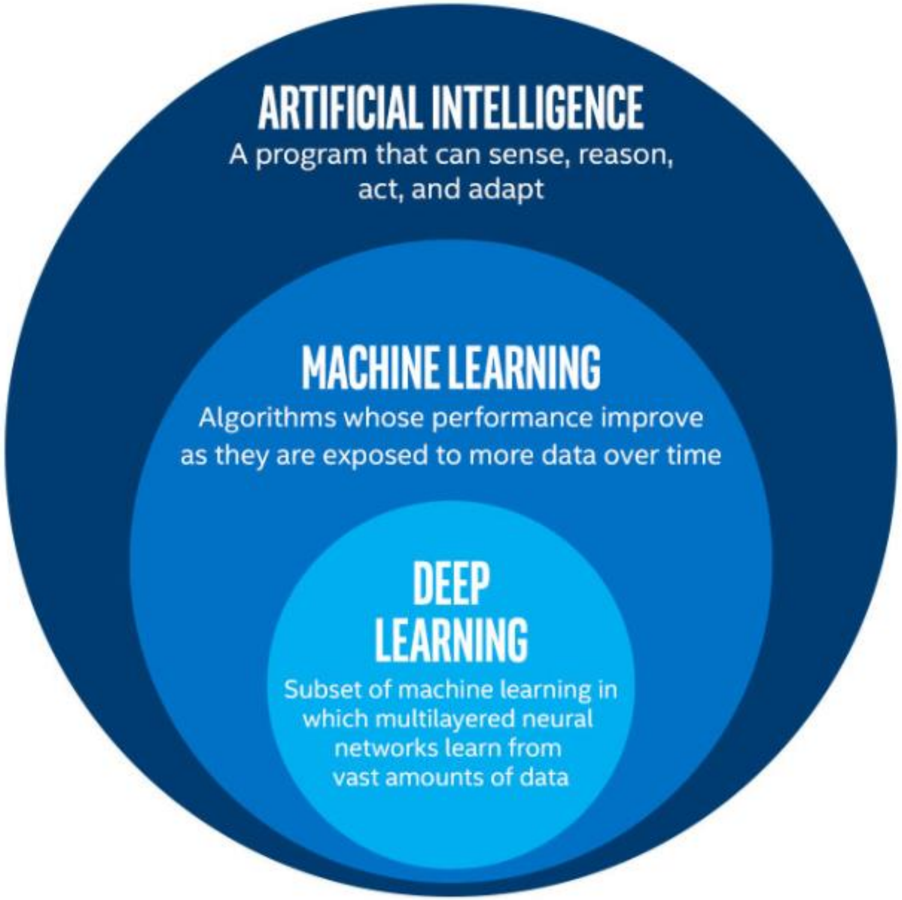
  Ex) AlphaGo



<u>Given sufficient training samples,</u>
data-driven approach can be better than knowledge-based approach

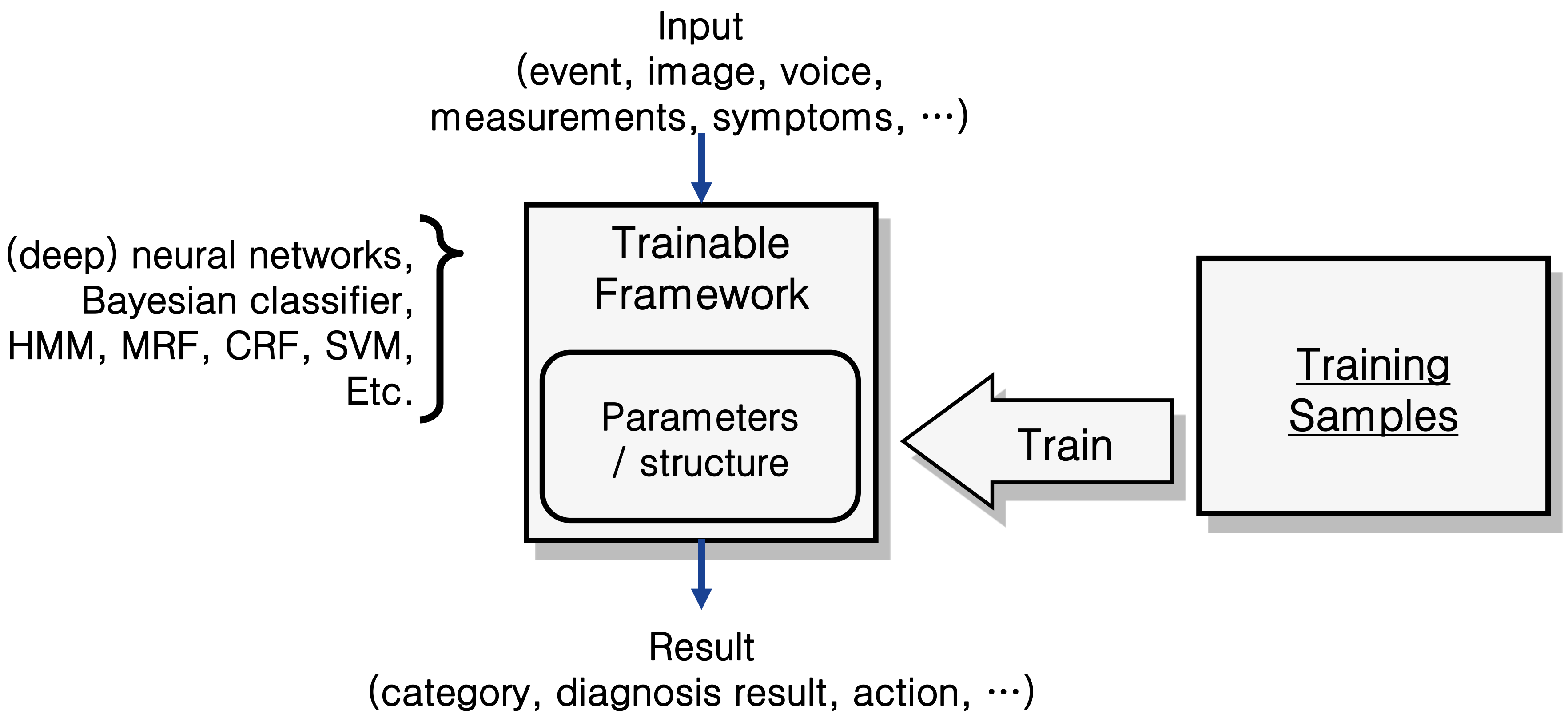
AI, Machine Learning, Deep Learning

- Artificial intelligence: the intelligence exhibited by machines or software
- Machine learning: a field of study that gives computers the ability to learn from data without being explicitly programmed.
- Deep learning: a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data, mostly, based on deep neural networks.



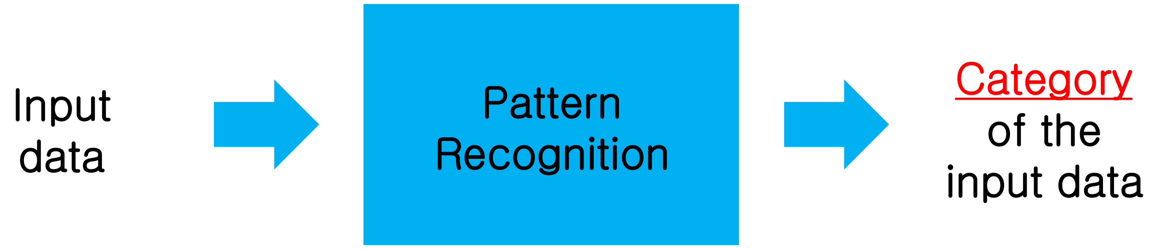
Machine Learning

- Field of study that gives computers the ability to learn without being explicitly programmed.
  - Data-driven approach



Classification

- The act of taking in raw data and taking an action based on the category (or class) of the data.
  - Input: a data (event, object, observation, …) that belongs to one of predefined categories
  - Output: category of the input event



Regression

- Regression: techniques for modeling and analyzing several variables focusing on the relationship between a dependent variable and independent variables (or 'predictors').

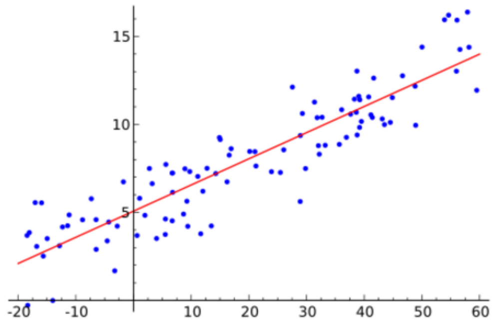
  Ex) Predicting a variable from other variables

  (bankrupt prediction, sales prediction, object coordinate estimation, …)

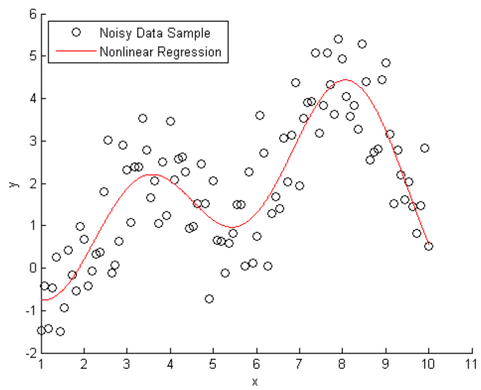
Regression

- Regression model: $\hat{y} \approx F(X;\ \theta)$
  - The form of function F is specified.
    - θ: set of model parameters

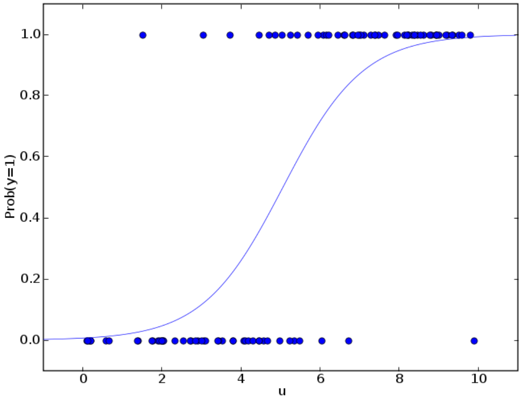
linear regression



nonlinear regression



logistic regression



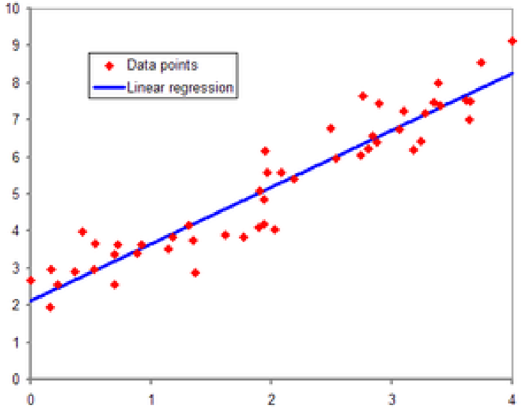
Machine Learning Methodologies

- Linear tasks
  - Linear regression/classification (simple and explainable)
- Nonlinear tasks (simple)
  - Decision tree (explainable)
  - Tree ensemble (Random Forests, GBM, XGBoost)
  - ANN (versatile, scalable)
  - SVM (good generalization)
- Nonlinear tasks (complex)
  - Image: CNN
  - Sequence, time-series data: RNN, CNN, Transformers
- Probabilistic modeling (generation, transform, anomaly detection)
  - Gaussian models, GMM, HMM
  - GAN, VAE, auto-regressive models, flow-based models etc.

Linear Models

- Linear regression

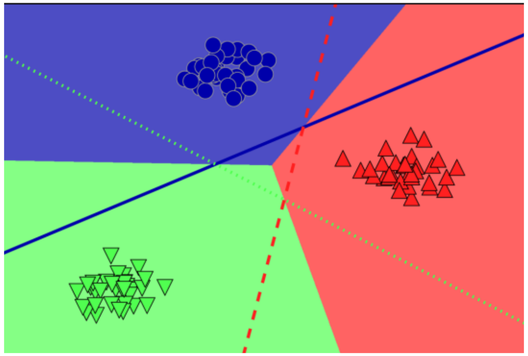
$$\hat{y}_j = w_{1j}x_1 + w_{2j}x_2 + \cdots + b_j = \sum_i w_{ij}x_i + b_j$$



- Linear classification
  - Discriminant function of a class j :

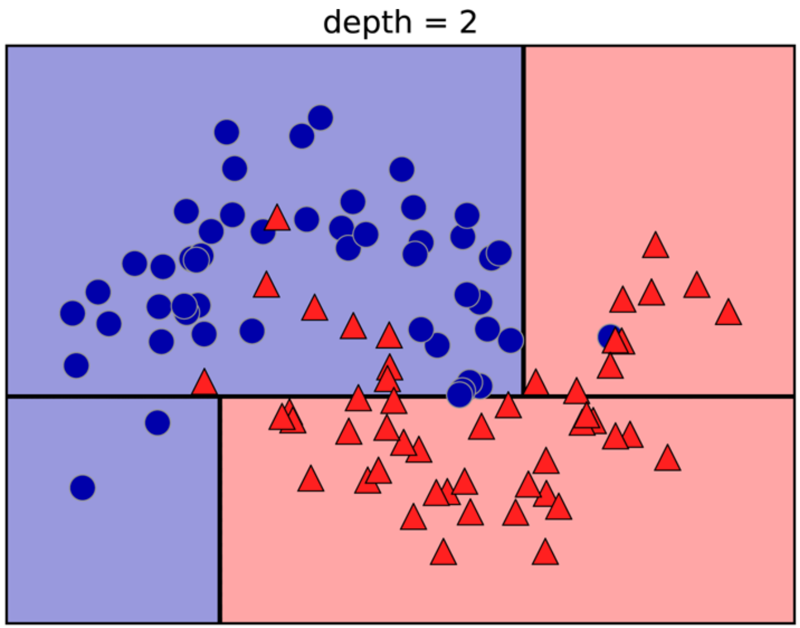
    $f_j(x) = \sum_i w_{ji}x_i + b_j$
  - Decision rule

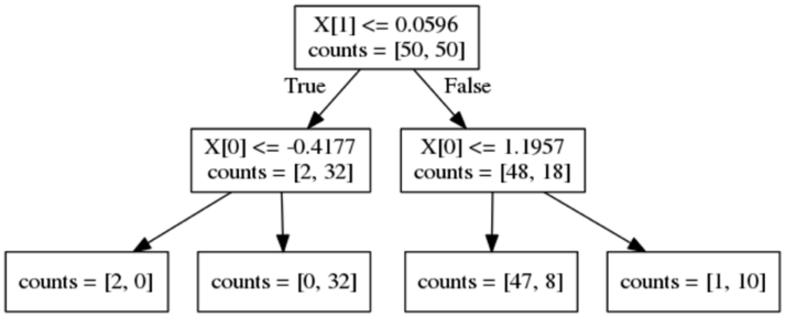
    $\hat{y} = argmax_j[f_j(x)]$



Decision Trees

- Decision tree: a tree structure that represents a hierarchy of if/else questions leading to a decision.
  - If/else question (feature, threshold) splits feature space



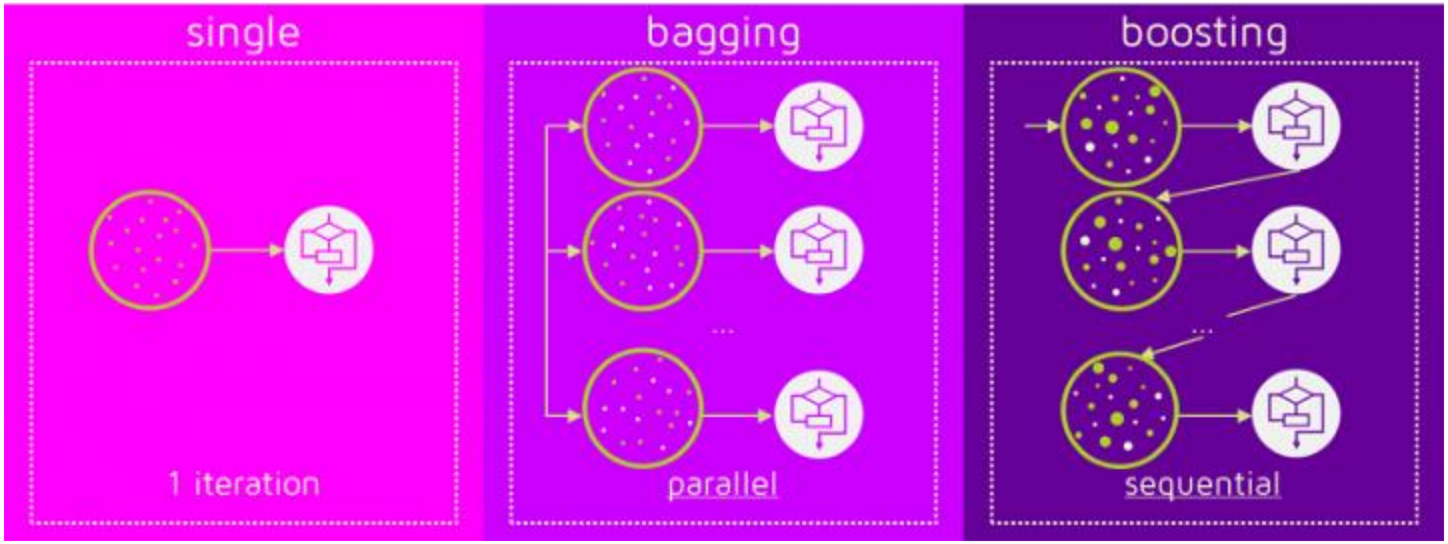


Ensemble

- Bagging: parallel combination
  - Reduces variance

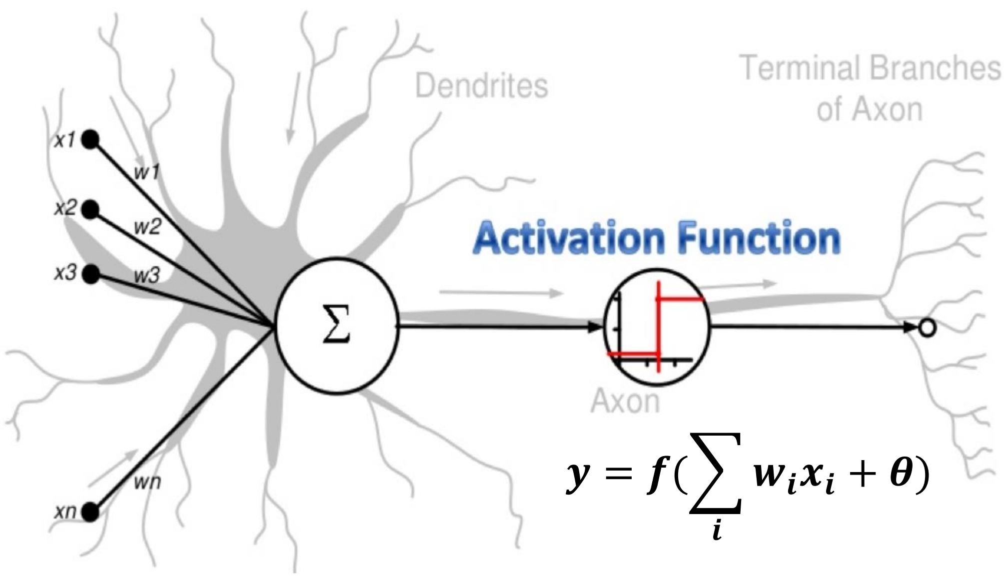
  Ex) Random forest
- Boosting: sequential combination
  - Primarily reduces bias, and also variance

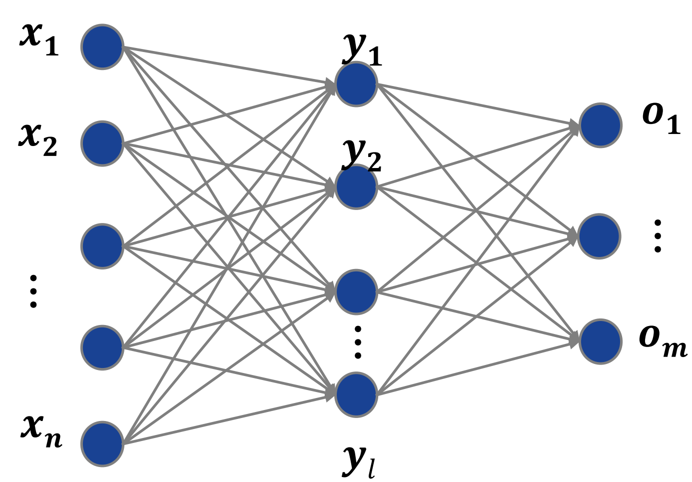
  Ex) AdaBoost, GBM, XGBoost, etc.



Neural Networks

- An artificial neural network is a mathematical model inspired by biological neural networks.
  - Intelligence comes from their connection weights
  - Connection weights are decided by learning or adaptation

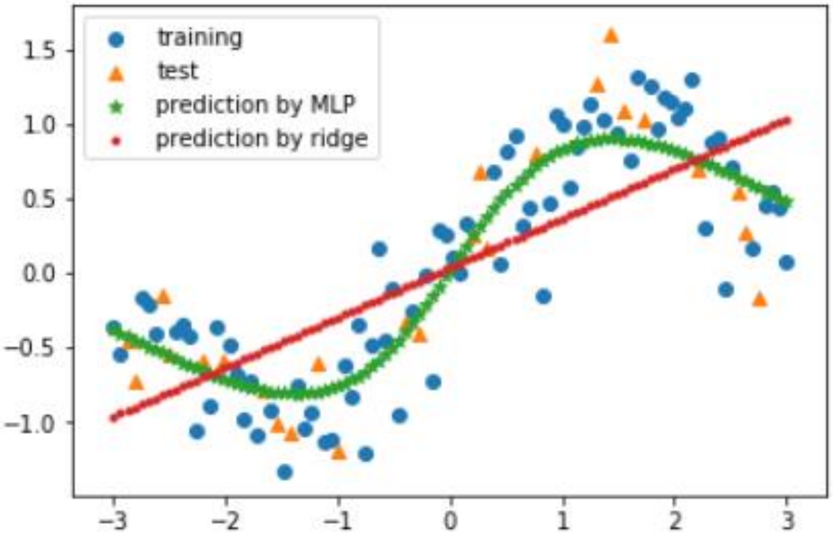




Neural Networks

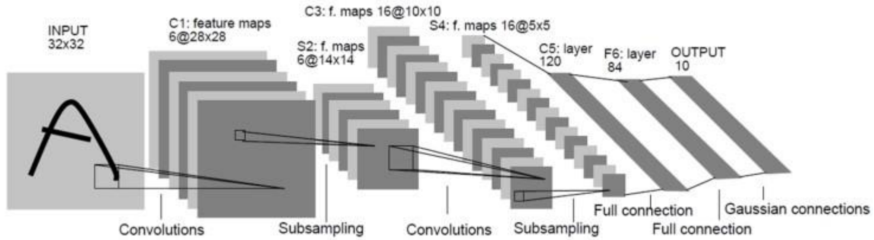
- Universal approximation theorem
  - Neural network with nonlinear hidden layers can approximate any Borel measurable function
  - Can approximate any linear/nonlinear functions

- Applicable to various tasks
  - Classification
  - Regression
  - Generation, transform
  - Anomaly detection

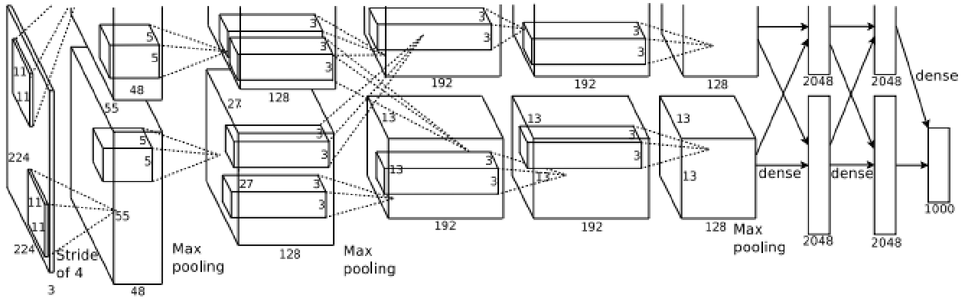


Convolutional Neural Networks

- Convolutional Neural networks (CNN): a class of deep feed-forward network designed to mimic human/animal visual systems
  - LeNet 1998, MNIST, CPU

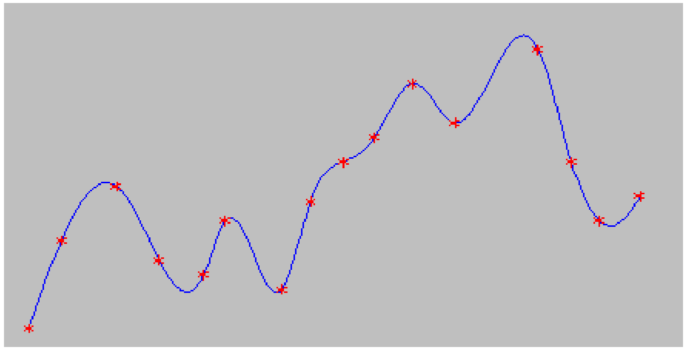


  - AlexNet 2012, ImageNet, GTX 580 x 2

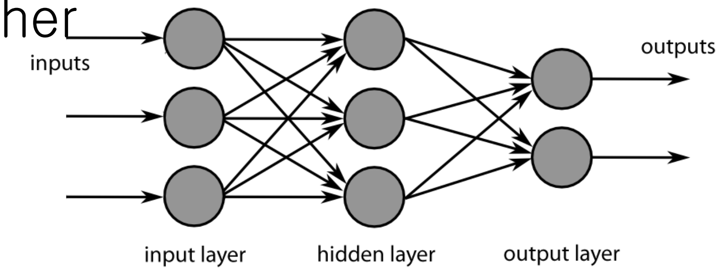


Recurrent Neural Networks

- Artificial neural net designed to analyze sequence or time series data
  - Many real world data are dependent on the previous or next data.

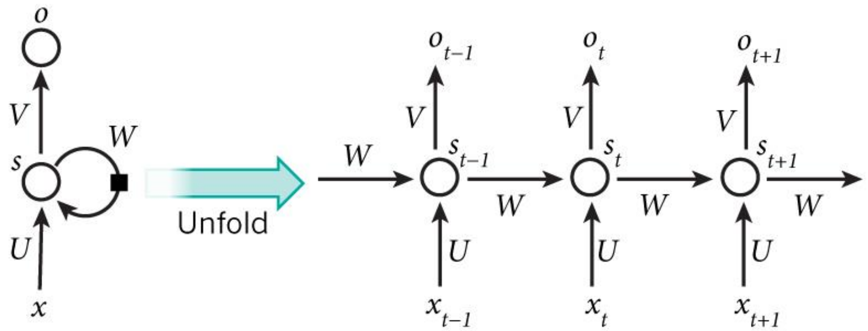


  - Feed forward networks assumes all inputs are independent from each other



Recurrent Neural Networks

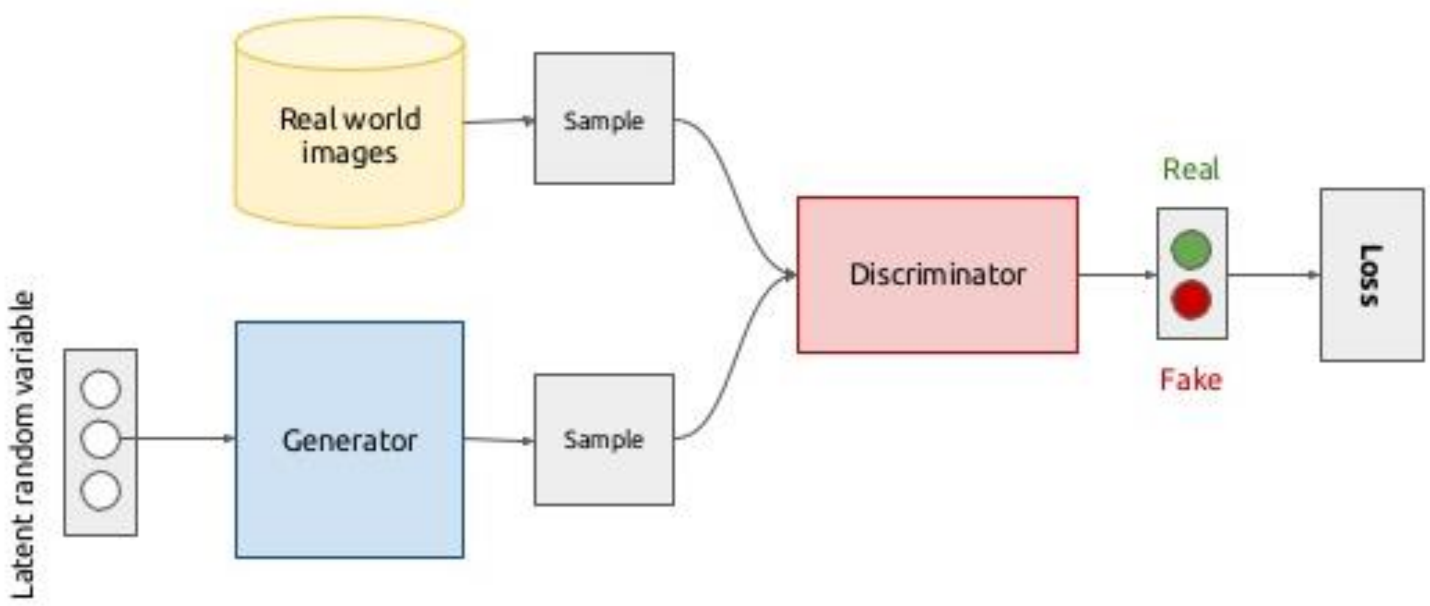
- Recurrent neural network is a neural network specialized for processing a sequence of values $x^{(1)}, x^{(2)}, \ldots, x^{(\tau)}$.
  - Neural networks with recurrent connection
  - State of nodes affect the output and the next state
  - Model for dynamic process
  - Temporarily shared connections



Generative Adversarial Networks (GANs)

- I. Goodfellow, et.al, “Generative Adversarial Nets,” 2014. 6.

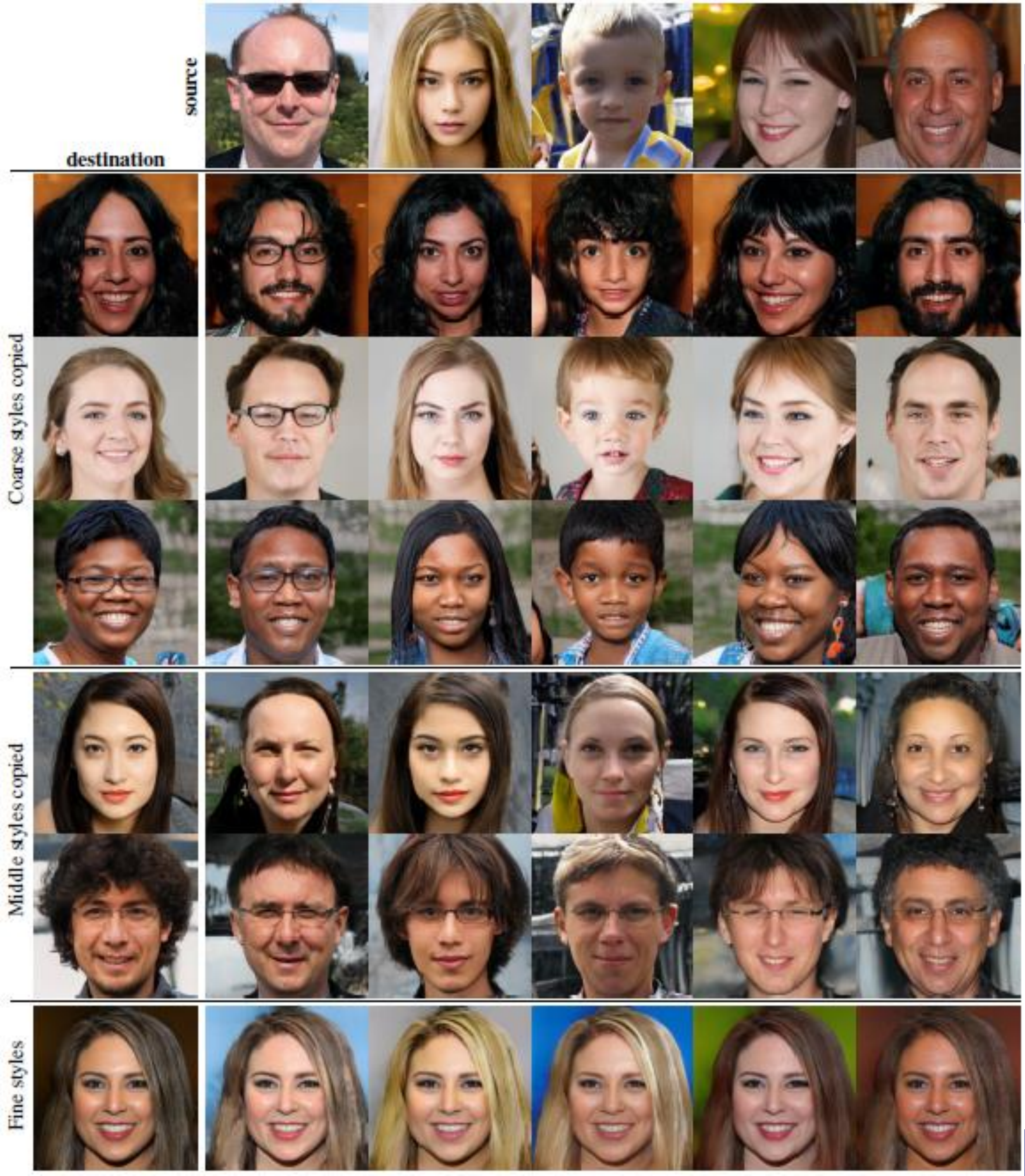
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



BigGAN

- A. Brock, et al., "LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS" 2018.
  - Large-scale GAN training using large batch
  - Truncation trick for random noise generation
    - Trade-off between variety and fidelity)
  - Orthogonal regularization to the generator

Figure 1: Class-conditional samples generated by our model.


source
destination
Coarse styles copied
Middle styles copied
Fine styles

Agenda

- Introduction
- Machine Learning Models
- <u>Types of Machine Learning</u>
- Machine Systems
- k-Nearest Neighbor
- Q&A

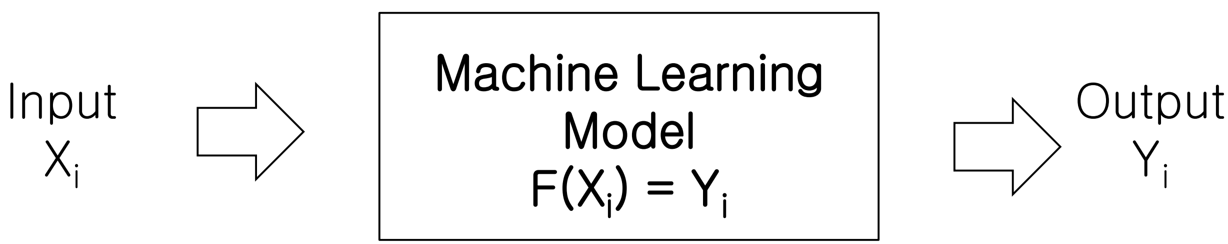
Types of Machine Learning

- Supervised learning (labeled data)
  - Learning **with teacher**
- Unsupervised learning (unlabeled data)
  - Learning **without teacher**
- Semi-supervised learning (partially labeled data)
  - Hybrid of supervised/unsupervised learning
- Reinforcement learning (reward)
  - Learning **from critics**

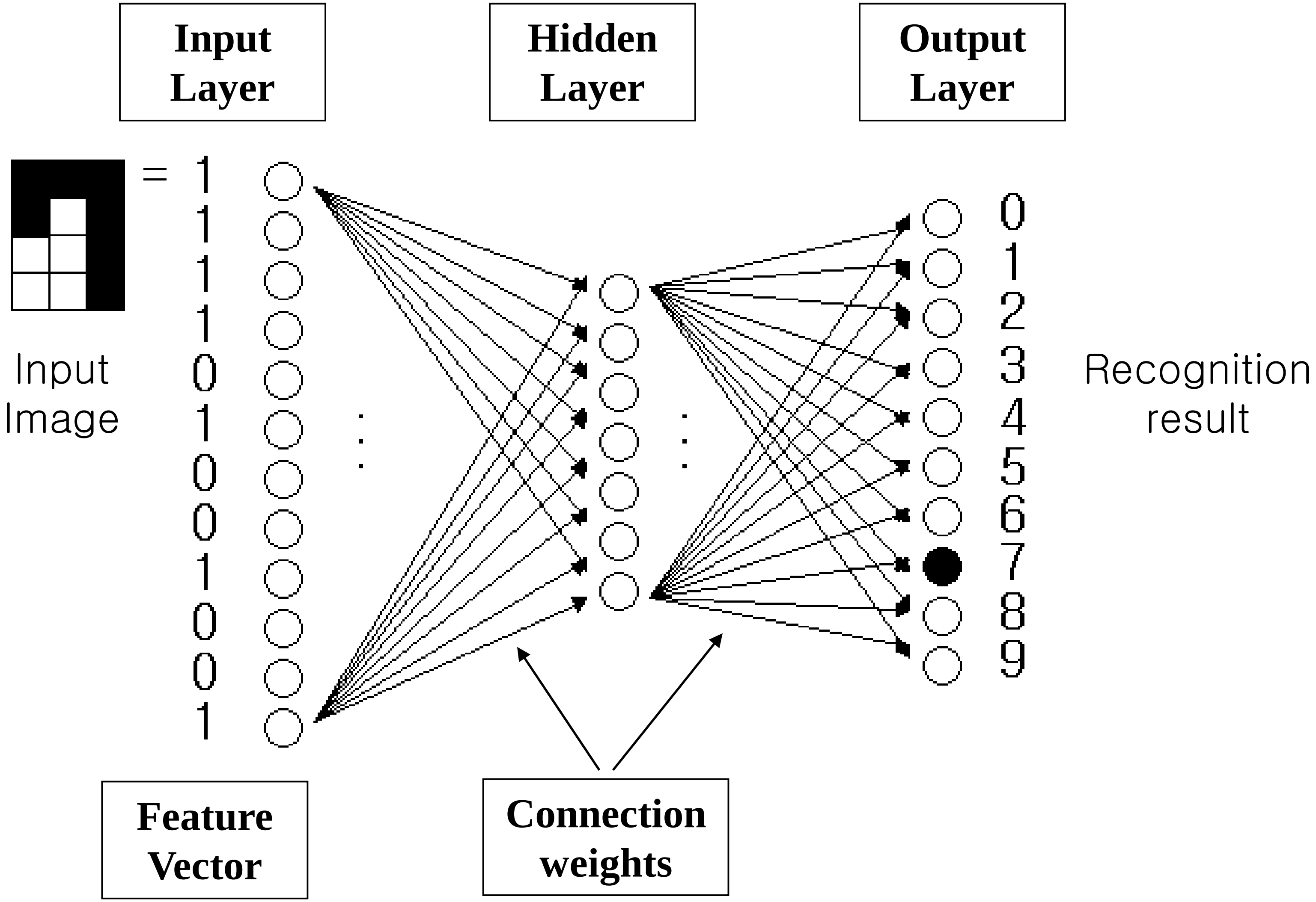
Input X_i ⇨ Machine Learning Model $F(X_i) = Y_i$ ⇨ Output Y_i

Supervised Learning

- **Supervised learning** is the machine learning task of inferring a function from labeled training data.
  - Learn to produce desired outputs for each training samples
  - Given a set of labeled training samples { (X_1, Y_1), (X_2, Y_2), (X_3, Y_3) ⋯ }, learn $F(X_i) = Y_i$

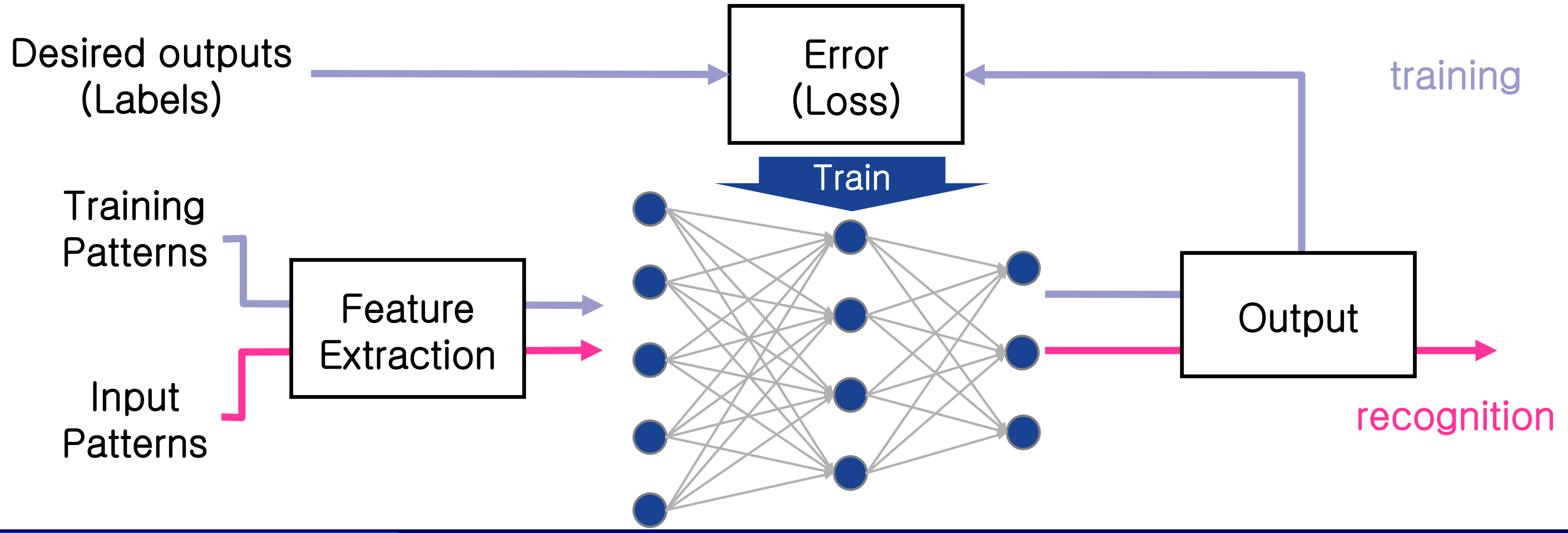


Ex) Digit Recognition using Neural Network

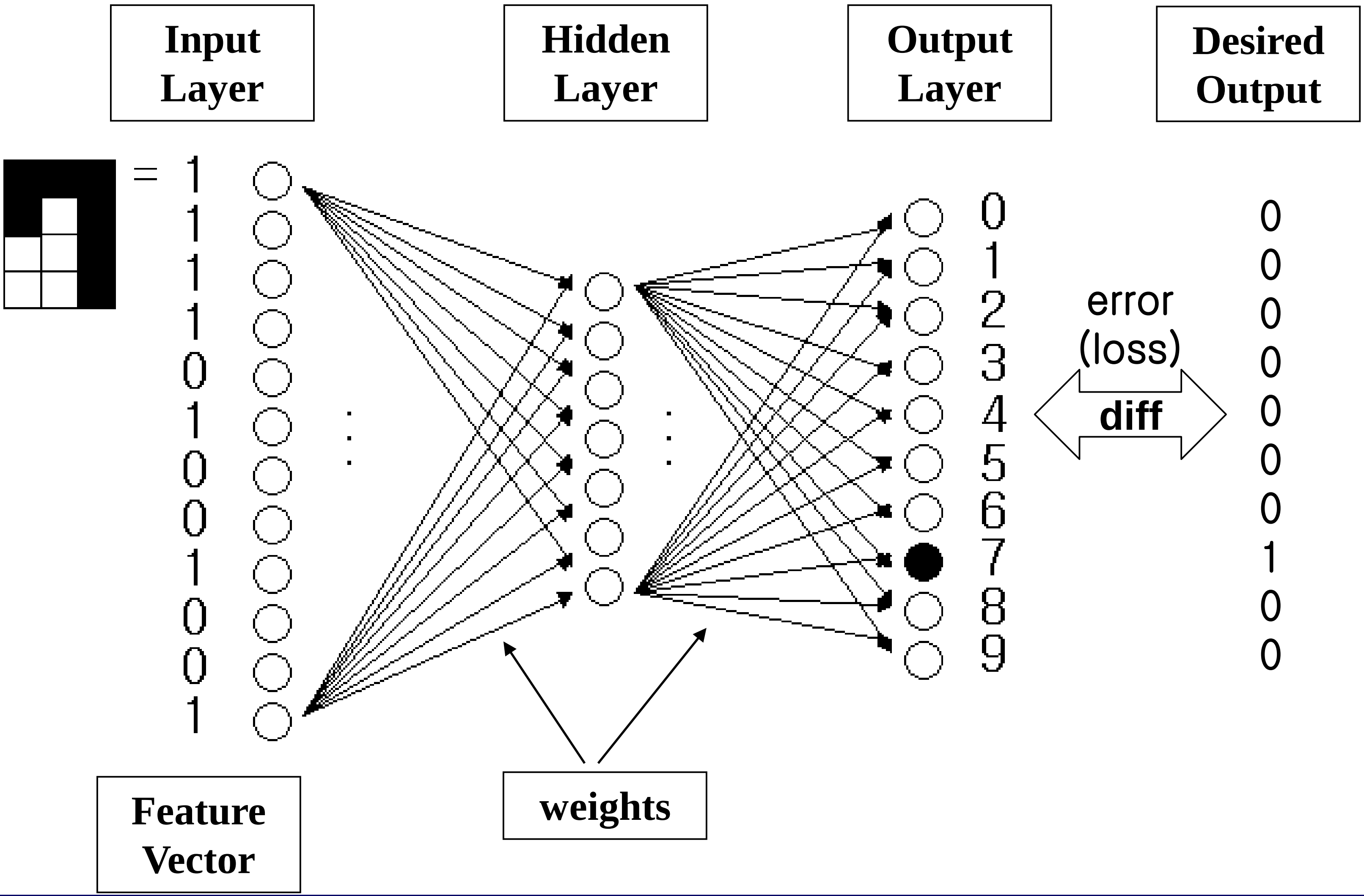


Ex) Building Neural Network Recognizer

1. Design network structure
2. Collect or acquire training samples (with labels)
3. Train connection weights
   - Given training samples and desired outputs, find weights that minimizes error.
4. Apply the trained neural network to target data



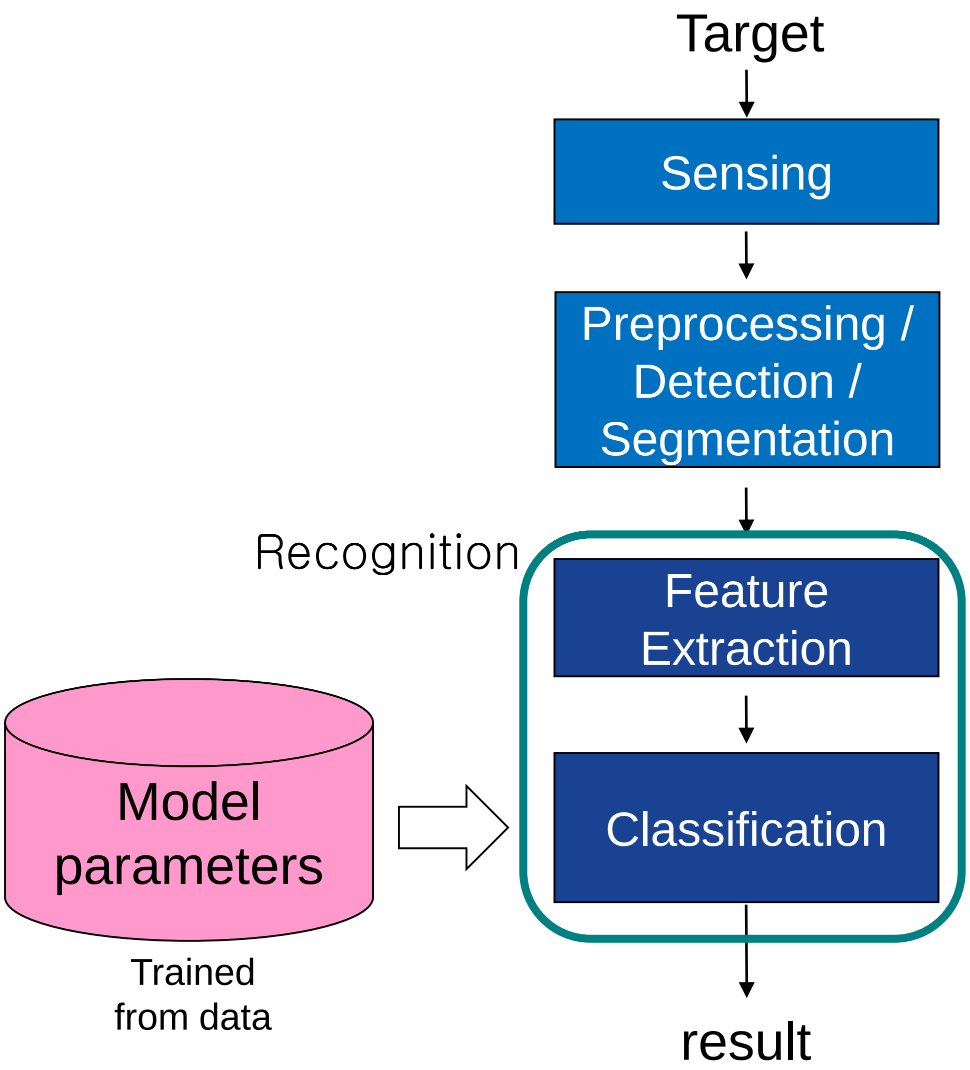
Neural Networks Classifier



Agenda

- Introduction
- Machine Learning Models
- Types of Machine Learning
- <u>Machine Systems</u>
- k-Nearest Neighbor
- Q&A

Recognition System



camera, mic, accelerometer, data glove, GPS, web crawler, …

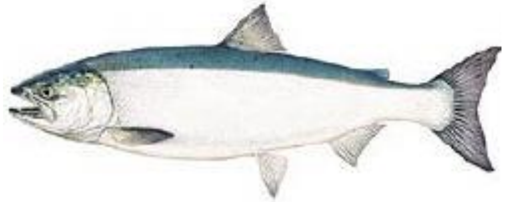
Feature extraction / transform methods

Statistics, machine learning, optimization, …

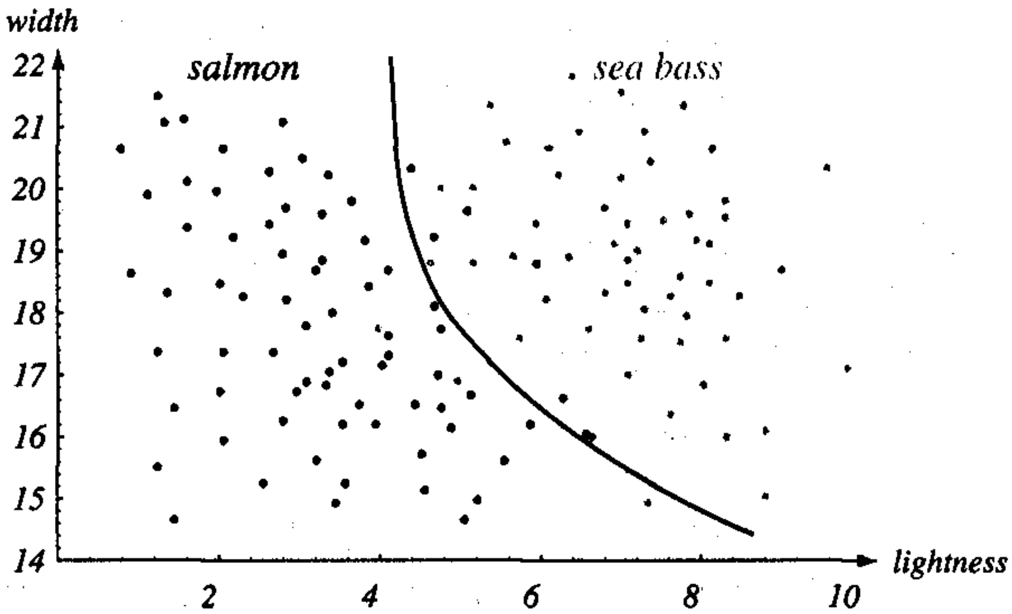
Feature Space

- Representation of fish: (lightness, width)

salmon



sea bass



feature space

Why Feature Vector?

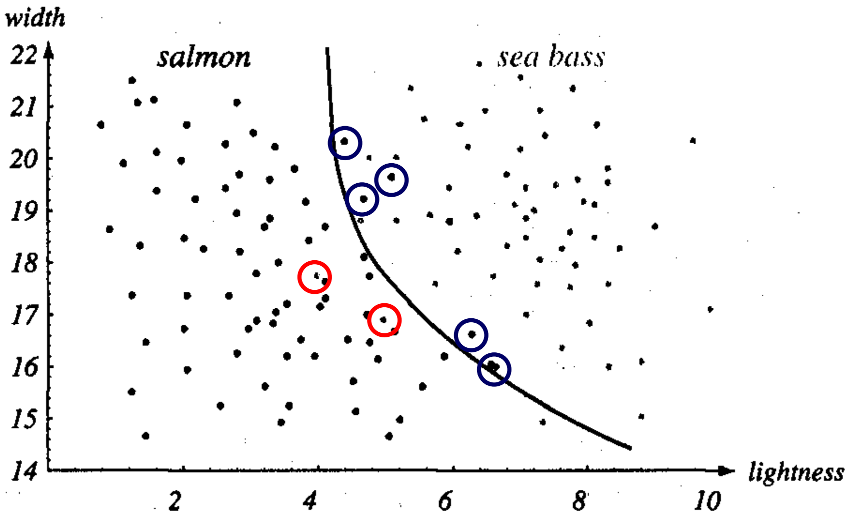
- Representation of data or concept
  - Most machine learning models takes vectors (or sequences) as input
  - Focus on relevant information while ignoring irrelevant information
  - Desirable properties
  - Efficient

Good Representation

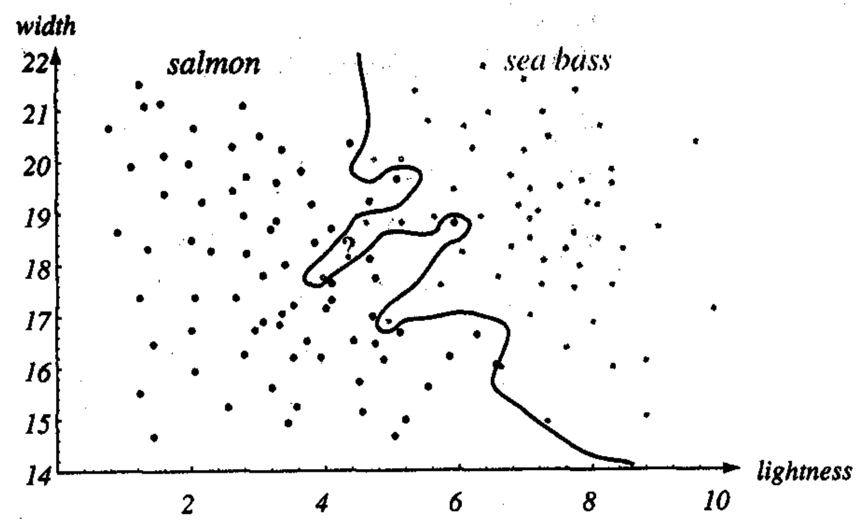
- Inclusive of rich information
- Independence (no redundancy)
- Accessibility (disentangling knowledge)
- Efficiency, sparsity, etc.
- Meaningful distance/similarity measure

Complexity and Generalization Ability

- Complex classifier requires more training data
  - Otherwise, suffer from overfitting
- Simple classifier has advantage on unknown patterns



Simple model



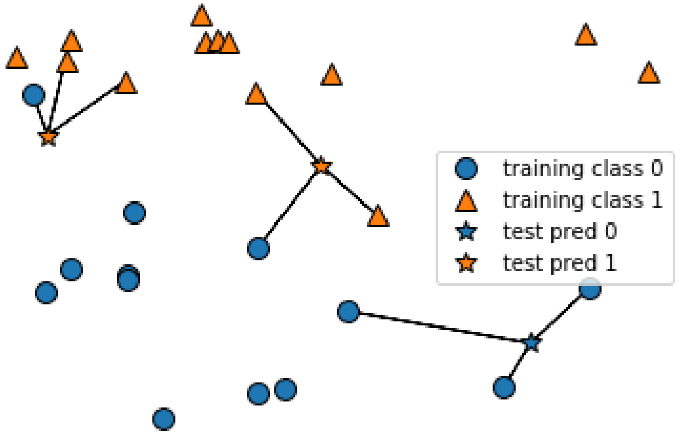
Too complex model

Agenda

- Introduction
- Machine Learning Models
- Types of Machine Learning
- Machine Systems
- **k-Nearest Neighbor**
- Q&A

k-Nearest Neighbor Algorithm

- A non-parametric method used for classification and regression.
  - The input consists of the ***k*** **closest training examples** in the feature space.
  - **Classification**: the input object is classified by a plurality vote of its neighbors
  - **Regression**: the output is the average of the values of k nearest neighbors.

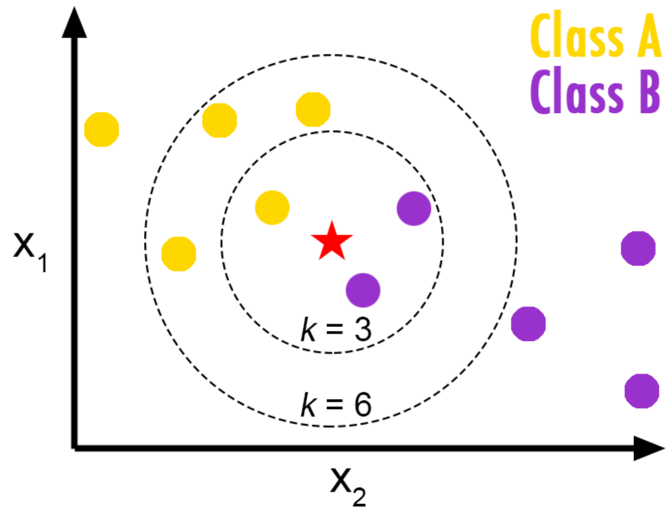


K-NN Classifier

- Approximate $P(\omega_j|X)$ by distribution of classes around X

$$P(\omega_j|X) \approx \frac{N_j}{\sum_j N_j} = \frac{N_j}{k}$$

  - ω_j: classes, N_j: # of class-j samples among neighbors
  - k controls the size of neighborhood (typically, 1, 3 or 5)



k-Nearest Neighbor Algorithm

- Upper-bound of error rate

$$R^* \leq R_{k\mathrm{NN}} \leq R^* \left(2 - \frac{MR^*}{M-1}\right)$$

  - R^*: Bayesian error rate (optimum)
  - M: # of classes

- Limitations
  - Relies on distance metric
  - Not applicable to a large volume of data

- Combined with metric learning
  - Deep learning + k-NN ➔ few-shot learning

k-NN using scikit-learn

- Import packages

```
import numpy as np
import sklearn as sk
```

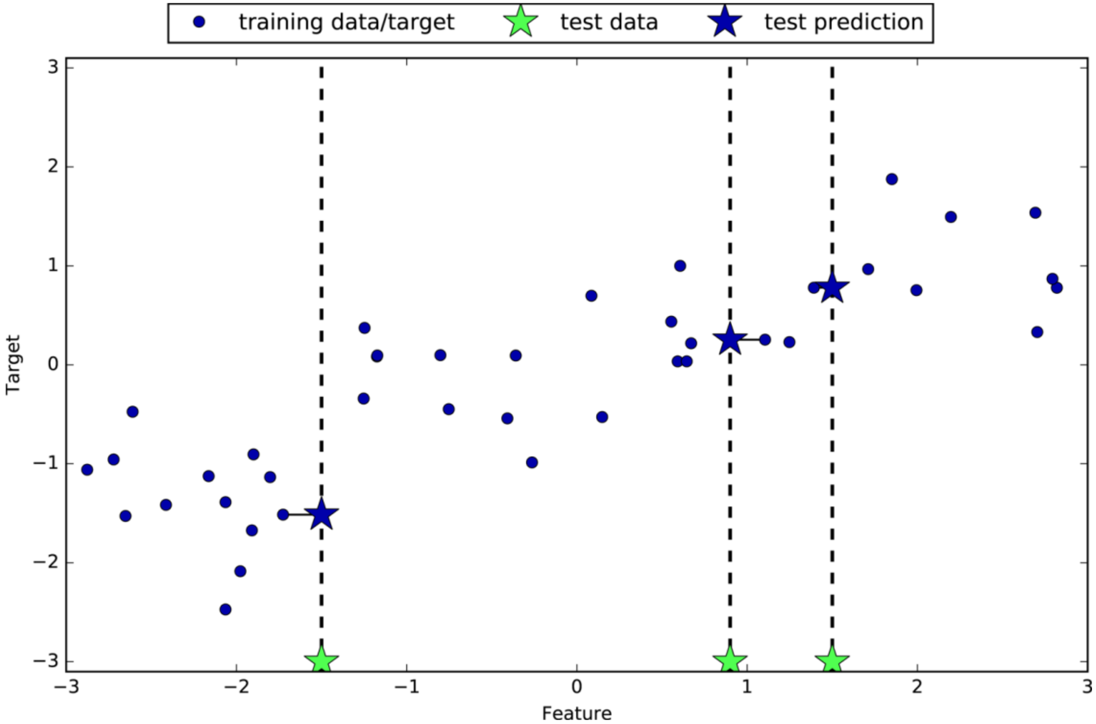
- Load dataset

```
from sklearn.datasets import load_iris
iris_dataset = load_iris()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
iris_dataset.data., iris_dataset.target, random_state=0)
```

k-Nearest Neighbor Regression

- k-NN regression
  - Find k-nearest neighbors
  - Prediction = average value of the neighbors



K-NN using scikit-learn

- Create and train kNN classifier

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
```

- Apply to new samples

```
X_new = np.array([[5,2.9,1,0.2]])    # 2D array
prediction = knn.predict(X_new)
print("prediction = {} ({})".format(prediction.squeeze(),
                    iris_dataset.target_names[prediction].squeeze()))
```

- Evaluate
  - print("Accuracy = {}".format(knn.score(X_test, y_test)))

Numpy Functions: reshape()

- Creating a 1D array
  - a = np.arange(0, 20)
  - print("a.shape = ", a.shape)

    a.shape = (20,)
  - print(a)

    [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

- reshape()
  - b = np.reshape(a, (4, 5))
  - print("b.shape = ", b.shape)

    b.shape = (4, 5)
  - print(b)

    [[0 1 2 3 4] [5 6 7 8 9] [10 11 12 13 14] [15 16 17 18 19]]

Numpy Functions: expand_dims()

- expand_dims()
  - parameter axis specifies the new axis

  - # the new dimension becomes axis 0
  - c = np.expand_dims(a, axis = 0)
  - print("c.shape = ", c.shape)
    c.shape = (1, 20)
  - print(c)
    [[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]]

  - # the new dimension becomes axis 1
  - d = np.expand_dims(a, axis = 1)
  - print("d.shape = ", d.shape)
    d.shape = (20, 1)
  - print(d)
    [[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13]
    [14] [15] [16] [17] [18] [19]]

Numpy Functions: squeeze()

- squeeze() removes single dimensional entries
  - print("c.shape = ", c.shape)
    c.shape = (1, 20)
  - f = np.squeeze(c)
  - print("f.shape = ", f.shape)
    f.shape = (20,)
  - print(f)
    [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

  - print("d.shape = ", d.shape)
    d.shape = (20, 1)
  - g = np.squeeze(d)
  - print("g.shape = ", g.shape)
    g.shape = (20,)
  - print(g)
    [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

Q&A

Thank you
for your attention!