Artificial Intelligence

wu

May 8, 2019

Contents

1 Inference and Reasoning

1.1 Propositional logic

1.2 Predicate logic

1.3 First Order Inductive Learner

knowledge graph: node = entity, edge = relation. triplet (head entity, relation, tail entity)

2 Statistical learning and modeling

2.1 Machine Learning: the concept

2.1.1 Example and concept

Supervised learning problems applications in which the **training data** comprises examples of the input vectors along with their corresponding **target vectors** are known

classification and regression

Unsupervised learning problems the training data consists of a set of input vectors X **without any corresponding target values**

density estimation, clustering, hidden markov models

Reinforcement learning problem finding suitable actions to take in a given situation in order to maximize a reward. Here the learning algorithm is not given examples of optimal outputs, in contrast to supervised learning, but must instead discover them by a process of trial and error. A general feature of reinforcement learning is the trade-off between exploration and exploitation

types of machine learning

- supervised learning
  - classification: the output is categorical or nominal variable
  - regression: the output is read-valued variable
- unsupervised learning
- semi-supervised learning

- reinforcement learning
- deep learning

2.1.2 supervised learning: important concepts

- Data: labeled instances $< \boldsymbol{x}_i, \boldsymbol{y} >$
- features: attribute-value pairs which characterize each $\boldsymbol{x}$
- learning a discrete function: **classification**
- learning a continuous function: **regression**

Classification - A two-step process

- **model construction**
- **model usage**

regression

- Example: price of a used car
  $\boldsymbol{x}$: car attributes. $\boldsymbol{y} = g(\boldsymbol{x} \mid \boldsymbol{\theta})$: price. g: model. θ parameter set.

2.2 example: polynomial curve fitting

2.3 probability theory review and notation

rules of probability

- **sum rule** $p(X) = \sum_Y p(X, Y)$
- **product rule** $p(X, Y) = p(Y|X)p(X)$

Bayes' Theorem: $p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$. Using sum rule $p(X) = \sum_Y p(X|Y)p(Y)$

probability densities.

$$p(x \in (a, b)) = \int_a^b p(x)dx$$

$$P(z) = \int_{-\infty}^z p(x)dx$$

$$\int_{-\infty}^{\infty} p(x)dx = 1 \quad p(x) \leq 0$$

expectation $\mathbb{E}[f] = \begin{cases} \sum_x p(x)f(x) & \text{discrete variables} \\ \int p(x)f(x)dx & \text{continuous variables} \end{cases}$. In either cases, $\mathbb{E}[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n)$. **conditional expectation**: $\mathbb{E}_x[f|y] = \sum_x p(x|y)f(x)$.

The **variance** of $f(x)$ is

$$\begin{aligned} var[f] &= \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] \\ &= \mathbb{E}[f(x)^2 - 2f(x)\mathbb{E}[f(x)] + \mathbb{E}[f(x)]^2] \\ &= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2 \end{aligned}$$

The **covarian ce** is

$$\begin{aligned} cov[x, y] &= \mathbb{E}_{x,y}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

the variance of the sum of two independent random variables is the sum of variance. Given

X	probability
x_1	p_1
$\dots$	$\dots$
x_n	p_n

Y	probability
y_1	q_1
$\dots$	$\dots$
y_m	q_m

$$var(X + Y) = var(X) + var(Y)$$

In case of two vectors of random variables $\boldsymbol{x}$ and $\boldsymbol{y}$, the covariance is a matrix

$$\begin{aligned} cov[\boldsymbol{x}, \boldsymbol{y}] &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])(\boldsymbol{y}^T - \mathbb{E}[\boldsymbol{y}^T])] \\ &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}\boldsymbol{y}^T] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{y}^T] \end{aligned}$$

Bayesian probabilities: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$. For a data set $\mathcal{D} = \{t_1, \ldots, t_n\}$ and assumption w, $p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$. $p(w)$ is **prior probability**, $p(\mathcal{D}|w)$ is **likelihood** (the probability $\mathcal{D}$ happens). Hence

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

Gaussian distribution.

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

μ is called **mean**, σ^2 is called **variance**, σ **standard deviation**, $\beta = 1/\sigma^2$ **precision**

$$\begin{aligned}
\mathbb{E}[x] &= \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x dx = \mu \\
\mathbb{E}[x^2] &= \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x^2 dx = \mu^2 + \sigma^2 \\
var[x] &= \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2
\end{aligned}$$

For \$D\$-dimensional vector $\boldsymbol{x}$ of continuous variables

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

To determine values for the unknown parameters given μ and σ^2 by maximizing the likelihood function. Use log.

$$\begin{aligned}
P(\boldsymbol{X}|\mu, \sigma^2) &= \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \sigma^2) \\
\Rightarrow \ln P(\boldsymbol{X}|\mu, \sigma^2) &= -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi)
\end{aligned}$$

Hence $\mu_{ML} = \frac{1}{N} \sum_{n=1}^{N} x_n$, $\sigma^2_{ML} = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu_{ML})^2$ by partial derivative. Maximum likelihood estimator for mean is unbiased, that is, $\mathbb{E}(\mu_{ML}) = \mu$. Maximum likelihood estimator for variance is biased. $\mathbb{E}(\sigma^2_{ML}) = \mathbb{E}(x^2) - \mathbb{E}(\mu^2_{ML}) = \frac{N-1}{N}\sigma^2_x$

2.4 information theory

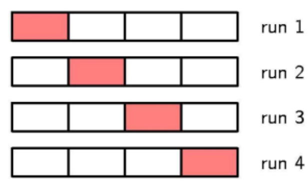
entropy: measuring uncertainty of a random variable X. $H(X) = H(p) = -\sum_{x\in\Omega} p(x)\log p(x)$ where Ω is all possible values and define $0\log 0 = 0, \log = \log_2$

$H(X) = \sum_{x\in\Omega} p(x)\log_2 \frac{1}{p(x)} = E(\log_2 \frac{1}{p(x)})$. And "information of x"="#bits to code x"$=-\log p(x)$

Kullback-Leibler divergence: comparing two distributions

2.5 model selection

The technique of S-fold cross-validation, illustrated here for the case of $S = 4$, involves taking the available data and partitioning it into S groups (in the simplest case these are of equal size). Then $S - 1$ of the groups are used to train a set of models that are then evaluated on the remaining group. This procedure is then repeated for all S possible choices for the held-out group, indicated here by the red blocks, and the performance scores from the S runs are then averaged.



cross-validation

split training data into **training set** and **validation set**. Train different models on training set and choose model with minimum error on validation set.

2.6 decision theory

Suppose we have an input vector $\boldsymbol{x}$ together with a corresponding vector $\boldsymbol{t}$ of target variables and our goal is to predict $\boldsymbol{t}$ given new value for $\boldsymbol{x}$. The joint probability distribution $p(\boldsymbol{x}, \boldsymbol{t})$ provides a complete summary of the uncertainty with these variables

3 Statistical learning and modeling - Supervised learning

3.1 Basic concepts

- **Linearly separable**
  - decision regions:
    input space is divided into several regions
  - decision boundaries:

    * under linear models, it's a linear function
    * (D-1)-dimensional hyper-plane within the D-dimensional input space

- **representation of class labels**
  - Two classes K = 2
  - K classes
    * 1-of-K coding scheme $\boldsymbol{t} = (0, 0, 1, 0, 0)^T$
  - Predict discrete class labels
    * linear model prediction $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$ w: weight vector, w_0 bias/threshold
    * nonlinear function $f(.) : R \to (0, 1)$
    * generalized linear models $y(\boldsymbol{x}) = f(\boldsymbol{w}^T\boldsymbol{x} + w_0)$ f:activation function
    * dicision surface $y(\boldsymbol{x}) = \text{constant} \to \boldsymbol{w}^T\boldsymbol{x} + w_0 = \text{constant}$

- **Three classification approaches**
  - discriminant function
    * least squares approach
    * fisher's linear discriminant
    * the perceptron algorithm of rosenblatt
  - use discriminant functions directly and don't compute probabilities

    Given discriminant functions $f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})$. Classify $\boldsymbol{x}$ as class $\mathcal{C}_k$ iff $f_k(\boldsymbol{x}) > f_j(\boldsymbol{x}), \forall j \neq k$
    * **least-squares approach**: making the model predictions as close as possible to a set of target values
    * **fisher's linear discriminant**: maximum class separation in the ouput space
    * **the perceptron algorithm of rosenblatt**
  - generative approach
    * model the class-conditional densities and the class priors
    * compute posterior probabilities through Bayes's theorem

$$\underbrace{p(\mathcal{C}_k|\boldsymbol{x})}_{\text{posterior for class}} = \frac{\overbrace{p(\boldsymbol{x}|\mathcal{C}_k)}^{\text{class conditional density}} \quad \overbrace{p(\mathcal{C}_k)}^{\text{class prior}}}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

3.2 discriminant functions

3.2.1 Two classes

- Linear discriminant function $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$
  - Dicision surface $\Omega : y(\boldsymbol{x}) = 0$
  - the normal distant from the origin to the dicision surface $\frac{\boldsymbol{w}^T\boldsymbol{x}}{\|\boldsymbol{w}\|} = -\frac{w_0}{\|\boldsymbol{w}\|}$
  - if x_A, x_B lie on the decision surface $y(\boldsymbol{x}_A) = y(\boldsymbol{x}_B) = 0$, then $\boldsymbol{w}^T(\boldsymbol{x}_A - \boldsymbol{x}_B) = 0$. hence w is orthogonal to every vector lying within . $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ is the normal vector of
  - $\boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ hence $r = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$. $y(\boldsymbol{x}_\perp) = 0 \rightarrow \boldsymbol{w}^T\boldsymbol{x} = -w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|}$
  - $\tilde{\boldsymbol{w}} = (w_0, \boldsymbol{w}), \tilde{\boldsymbol{x}} = (x_0, \boldsymbol{x}), y(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}}$

3.2.2 K-class

- One-versus-the-rest classifier K - 1 classifiers each of which solves a two-class problem

- One-versus-one classifier K(K-1)/2 binary discriminant functions

- single K-class discriminant comprising K linear functions $y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k_0}$
  - assigning a point x to class $\mathcal{C}_k$ if $y_k(\boldsymbol{x} > y_j(\boldsymbol{x}))$ for all jk
  - dicision boundary between class $\mathcal{C}_k, \mathcal{C}_j$ is given $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x}) \rightarrow (\boldsymbol{w}_k - \boldsymbol{w}_j)^T\boldsymbol{x} + (w_{k_0} - w_{j_0}) = 0$
  - $\mathcal{R}_k$ is singly connected convex
  - $\hat{\boldsymbol{x}} = \lambda\boldsymbol{x}_A + (1-\lambda)\boldsymbol{x}_B$ where $0 \leq \lambda \leq 1$, $y_k(\hat{\boldsymbol{x}}) = \lambda y_k(\boldsymbol{x}_A) + (1-\lambda)y_k(\boldsymbol{x}_B)$ and hence $\hat{x}$ also lies inside $\mathcal{R}_k$

3.2.3 Learning the parameters of linear discriminant functions

1. Linear basis function models **linear regression**: $y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1x_1 + \cdots + w_Dx_D = \boldsymbol{w}^T\boldsymbol{x}$.

   For nonlinear functions ϕ_j, $y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j\phi_j(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi(x)}$ where $\phi_j(\boldsymbol{x})$ are **basis functions**

2. **parameter optimization via maximum likelihood**

   Assume target variable t is given by a deterministic function $y(\boldsymbol{x}, \boldsymbol{w})$ with additive Gaussian noice so that $t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon$ where ϵ is a zero mean Gaussian random variable with precision β, hence we can write

   $$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1})$$

   and $\mathbb{E}(t|\boldsymbol{x}) = \int tp(t|\boldsymbol{x})dt = y(\boldsymbol{x}, \boldsymbol{w})$

   For data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}, \boldsymbol{t} = (t_1, \dots, t_n)^T,\ p(t|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1})$

   $\ln p(t|\boldsymbol{w}, \beta) = \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}) = \dfrac{N}{2}\ln\beta - \dfrac{N}{2}\ln(2\pi) - \beta E_D(\boldsymbol{w})$

   $E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right\}^2 = \dfrac{1}{2}\|t - \Phi\boldsymbol{w}\|$ is sum-of-squares error function

   solve $\boldsymbol{w}$ by maximum likelihood.

   $$\nabla \ln p(\boldsymbol{t}|\boldsymbol{w}, \beta) = \sum_{n=1}^{N}\left\{t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right\}\phi(\boldsymbol{x}_n)^T$$

   $$0 = \sum_{n=1}^{N} t_n\boldsymbol{\phi}(\boldsymbol{x}_n)^T - \boldsymbol{w}^T(\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^T)$$

   Hence we get

   $$\boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$$

   Φ is **design matrix**.

   $$\Phi = \begin{pmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \dots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \dots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \dots & \phi_{M-1}(\boldsymbol{x}_N) \end{pmatrix}$$

   For bias parameter w_0. $E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j\phi_j(\boldsymbol{x}_n)\}^2$.

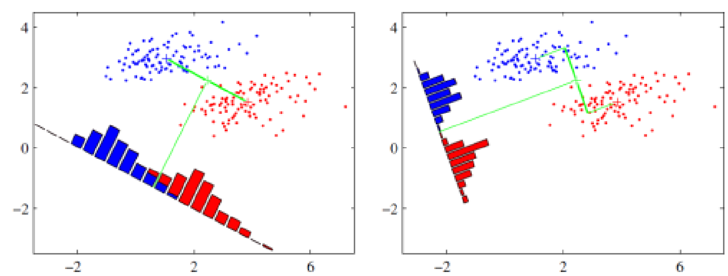
   Hence $w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j\bar{\phi_j}$, $\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n$, $\bar{\phi_j} = \frac{1}{N}\sum_{n=1}^{N}\phi_j(\boldsymbol{x}_n)$.

$$fracN2\beta = E_D(\boldsymbol{w}).\ \frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left\{t_n - \boldsymbol{w}_{ML}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right\}^2$$

3. Least-squares approach

   - Problem
     - Each class $\mathcal{C}_k$ is described by its own linear model $y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0}$
     - group together: $y(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}^T\tilde{\boldsymbol{x}},\ \tilde{\boldsymbol{w}}_k = (w_{k0}, \boldsymbol{w}_k^T)^T,\ \tilde{\boldsymbol{x}} = (1, \boldsymbol{x}^T)^T$
   - Learning
     - minimizing SSE function sum-of-squares $SSE = \sum_{i=1}^{n}(y_i - f(x_i))^2$ $E_D(\widetilde{\boldsymbol{W}}) = 1/2\mathrm{Tr}\{(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T})^T(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T})\}$
       $\widetilde{\boldsymbol{W}} = (\widetilde{\boldsymbol{X}}^T\widetilde{\boldsymbol{X}})^{-1}\widetilde{\boldsymbol{X}}^T\boldsymbol{T}$

4. fisher's linear discriminant

   $y \geqslant -w_0$ as class $\mathcal{C}_1$ belonging to $\mathcal{C}_1$ if $y(\mathbf{x}) \geqslant y_0$

   from the view of dimensionality reduction $y \geq -w_0$ as class $\mathcal{C}_1$

   $$m_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1} x_n, m_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2} x_n \xrightarrow{y=\boldsymbol{w}^T\boldsymbol{x}} m_2 - m_1 = \boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

5. the perceptron algorithm of rosenblatt

3.3 probalibilistic generative models

A probabilistic view of classification from simple assumptions about the distribution of the data

$$
\begin{aligned}
p(\mathcal{C}_1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\
&= \frac{1}{1+\exp(-a)} = \sigma(a)
\end{aligned}
$$

where

$$a = \ln \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

and $\sigma(a)$ is the **logistic sigmoid** function defined by

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

and $\sigma(-a) = 1 - \sigma(a)$, its inverse is **logit** function

$$a = \ln(\frac{\sigma}{1-\sigma})$$

For case of $K > 2$ classes, we have the following **multi-class generalization**

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, a_k = \ln\left[p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)\right]$$

The **normalized exponential** is known as the **softmax function** as it represents a *smoothed version of the max function*

$$\text{if } a_k \ll a_j, \forall j \neq k, \text{then } p(\mathcal{C}_k|\boldsymbol{x}) \approx 1, p(\mathcal{C}_j|\boldsymbol{x}) \approx 0$$

For **continuous inputs**, assume

$$p(\boldsymbol{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma^{-1}} (\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}$$

1. 2 classes

$$
\begin{aligned}
p(\mathcal{C}_1|\boldsymbol{x}) &= \sigma(\boldsymbol{w}^T\boldsymbol{x} + w_0) \\
\boldsymbol{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}
\end{aligned}
$$

2. K classes

$$
\begin{aligned}
a_k(\boldsymbol{x}) &= \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0} \\
\boldsymbol{w}_k &= \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k \\
w_{k0} &= -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(\mathcal{C}_k)
\end{aligned}
$$

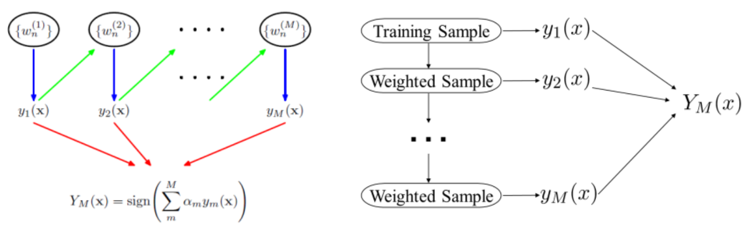
3.4 probabilistic discriminative models

3.5 Boosting

Originally designed for classification problems.

Motivation: a procedure that combines the outputs of many "weak" classifiers to produce a strong/accurate classifier

3.5.1 AdaBoost



4 wef

4.1 wfe

K-means