# Artificial Intelligence (CS303)

## Lecture 5: Supervised Learning Methods

# Hints for this lecture

- Technically, learning tasks are tackled by optimizing a (partially pre-defined) model with respect to a user-defined evaluation (objective) function.

# Recap

- We mainly use classification problem as an example:
  - Classification is a sufficiently abstract and have numerous applications.
  - It has so far led to many successful applications of machine learning and AI.

- Given a set of class labels and a set of training data, usually represented by a set of features, to achieve a classifier that (ideally) assigns the correct label to any previous unseen data.

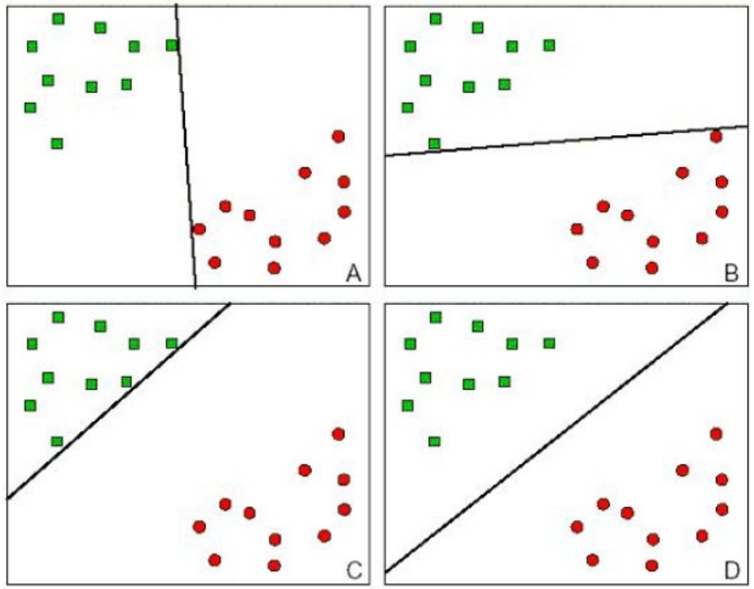
| ID | Height | Weight |
|---|---|---|
| John | 1.75米 | 80KG |
| Mike | 1.8米 | 75KG |

# Outline of this lecture

- **Preliminary Notes**
- **From Linear Regression to Support Vector Machines**
- **Artificial Neural Networks**
- **Decision Trees**

# Linear Discriminant Analysis

- Linear Discriminant Analysis: Viewing each datum to lie in a Euclidean space, find a straight line (a linear function) in the space (recall the example used in the last lecture).
- In linearly separable case, there are more than one optimal hyperplane

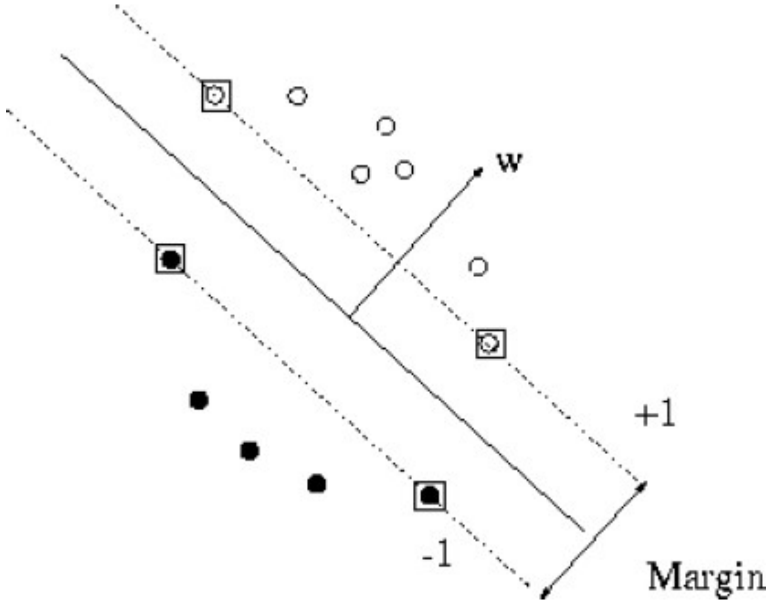



$$g(\boldsymbol{x}) = \boldsymbol{w}^t \boldsymbol{x} + b$$

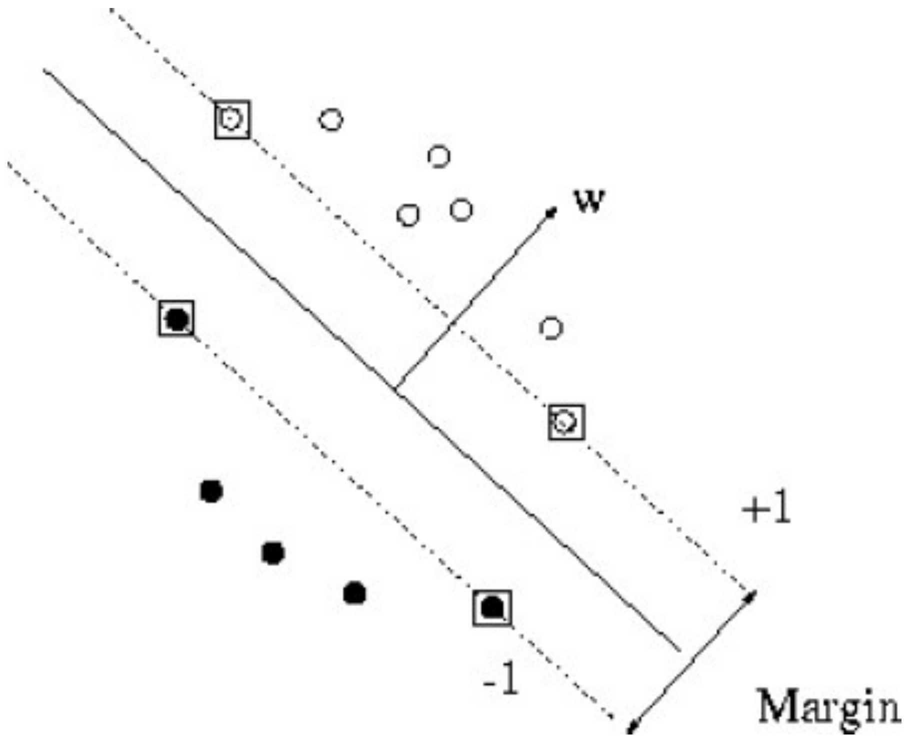
# Support Vector Machines

- **Basic idea: margin maximization**

➢ the **minimum** distance between a data point to the decision boundary is maximized
➢ intuitively, the safest and most robust
➢ **support vectors**: datapoints the margin pushes up



# Support Vector Machines




- **decision boundary**: $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$

- **plus-plane**: hyperplane touching some positive examples, parallel to the decision boundary

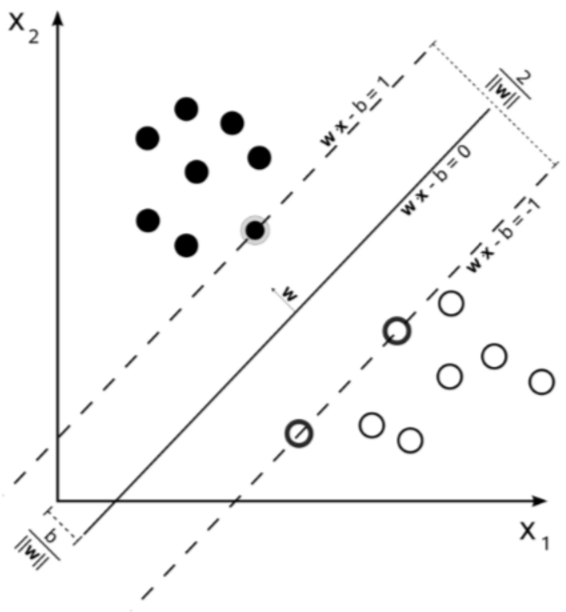
$$\langle \mathbf{w}, \mathbf{x} \rangle + b = c \text{ for some constant } c$$

- **minus-plane**: hyperplane touching some negative examples, taking the form below since **decision boundary** is half way between plus and minus planes:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = -c$$

# Support Vector Machines

- **Basic idea: margin maximization** (margin= $\frac{2}{\|\mathbf{w}\|_2}$)




$$\min \frac{\|\boldsymbol{w}\|^2}{2}$$
$$\text{s.t.} \ \ y_i(w^t x_i + b) \geq 1 \ \ \forall\, i$$

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^t \boldsymbol{x}_j$$
$$\text{subject to} \sum_{i=1}^{n} \alpha_i y_i = 0;\ \alpha_i \geq 0\ \forall i$$

**Dual problem**

# Kernel Trick

- **SVM only involves dot product between vectors, thus kernel function can be used to introduce non-linearlity.**

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^t \boldsymbol{x}_j$$
$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0; \alpha_i \geq 0\ \forall i$$

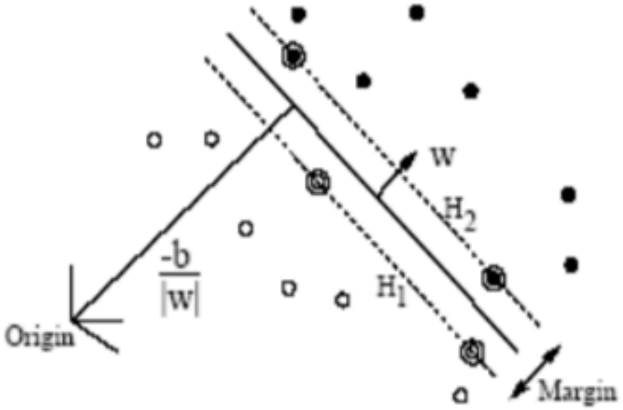
$\Rightarrow$

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_j)$$
$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0; \alpha_i \geq 0\ \forall i$$

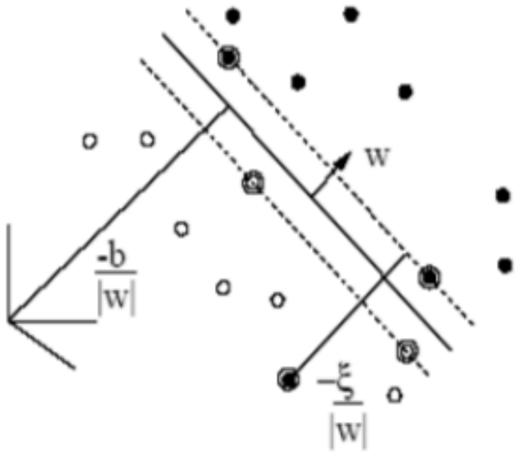
- RBF
$$K(x_i, x_j) = \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_j) = \exp(-\sigma^2 \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2)$$
- Polynomial
$$K(x_i, x_j) = \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_j) = (\boldsymbol{x}_i^t \boldsymbol{x}_j - b)^p$$
- Sigmoid
$$K(x_i, x_j) = \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_j) = \tanh(\boldsymbol{x}_i^t \boldsymbol{x}_j - b)$$

# Soft Margin SVM

- **Even with kernel trick, it is hardly to guarantee that the training data are linearly separable, thus a soft margin rather than hard margin is used in practice.**




Hard Margin（硬间隔）




Soft Margin（软间隔）

$$\min \frac{||w||}{2} + C\sum_i \varepsilon_i$$

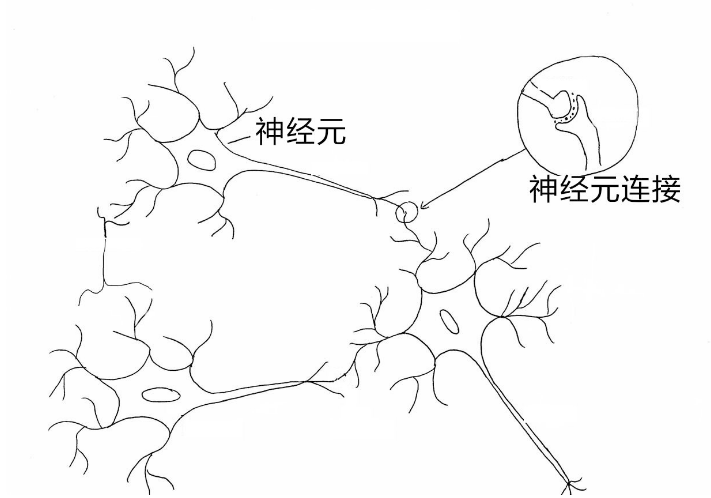
$$s.t.\, y_i(w^t x_i - b) \geq 1 - \varepsilon_i, \varepsilon_i \geq 0, \forall\, i$$

$$\max_{\alpha} \left\{ \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^t \phi(x_j) \right\}$$

$$\text{s.t.} \sum_i \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, \forall i$$

# Artificial Neural Networks

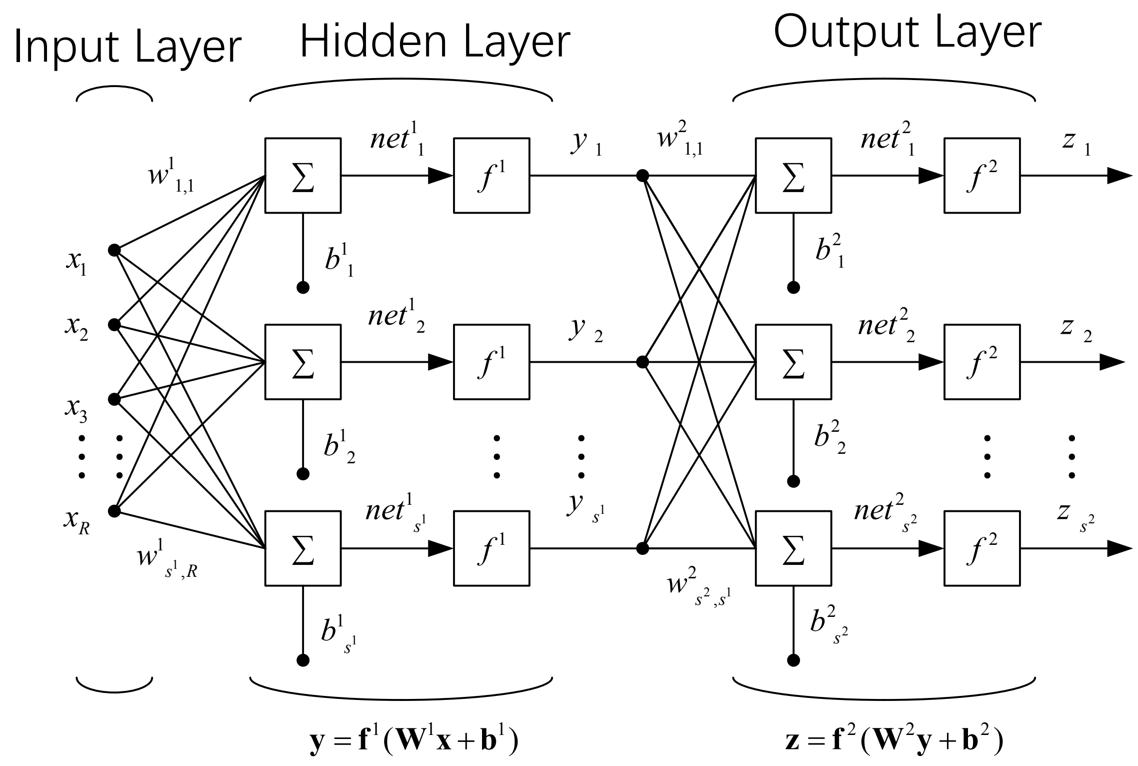
- Inspired by biological neural networks (fancy story).




- In many AI literature, simply referred to neural networks.

# Artificial Neural Networks

- A highly nonlinear function that mimic the structure of biological NN.




Multi-Layer Perceptron

# Training NNs

- Optimize weights to minimize the Loss function

$$J(w) = \frac{1}{2}\sum_{k=1}^{c}(y_k - z_k)^2 = \frac{1}{2}\left|\left|\mathbf{y} - \mathbf{z}\right|\right|^2$$

- Training algorithm: gradient descent, with Back Propagation algorithm as a representative example.

Chain rule

$$\nabla w_{ji} = -\eta \frac{\partial J}{\partial w_{ji}}$$

Update weights between output and hidden layers

$$\frac{\partial J}{\partial w_{ki}} = \frac{\partial J}{\partial net_k}\frac{\partial net_k}{\partial w_{ki}} = -\delta_k \frac{\partial net_k}{\partial w_{ki}}$$

Update weights between input and hidden layers

# Back Propagation Algorithm

**input:** training examples:

$$D = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}, \mathbf{x}_i \in \mathbb{R}^d, \mathbf{y}_i \in \mathbb{R}^l$$

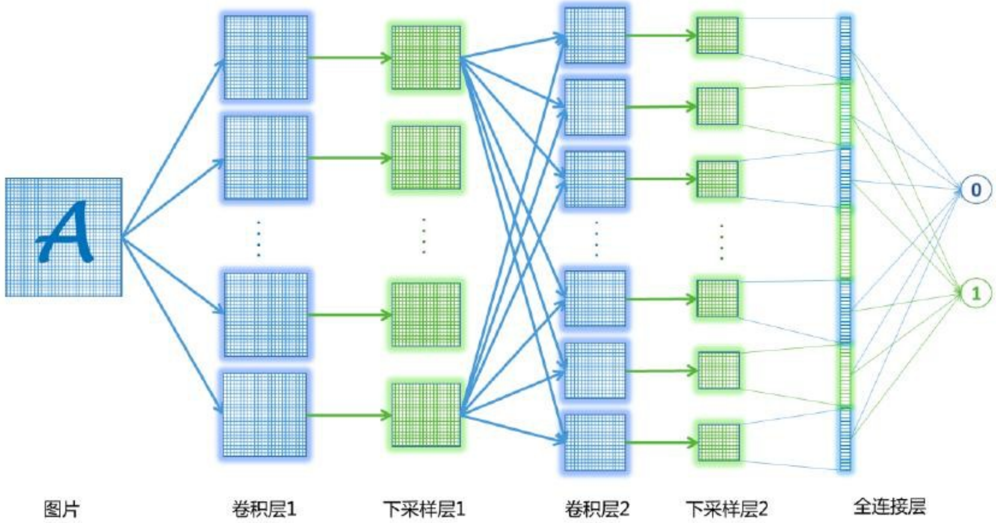
learning rate $\eta$

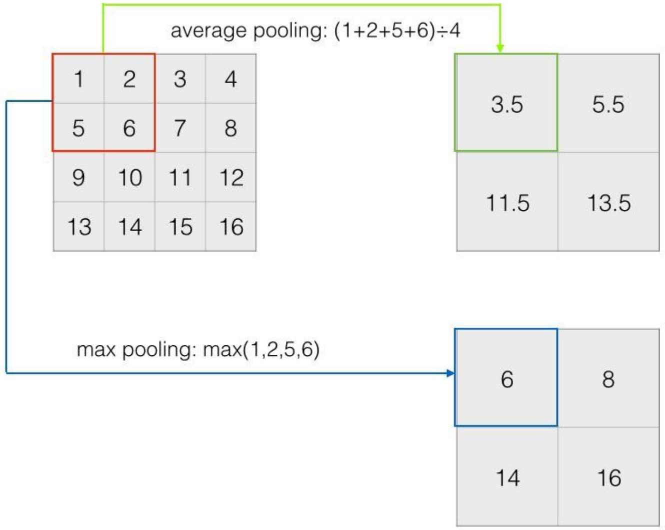
1. **while Optimality conditions are not satisfied do**
2. **for all** $(\mathbf{x}_k, \mathbf{y}_k) \in D$ do
3. calculate current outputs by current parameters
4. calculate gradient for output-layer neurons
5. calculate gradient for hidden-layer neurons
6. update weights and thresholds
7. **end**

# Convolutional Neural Networks

Deep CNN classifier

- Bear 0.01
- Plane 0.00
- Cat 0.82
- Aliens 0.10
- Boat 0.01
- Ball 0.01
- Car 0.01
- Banana 0.01
- …….



# More Noteworthy Issues

- Universal Approximation Theory (Google it by yourself).
- According to UAT, in ideal case, one hidden layer is sufficient for most (if not all) problem.
- Fully connected NN (MLP) with more than 1 hidden layer is very difficult to train.
- CNN is basically a manually designed method for introducing sparsity into NN, such that it can be trained more easily.

# Nonmetric Data

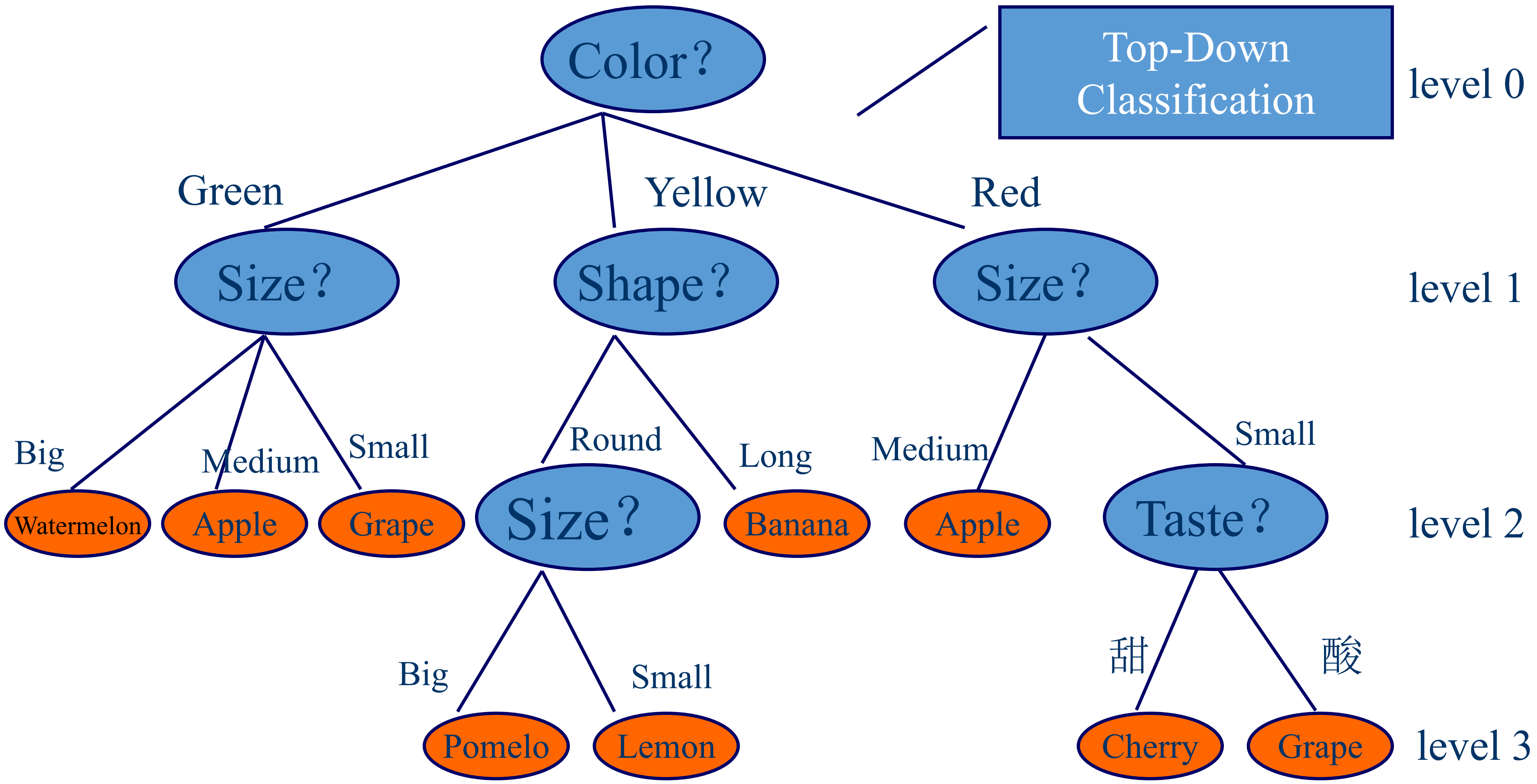
- In many cases, data are not represented as numerical values, e.g.,
  - Color, taste of fruits
  - DNA sequences (AGCTTCAGATTCCA)
  - Gender, Nationality, place of birth

- Such data can be treated as numerical values to make methods such as SVM and ANN applicable, but the best way to represent (encode) them is usually unknown, and it is hard to interpret the classification results.

# Decision Trees

- A natural way to handle nonmetric data (also applicable to real-valued data).
- Analogous to a search tree, but each leaf node corresponds to a class label, and the root node and each intermediate node corresponds to a decision rule.
- For a given datum/instance/example, the classification process starts from the root node and eventually arrives a leaf node.
- Potential advantages: efficient (a number of simple queries) and interpretable.

# Decision Trees



# How to construct a DT?

- Objective function: Impurity of the leaf nodes, i.e., how much training data from different classes would fall into the same leaf node?

- There are lots of choices to measure the impurity:

**Entropy** $$i(N) = -\sum_i p(\omega_i) \log_2 p(\omega_i)$$

**Variance** $$i(N) = \sum_{i \neq j} p(\omega_i) p(\omega_j) = 1 - \sum_i p^2(\omega_i)$$

**Misclassification rate** $$i(N) = 1 - \max_i p(\omega_i)$$

# How to construct a DT?

**Training methods: Heuristic Search (Tree Search)**

1. Start from the root, keep searching for a rule to branch a node.

2. At each node, select the rule that leads to the most significant decrease in impurity (similar to gradient descent).

$$\Delta i(N) = i(N) - p_L i(N_L) - (1 - p_L) i(N_R)$$

3. When the process terminates, assign class label to the leaf nodes. A common practice is to label a leaf node with the label of majority instances that fall into it.

# Complexity Control

- The construction process will not terminate automatically unless each leaf node only contains instances from 1 class, possibly only 1 instances – overfitting.

- Thus complexity of the tree needs to be controlled, approaches include:
  - Setting the maximum height of the tree (early stopping)
  - Introduce the tree height (or any other complexity measure as a penalty)
  - Fully grow the tree first, and then prune it (post processing)

# Summary

- Learning approaches for 3 types of model representation.
  - SVM – Quadratic Programming
  - NN – Gradient Descent (BP)
  - DT – Heuristic Search (A specialized version)

To be continued