Artificial Intelligence (CS303)

Lecture 11: Supervised Learning I

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.**

Outline of this lecture

- **Preliminary Notes**
- **Bayesian Decision Theory and Parametric methods**
- **Linear Discriminant Analysis and Support Vector Machines**
- **Artificial Neural Networks**
- **Decision Trees**

I. Preliminary Notes

To Learn What?

- **Ideally, the purpose of (Machine) Learning is to achieve a "universal" agent function that can give the appropriate output for any input that we can imaging.**

- **Unfortunately, getting a universal function is impractical (at least for now).**

- **To be realistic, we need to focus on simpler learning tasks, e.g., the ability of recognizing a hand-written digit should be much easier to learn than the ability to make a good living.**

To Learn What?

- **We focus on one of the simplest learning task, i.e., classification, because:**
  - **Classification is a sufficiently abstract problem class that have numerous applications.**
  - **It has so far led to many successful applications of machine learning and AI.**

- **Note that supervised, unsupervised, and reinforcement learning are categorized according to the feedback we can get through the learning course, not the learning problems, thus they could, in many cases, applicable to the same learning task.**

What is classification?

- 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) can assign the correct label to any previous unseen data.

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

- In most literature, classification refers to supervised learning, i.e., labels for the training data are given to the learning algorithm.

II. Bayesian Decision Theory and Parametric Methods

Bayesian Decision Theory

- **Classification is essentially a decision problem.**

- **Idea: classify a data to the class with the highest posteriori probability**

$$P(w_j|x) = \frac{p(x|w_j)P(w_j)}{p(x)}$$

which is equivalent to (in the context of classification)

$$g_i(x) = \ln p(x|w_i) + \ln P(w_i)$$

Parametric Methods

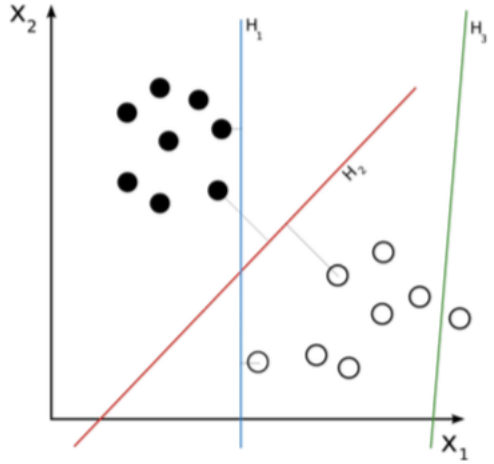
- In addition to the i. i. d assumption, further assume the data follows a specific distribution (e.g., Gaussian distribution)
- Estimate the prior and the likelihood.
- The term "parametric" comes from the assumption on the probability density function.
- Parametric methods usually do not involve parameters to fine-tune, while Nonparametric methods usually do.

III. Linear Discriminant Analysis and Support Vector Machines

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).

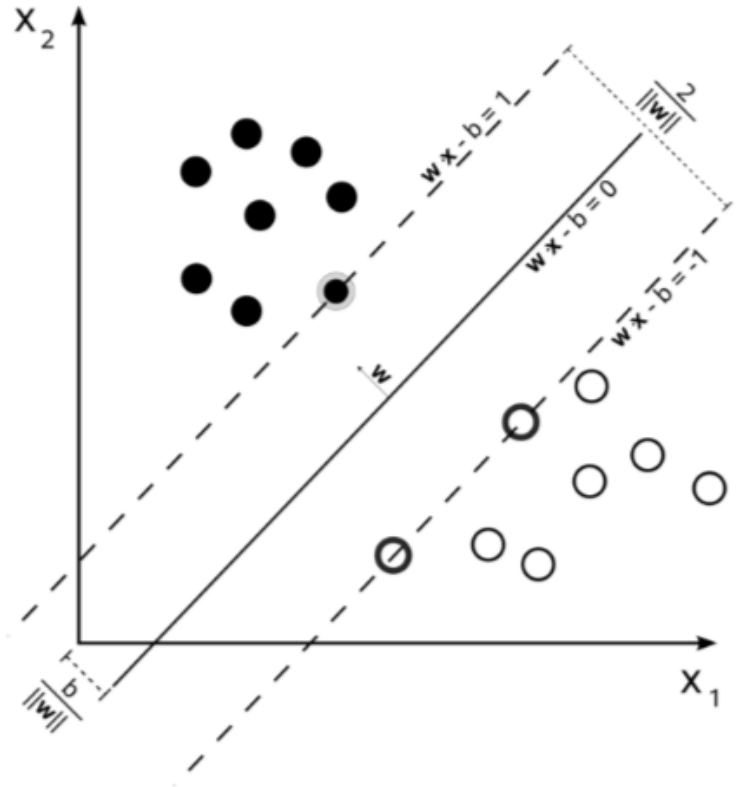
- In linearly separable case, there are more than one optimal hyperplane, SVM was motivated by this case.



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

Support Vector Machines

- **Basic idea: margin maximization**



$$\min \frac{||\boldsymbol{w}||^2}{2}$$

$$\text{s.t. } y_i(w^t x_i + b) \geq 1 \quad \forall i$$

Dual problem

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^t x_j}$$

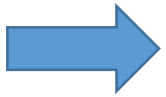
$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0; \alpha_i \geq 0 \; \forall i$$

Kernel Trick

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

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^t x_j}$$

$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0; \alpha_i \geq 0 \; \forall i$$



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

$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0; \alpha_i \geq 0 \; \forall i$$

- RBF

$$K(x_i, x_j) = \phi(\boldsymbol{x_i})^t \phi(\boldsymbol{x_j}) = \exp(-\sigma^2 \left|\left|\boldsymbol{x_i} - \boldsymbol{x_j}\right|\right|^2)$$

- Polynomial

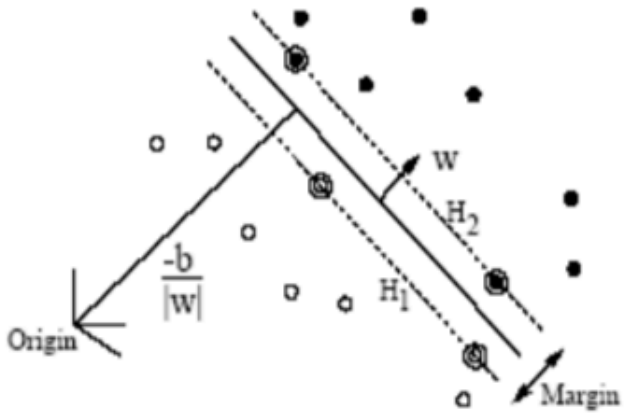
$$K(x_i, x_j) = \phi(\boldsymbol{x_i})^t \phi(\boldsymbol{x_j}) = (\boldsymbol{x_i^t x_j} - b)^p$$

- Sigmoid

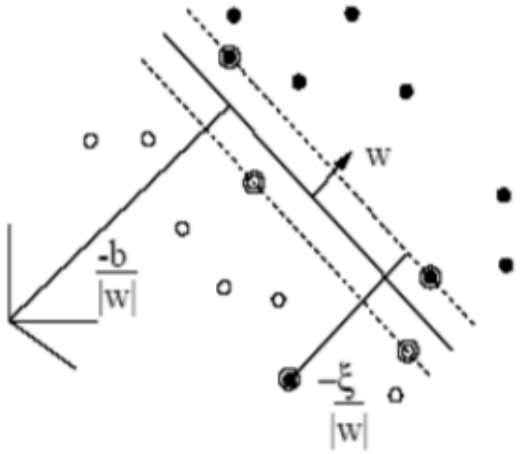
$$K(x_i, x_j) = \phi(\boldsymbol{x_i})^t \phi(\boldsymbol{x_j}) = \tanh(\boldsymbol{x_i^t x_j} - b)$$

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）

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

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

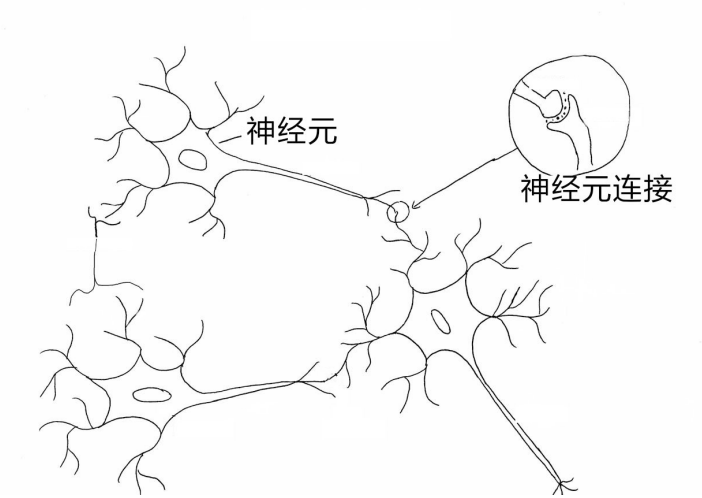
$$\max_{\alpha}\left\{\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \phi(x_i)^t\phi(x_j)\right\}$$

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

IV. Artificial Neural Networks

Inspiration (fancy story)

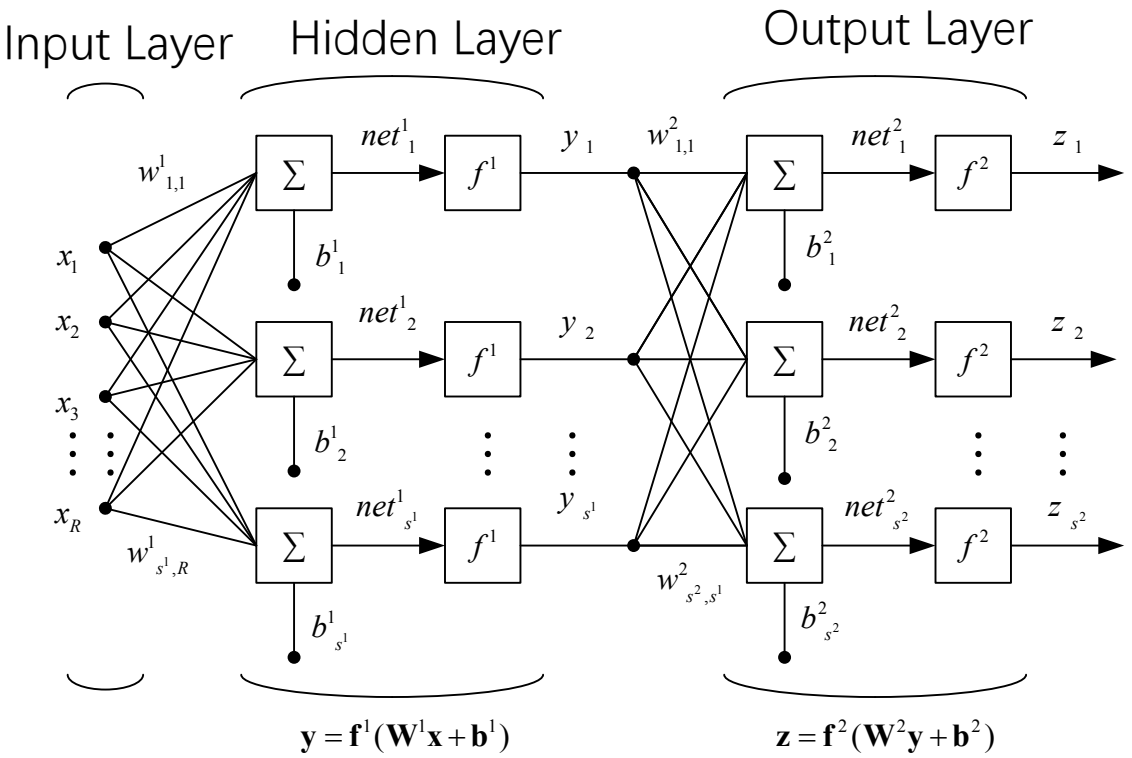
- **Inspired by biological neural networks.**



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

Fact

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



Multi-Layer Perceptron

Training NNs

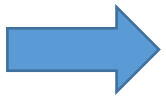
- **Optimize weights to minimize the Loss function**

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

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

Update weights between output and hidden layers

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



Chain rule

Update weights between input and hidden layers

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

More issues that worthy of knowing

- **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.**

V. Decision Trees

Nonmetric Data

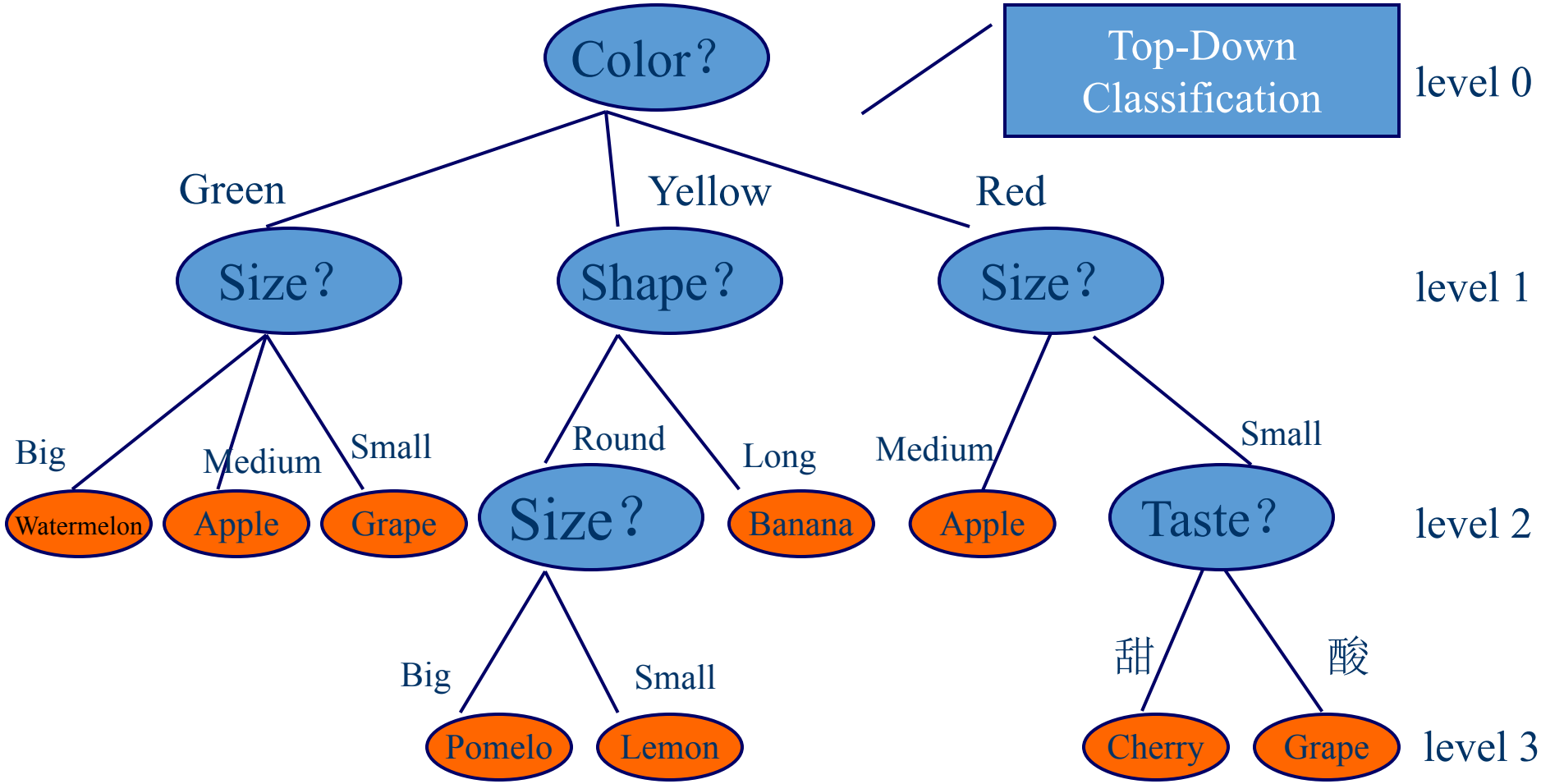
- **In many cases, data are not represented as numerical values, e.g.,**
  - **Color, taste of fruits**
  - **DNA sequences (AGCTTCAGATTCCA)**
  - **Gender, Nationality, place of birth**

- **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 (and is 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: Constructive Heuristic**

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 three types of model representation.
  - Parametric methods – estimate the distribution (no explicit optimization for methods in this lecture, but more complicated methods do need optimization)
  - SVM – Quadratic Programming
  - NN – Gradient Descent (BP)
  - DT – Heuristic Search (A specialized version)

- Recurrent NN is not covered in this lecture, while its structure/representation is quite different from MLP.

To be continued