REVIEW OUTLINE

CONTENT

1. Introduction
2. Overview of Supervised Learning
3. Linear Model
4. Decision Tree
5. Neural Network (NN)

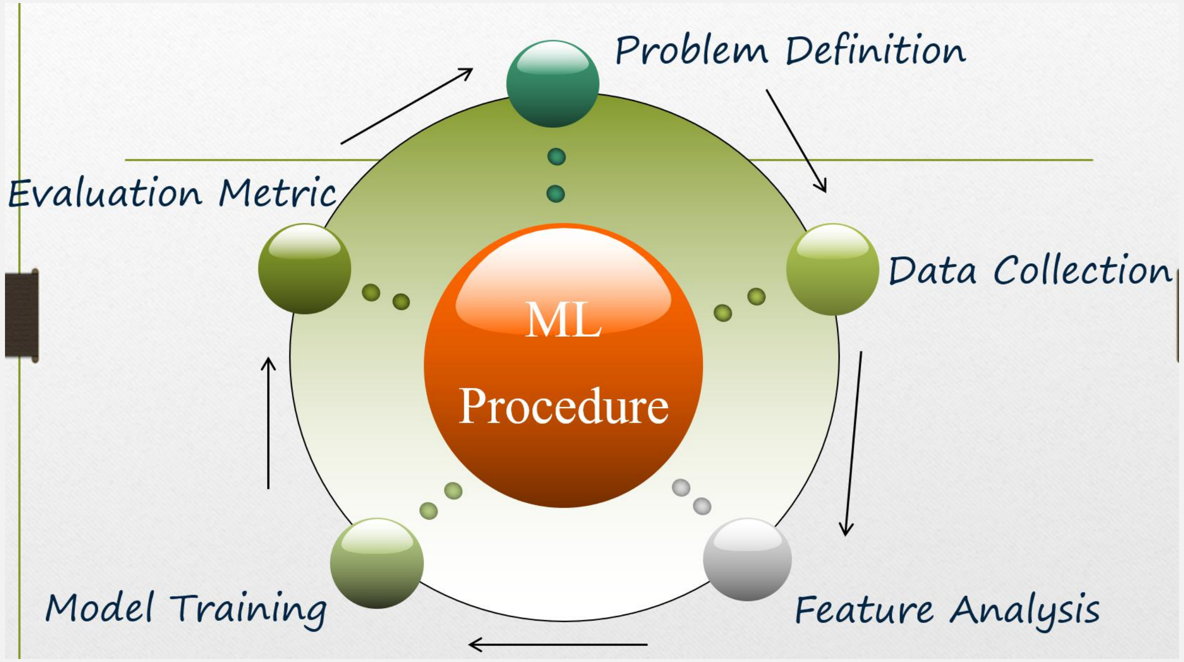
INFO OF EXAM

- Closed-book examination: 2024/01/03 (Wednesday) 10:20-12:10
- Presentation (Tencent Meeting, last chance): 2023/12/18 (Monday), 08:30-10:30
- Q & A (Tencent Meeting): 2020/12/25 (Monday), 08:30-10:00

- Problem types (at least 4):
  - Fill-in-the-blank (30%)
  - True/False (10%)
  - Short questions (20%)
  - Comprehensive questions (40%)

1. INTRODUCTION

1.1 General Procedure in ML



1. INTRODUCTION

1.2 Supervised vs. Unsupervised vs. Semi-supervised

- Supervised learning: learn with labeled training data
- Unsupervised learning: learn with unlabeled training data
- Semi-supervised: a small amount of labeled data with a large amount of unlabeled data.
  - Train model with labeled data
  - Use the learned model to predict unlabeled data, then adjust parameter

2. OVERVIEW OF SUPERVISED LEARNING

2.1 Least squares

- Linear model: given a vector of inputs $X^T = (X_1, X_2, ..., X_p)$

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j.$$

bias weight

- Residual sum of squares (平方残差和):

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2.$$

$$\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta),$$

2. OVERVIEW OF SUPERVISED LEARNING

2.2 k-nearest neighbors

- $N_k(x)$ is the neighborhood of x defined by the k closest points x_i

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

- Common closeness measurement
  - Euclidean distance

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

2. OVERVIEW OF SUPERVISED LEARNING

2.3 Loss function vs. Expected prediction error

- Loss function: penalize errors in prediction
  - Squared loss function for regression (*f* is continuous)
  - Zero-one loss function for classification (*f* is discrete)
- Expected prediction error : expectation of loss function
- Optimal prediction: to minimize the EPE

2. OVERVIEW OF SUPERVISED LEARNING

2.4 Curse of dimensionality

- When the dimensionality increases, the volume of the space increases so fast that the available data becomes sparse
- The amount of data needed to support the result often grows exponentially with the dimensionality
- Difficult for sampling; local methods inefficient

2. OVERVIEW OF SUPERVISED LEARNING

2.5 Bias–variance decomposition

- MSE = variance + squared bias

$$
\begin{aligned}
\mathrm{MSE}(x_0) &= \mathrm{E}_{\mathcal{T}}[f(x_0) - \hat{y}_0]^2 \\
&= \mathrm{E}_{\mathcal{T}}[\hat{y}_0 - \mathrm{E}_{\mathcal{T}}(\hat{y}_0)]^2 + [\mathrm{E}_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\
&= \mathrm{Var}_{\mathcal{T}}(\hat{y}_0) + \mathrm{Bias}^2(\hat{y}_0).
\end{aligned}
$$

- Variance: changes in learning performance due to changes in the training set, i.e., impact of data perturbation

$$\mathrm{E}_{\mathcal{T}}[\hat{y}_0 - \mathrm{E}_{\mathcal{T}}(\hat{y}_0)]^2$$

- Bias: deviation between the expected and real results of the learning algorithm, i.e., fitting ability of learning algorithm

$$\mathrm{E}_{\mathcal{T}}(\hat{y}_0) - f(x_0)$$

29

$$
\begin{aligned}
& E_T[\hat{y}_0 - E_T(\hat{y}_0)]^2 + [E_T(\hat{y}_0) - f(x_0)]^2 \\
&= E_T[\hat{y}_0^{\,2} - 2\hat{y}_0 E_T(\hat{y}_0) + (E_T(\hat{y}_0))^2] + E_T(\hat{y}_0)^2 - 2E_T(\hat{y}_0) f(x_0) + f(x_0)^2 \\
&= E_T(\hat{y}_0^{\,2}) - 2E_T(\hat{y}_0)E_T(\hat{y}_0) + E_T(\hat{y}_0)^2 + E_T(\hat{y}_0)^2 - 2E_T(\hat{y}_0) f(x_0) + f(x_0)^2 \\
&= E_T(\hat{y}_0^{\,2}) - 2E_T(\hat{y}_0) f(x_0) + f(x_0)^2 \\
&= E_T(\hat{y}_0^{\,2}) - 2E_T(\hat{y}_0 f(x_0)) + E_T(f(x_0)^2) \\
&= E_T[(\hat{y}_0^{\,2}) - 2\hat{y}_0 f(x_0) + f(x_0)^2] \\
&= E_T[f(x_0) - \hat{y}_0]^2
\end{aligned}
$$

3.LINEAR MODEL

3.1 Linear regression

- **Univariate linear regression**

$$f(x_i) = wx_i + b$$ such that $f(x_i) \approx y_i$

Where x_i is a scalar

- **Multivariate linear regression**

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b$$ such that $f(x_i) \approx y_i$

Where x_i is a vector

- **Generalized linear model**

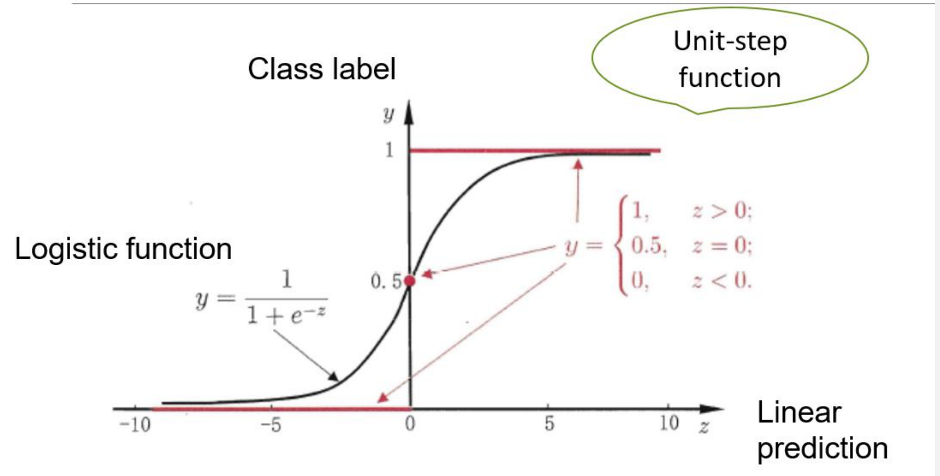
$$y = g^{-1}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

Where $g(\cdot)$ is a monotone differentiable function

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2},$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$$

$$\hat{\boldsymbol{w}}^* = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$



- Posterior probability estimation

$$\ln \frac{y}{1-y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \Rightarrow \ln \frac{p(y=1 \mid \boldsymbol{x})}{p(y=0 \mid \boldsymbol{x})} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

$$\Downarrow$$

$$p(y=1 \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}},$$

$$p(y=0 \mid \boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}.$$

$\Leftarrow$ maximum likelihood method

$\boxed{?}$

$$\prod_{i=1}^{m} p(y_i \mid x_i; w, b)$$

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{m}\left(-y_i\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}_i + \ln\left(1+e^{\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}_i}\right)\right)$$

$$l(w,b) = \sum_{i=1}^{m} \ln p(y_i \mid x_i; w, b)$$

Max $l(w,b) \leftrightarrow$ Min $l(\beta)$

$$= \sum_{i=1}^{m} \ln[y_i p_1(\hat{x}_i;\beta) + (1-y_i) p_0(\hat{x}_i;\beta)]$$

$$= \sum_{i=1}^{m} \ln[y_i \frac{e^{\beta^T \hat{x}_i}}{1+e^{\beta^T \hat{x}_i}} + (1-y_i)\frac{1}{1+e^{\beta^T \hat{x}_i}}]$$

$$= \sum_{i=1}^{m} \ln \frac{y_i e^{\beta^T \hat{x}_i} + (1-y_i)}{1+e^{\beta^T \hat{x}_i}}$$

$$= \begin{cases} y_i = 1, & \sum_{i=1}^{m} \ln \frac{e^{\beta^T \hat{x}_i}}{1+e^{\beta^T \hat{x}_i}} = \boxed{\sum_{i=1}^{m} (\beta^T \hat{x}_i - \ln(1+e^{\beta^T \hat{x}_i}))} \\ y_i = 0, & \sum_{i=1}^{m} \ln \frac{1}{1+e^{\beta^T \hat{x}_i}} = \boxed{\sum_{i=1}^{m} (-\ln(1+e^{\beta^T \hat{x}_i}))} \end{cases}$$

3.LINEAR MODEL

3.2 Linear discriminant analysis (LDA)

- Cast the samples onto a straight line
- Project the similar samples as close as possible
- Project the dissimilar samples as far as possible
- For a new sample, determine the class according to the relative position of its projection point.

● Goal: maximize

$$J = \frac{\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\|_2^2}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}} = \frac{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\boldsymbol{w}}$$

$$\Downarrow$$

$$J = \frac{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_b\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w}}$$

● Within-class scatter matrix

$$\mathbf{S}_w = \boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1 = \sum_{\boldsymbol{x}\in X_0}(\boldsymbol{x} - \boldsymbol{\mu}_0)(\boldsymbol{x} - \boldsymbol{\mu}_0)^{\mathrm{T}} + \sum_{\boldsymbol{x}\in X_1}(\boldsymbol{x} - \boldsymbol{\mu}_1)(\boldsymbol{x} - \boldsymbol{\mu}_1)^{\mathrm{T}}$$

● Between-class scatter matrix

$$\mathbf{S}_b = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\mathrm{T}}$$

$$L(w, \lambda) = -w^T S_b w + \lambda(w^T S_w w - 1)$$

$$\frac{\partial L}{\partial w} = -2S_b w + 2\lambda S_w w = 0 \Rightarrow S_b w = \lambda S_w w$$

$$J = \frac{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_b\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w}} \xrightarrow{\text{set } \boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w} = 1} \min_{\boldsymbol{w}} \ -\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_b\boldsymbol{w} \quad \text{s.t.} \ \boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w} = 1 \xrightarrow{\text{Lagrange multipliers}} \mathbf{S}_b\boldsymbol{w} = \lambda\mathbf{S}_w\boldsymbol{w}$$

$$\boldsymbol{w} = \mathbf{S}_w^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)$$

4.DECISION TREE

4.1 Basic algorithm

1. If all the instances are from exactly one class, then the decision tree is an answer node containing that class name.
2. Otherwise,
   (a) Define a_{best} to be an attribute with $\boxed{\text{some mechanism}}$
   (b) For each value $v_{best,i}$ of a_{best}, grow a branch from a_{best} to a decision tree constructed recursively from all those instances with value $v_{best,i}$ of attribute a_{best}.

4.DECISION TREE

4.2 Attribute Selection

- Information gain
- Gain ratio
- Gini index

$$\mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v)$$

$$\mathrm{Gain\_ratio}(D,a) = \frac{\mathrm{Gain}(D,a)}{\mathrm{IV}(a)} \qquad \mathrm{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

$$\mathrm{Gini\_index}(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Gini}(D^v)$$

4.DECISION TREE

4.3 Bi-partition for continuous value

- Sort *n* distinct values on a continuous attribute *a* $\{a^1, a^2, \dots, a^n\}$
- Split *D* into D_t^+ and D_t^- w.r.t. a splitting point *t*
- Candidate set for *t*

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leqslant i \leqslant n-1 \right\}$$

- Choose the best *t*

$$\begin{aligned} \mathrm{Gain}(D,a) &= \max_{t \in T_a} \mathrm{Gain}(D,a,t) \\ &= \max_{t \in T_a} \mathrm{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \mathrm{Ent}(D_t^\lambda) \end{aligned}$$

4.DECISION TREE

4.4 Reweight for missing value

- Information gain:

$$\text{Gain}(D,a) = \rho \times \text{Gain}(\tilde{D},a)$$

$$= \rho \times \left(\text{Ent}\left(\tilde{D}\right) - \sum_{v=1}^{V} \tilde{r}_v \,\text{Ent}\left(\tilde{D}^v\right)\right)$$

$$\text{Ent}(\tilde{D}) = -\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k$$

- Reset the *weight* w_x *of sample x* if need:
  - If x has some value on a, just keep w_x
  - Otherwise, first join x into each node corresponding to a^v and then set the weight of x to $\tilde{r}_v \cdot w_x$

4.DECISION TREE

4.5 Random forest

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

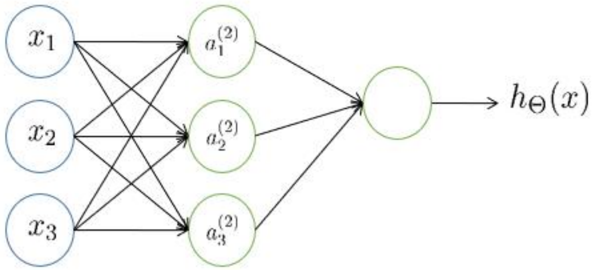
To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

5. NEURAL NETWORKS (NN)

5.1 Model representation



$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_\Theta(x) = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

Vectorized implementation

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{(1)}x$$
$$a^{(2)} = g(z^{(2)})$$

Add $a_0^{(2)} = 1$

$$z^{(3)} = \Theta^{(2)}a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

5. NEURAL NETWORKS (NN)

5.2 Backpropagation algorithm

Backpropagation algorithm

Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Set $\triangle_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

$$a_j^{l+1} = g\left(\sum_{i=1}^{S_l} \theta_{ji}^l \cdot a_i^l\right)$$

Using $y^{(i)}$, compute $\delta^{(L)} = (a_i^{(l)} - y) \cdot g'(z_i^{(l)})$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

$$\boxed{\delta_i^{(l)} = \frac{\partial E}{\partial z_i^{(l)}} = \sum_j^{N^{(l+1)}} (\delta_j^{(l+1)} \cdot \Theta_{ji}^{(l)}) \cdot g'(z_i^{(l)})}$$

$$\text{reset weights}: \theta_{ji}^l = \theta_{ji}^l - \alpha \cdot \delta_j^{(l+1)} \cdot a_i^{(l)}$$

5. NEURAL NETWORKS (NN)

5.3 Convolutional Neural Network (CNN)