

MachLe - Résumé Olivier D'Ancona

Evaluation Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{Specificity} = \frac{TN}{TN + FP}$$
$$\text{Fscore} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
$$\text{error rate} = 1 - \text{accuracy}$$
$$\text{macro average} = \frac{1}{n} \sum_{i=1}^n \text{avg}_i$$

Activation Functions

Sigmoid : $\frac{1}{1 + e^{-x}}$

Hyperbolic tangent : $\frac{e^x - e^{-x}}{e^x + e^{-x}}$

Relu : $\begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$

Gaussian : e^{-x^2}

Softmax : $\frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$

Neural Network

Structure

Biais : b , An extra weight that can be learned using a learning algorithm. The purpose is to replace threshold.

Input : I , Input vector

Weights : W , Vector of weights

Learning algorithm

1. Randomly initialize weights
2. Compute the neuron's output for a given input vector X
3. Update weights : $W_j(t+1) = W_j(t) + \eta(\hat{y}_i - y)x$ with η the learning rate and \hat{y}_i the desired output.
4. Repeat steps 2 and 3 for the number of epochs you need or until the error is smaller than a threshold.

KNN for classification and regression

Hyperparameters :

- Number of neighbours k
- Distance metric
- normalization type
- strategy if no majority

Big k :

(+) More confidence, probabilistic (-) No locality, heavier

Bayes

Théorème de Bayes :

$$P(C_k|x) = \frac{P(x|C_k) \cdot P(C_k)}{P(x)}$$

où

- C_k : Classe ciblée
- x : Évidence
- $P(C_k)$: Probabilité a priori de la classe C_k
- $P(x|C_k)$: probability of observing x given class j
- $P(C_k|x)$: Probabilité a posteriori de la classe C_k après observation de x
- $P(x)$: Probabilité de l'évidence x

avec

$$P(x) = \sum_{\text{toutes classes } C_k} P(x|C_k) \cdot P(C_k)$$

Exemple Classificateur Fille/Garçon :

- $P(C_f) = \frac{4}{70}, P(C_g) = \frac{66}{70}$
- $p(x|C_g) = 0.8, p(x|C_f) = 0.2$
- Calcul de $p(x)$:

$$p(x) = 0.2 \times \frac{4}{70} + 0.8 \times \frac{66}{70}$$

- Calcul de $P(C_f|x)$ et $P(C_g|x)$:

$$P(C_f|x) = \frac{0.2 \times \frac{4}{70}}{p(x)}, \quad P(C_g|x) = \frac{0.8 \times \frac{66}{70}}{p(x)}$$

(+)Can deal with imbalanced dataset, prior can be changed

Linear Regression

Soit un tableau de données :

x = Surface(g) , y = Price(cm) , $x \cdot y$, x^2

$$X = [1, \text{Surface}]$$

$$X^T X = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} = \begin{bmatrix} 7 & 38.5 \\ 38.5 & 218.95 \end{bmatrix}$$

$$X^T y = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix} = \begin{bmatrix} 348 \\ 1975 \end{bmatrix}$$

$$\hat{\theta} = (X^T X)^{-1} X^T y = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -2.67 \\ 9.51 \end{bmatrix}$$

$$\hat{y} = \theta_0 + \theta_1 x$$

Inverse d'une matrice 2x2 :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

System engineering / Data preparation / Tuning

Logistic regression / Overfitting

SVM

Clustering

Decision Trees

Convolutional Neural Networks

Recurrent Neural Networks

Dimensionality Reduction

Reinforcement Learning

Computational Complexity of ML Algorithms

Algorithm	Assumption	Train Time/Space	Inference Time/Space
KNN (Brute Force)	Similar things exist in close proximity	$O(knd) / O(nd)$	$O(knd) / O(nd)$
KNN (KD Tree)	Similar things exist in close proximity	$O(nd \log(n)) / O(nd)$	$O(k \log(n)d) / O(nd)$
Naive Bayes	Features are conditionally independent	$O(ndc) / O(dc)$	$O(dc) / O(dc)$
Logistic Regression	Classes are linearly separable	$O(nd) / O(nd)$	$O(d) / O(d)$
Linear Regression	Linear relationship between variables	$O(nd) / O(nd)$	$O(d) / O(d)$
SVM	Classes are linearly separable	$O(n^2d^2) / O(nd)$	$O(kd) / O(kd)$
Decision Tree	Feature selection by information gain	$O(n \log(n)d) / O(\text{nodes})$	$O(\log(n)) / O(\text{nodes})$
Random Forest	Low bias and variance trees	$O(kn \log(n)d) / O(\text{nodes} \times k)$	$O(k \log(n)) / O(\text{nodes} \times k)$
GBDT	High bias, low variance trees	$O(Mn \log(n)d) / O(\text{nodes} \times M + \gamma_m)$	$O(M \log(n)) / O(\text{nodes} \times M + \gamma_m)$