Introduction to Big Data Analysis
Classification : Part 1

Zhen Zhang

Southern University of Science and Technology

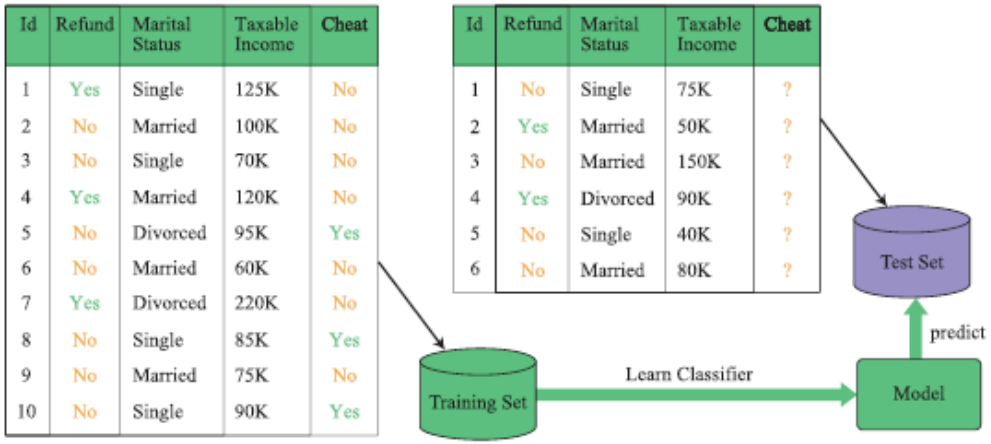
Outlines

Why We Need Classification

- Knowing the classes of the data, we could easily manage the data and react to the possible outcomes
- Predict whether users would default in the future based on their basic information and historical transaction records
- Predict whether a tumor is benign or malignant based on their physical and geometrical features
- Predict the users' interests in the new products based on their historical purchasing records and behaviorial preferences
- Separate spams and advertisements from emails

What is Classification

- Supervised learning : predict label y from features $\mathbf{x}$
- Training stage : Given a data set $D = \{(\mathbf{x}, y)\}$, including both features and labels, split $D = D_{train} \bigcup D_{test}$, find a classifier (function $y = f(\mathbf{x})$) that best relates y_{train} with $\mathbf{x}_{train}$, then evaluate how close $f(\mathbf{x}_{test})$ is to y_{test}
- Predicting stage : apply the predictor to the unlabeled data $\mathbf{x}_{pred}$ (only features) to find the proper labels $y_{pred} = f(\mathbf{x}_{pred})$



Id	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Id	Refund	Marital Status	Taxable Income	Cheat
1	No	Single	75K	?
2	Yes	Married	50K	?
3	No	Married	150K	?
4	Yes	Divorced	90K	?
5	No	Single	40K	?
6	No	Married	80K	?

Classification Methods

- Different assumptions on f lead to different models
- Basic classification models
  - k-nearest neighbor (kNN)
  - Decision trees
  - Naive Bayes
  - Support vector machines (SVM)
  - Logistic regression
  - Linear discriminant analysis (LDA)
  - Artificial neural network (ANN)
  - ...
- Ensemble learning : Random forest and Adaboost

Outlines

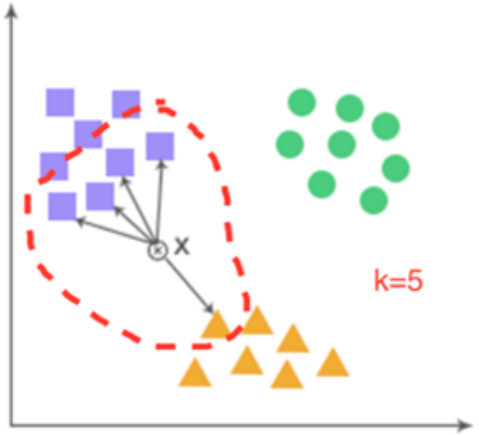
Introduction

- k-nearest neighbor (kNN) is the simplest supervised learning method, especially useful when prior knowledge on the data is very limited
- Do training and test simultaneously
- When classifying a test sample x, scan the training set and find the closest k samples $D_k = \{x_1, \ldots, x_k\}$ to the test sample ; make vote based on the labels of the samples in D_k ; the majority vote is the label of the test sample
- Low bias, high variance
- Advantages : not sensitive to outliers, easy to implement and parallelize, good for large training set
- Drawbacks : need to tune k, take large storage, computationally intensive

Algorithm

- Input : training set $D_{train} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, a test sample x without label y, k and distance metric $d(x, y)$
- Output : predicted label y_{pred} for x

1. Compute $d(x, x_j)$ for each $(x_j, y_j) \in D_{train}$
2. Sort the distances in an ascending order, choose the first k samples $(x_{(1)}, y_{(1)}), \ldots, (x_{(k)}, y_{(k)})$
3. Make majority vote $y_{pred} = \mathrm{Mode}(y_{(1)}, \ldots, y_{(k)})$



Distance Metrics

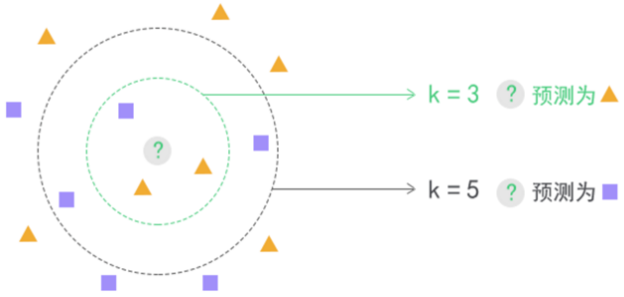
- Minkowski distance : $d_h(\mathbf{x}_1, \mathbf{x}_2) = \sqrt[h]{\sum_{i=1}^{d}(x_{1i} - x_{2i})^h}$; $h = 2$, Euclidean distance ; $h = 1$, Manhattan distance
- Mahalanobis distance : $d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \hat{\Sigma}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)}$, where $\hat{\Sigma}$ is the covariance matrix of sample set ; introduce correlations, could be applied to the non-scaling data
- Hamming distance : $Hamming(\mathbf{x}_1, \mathbf{x}_2) = d - \sum_{i=1}^{d} I(x_{1i} = x_{2i})$; used to compare two strings, e.g., $Hamming('toned', 'roses') = 3$, $Hamming('101110', '101101') = 2$

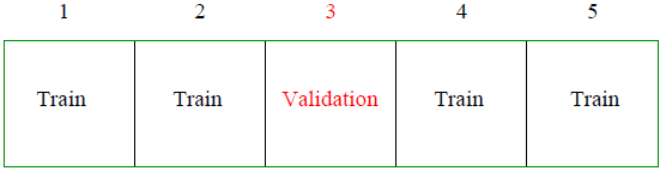
Distance Metrics - Similarity and Divergence

- Cosine similarity : $\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1^T \mathbf{x}_2}{|\mathbf{x}_1||\mathbf{x}_2|} = \frac{\sum_{i=1}^{d} x_{1i} x_{2i}}{\sqrt{\sum_{i=1}^{d} x_{1i}^2} \sqrt{\sum_{i=1}^{d} x_{2i}^2}}$; its range is $[-1, 1]$; the greater the cosine similarity, the more similar (closer) the two samples ; insensitive to absolute value, popular in measuring user rankings ; it is related to Pearson correlation coefficient
- Jaccard similarity for sets A and B : $Jaccard(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$, used in comparing texts
- Kullback-Leibler (KL) divergence : $d_{KL}(P \| Q) = \mathrm{E}_P \left[\log \frac{P(x)}{Q(x)} \right]$ measures the distance between two probability distributions P and Q ; in discrete case, $d_{KL}(p \| q) = \sum_{i=1}^{m} p_i \log \frac{p_i}{q_i}$

Tuning k

- Different values of $k = 3$ and $k = 5$ leads to different classification results
- M-fold Cross-validation (CV) to tune k : partition the dataset into M parts ($M = 5$ or 10), let $\kappa : \{1, \ldots, N\} \to \{1, \ldots, M\}$ be randomized partition index map, The CV estimate of prediction error is $CV(\hat{f}, k) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, k))$





Bayes Classifier (Oracle Classifier)

- Assume $Y \in \mathcal{Y} = \{1, 2, \ldots, C\}$, the classifier $f : \mathcal{X} \to \mathcal{Y}$ is a piecewise constant function
- For 0-1 loss $L(y, f)$, the learning problem is to minimize

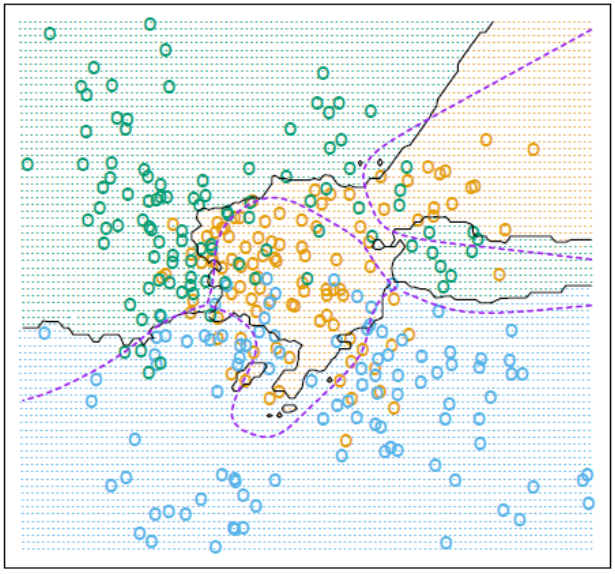
$$\begin{aligned}\mathcal{E}(f) &= \mathrm{E}_{\mathrm{P}(X,Y)} L(Y, f(X)) = 1 - \mathrm{P}(Y = f(X)) \\ &= 1 - \int_{\mathcal{X}} \mathrm{P}(Y = f(X) | X = x) p_X(x) \mathrm{d}x\end{aligned}$$

- Bayes rule : $f^*(x) = \arg\max_c \mathrm{P}(Y = c | X = x)$, "the most probable label under the conditional probability on x"
- Bayes error rate : $\inf_f \mathcal{E}(f) = \mathcal{E}(f^*) = 1 - \mathrm{P}(Y = f^*(X))$
- Bayes decision boundary : the boundary separating the K partition domains in $\mathcal{X}$ on each of which $f^*(x) \in \mathcal{Y}$ is constant. For binary classification, it is the level set on which $\mathrm{P}(Y = 1 | X = x) = \mathrm{P}(Y = 0 | X = x) = 0.5$.

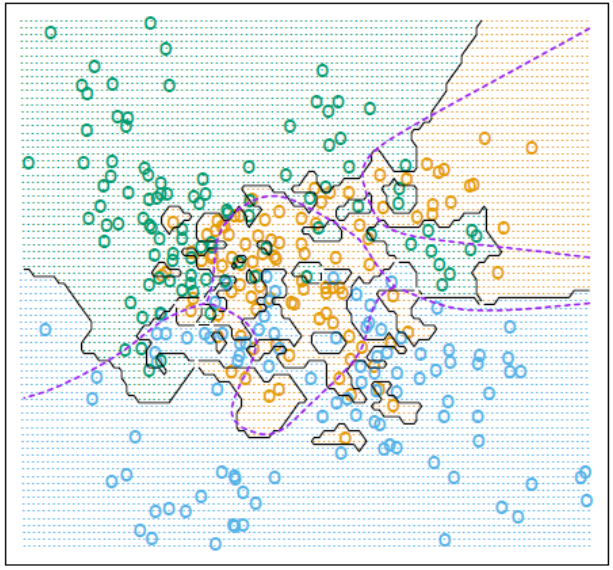
Decision Boundary

- The decision boundary of 15NN is smoother than that of 1NN

15-Nearest Neighbors



1-Nearest Neighbor



Analysis

- Time complexity : $O(mndK)$ where n is the number of training samples, m is the number of test samples, d is the dimension, and K is the number of nearest neighbors
- KD tree for indexing : K-dimensional binary search tree
- 1NN error rate is twice the Bayes error rate :
  - Bayes error $= 1 - p_{c^*}(x)$ where $c^* = \arg\max_c p_c(x)$
  - Assume the samples are i.i.d., for any test sample x and small δ, there is always a training sample $z \in B(x, \delta)$ (the label of x is the same as that of z), then 1NN error is

$$\begin{aligned}\epsilon = \sum_{c=1}^{C} p_c(x)(1 - p_c(z)) &\xrightarrow{\delta \to 0} 1 - \sum_{c=1}^{C} p_c^2(x) \\ &\leqslant 1 - p_{c^*}^2(x) \\ &\leqslant 2(1 - p_{c^*}(x))\end{aligned}$$

(Remark : In fact, $\epsilon \leqslant 2(1 - p_{c^*}(x)) - \frac{C}{C-1}(1 - p_{c^*}(x))^2$)

Case Study

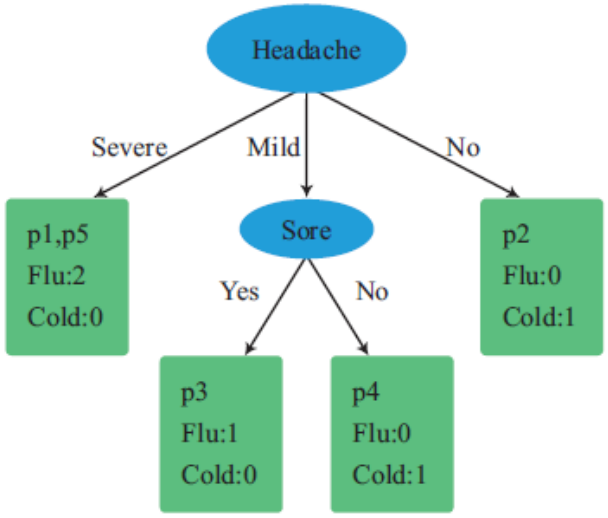
- Use kNN to diagnose breast cancer (cookdata)
- Data scaling : 0-1 scaling or z-score scaling
- from sklearn.neighbors import KNeighborsClassifier
- KNeighborsClassifier(n_neighbors = 10, metric = 'minkowski', p=2)

- radius （半径）
- texture （质地）
- perimeter （周长）
- area （面积）
- smoothness （光滑度）
- compactness （致密性=$perimeter^2/area - 1.0$）
- concavity （凹度）
- concave points (凹点)
- symmetry （对称性）
- fractal dimension （分形维数）

Outlines

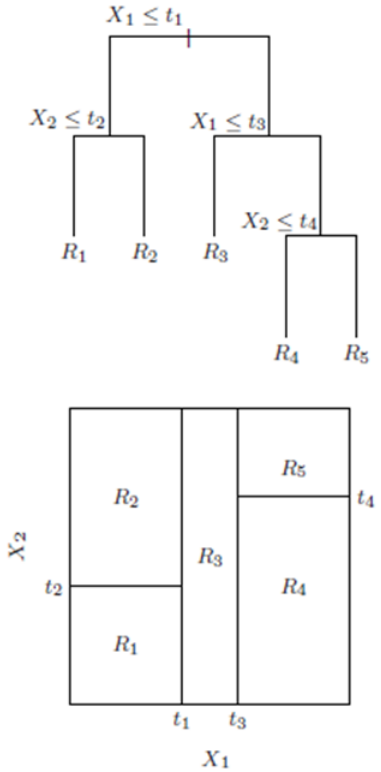
Decision Tree as Medical Diagnosis

- Diagnose whether it is flu or cold
- Rules :
  - If headache = severe, then flu
  - If headache = mild and sore = yes, then flu
  - If headache = mild and sore = no, then cold
  - If headache=no, cold



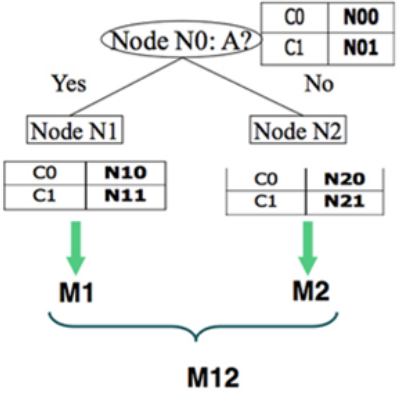
Decision Tree Algorithm

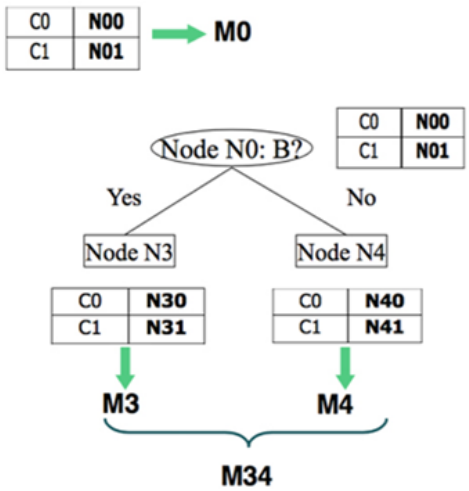
- Tree structure : internal nodes indicate features, while leaf nodes represent classes
- Start from root, choose a suitable feature x_i and its split point c_i at each internal node, split the node to two child nodes depending on whether $x_i \leqslant c_i$, until the child nodes are pure
- Equivalent to rectangular partition of the region



How to choose features and split points

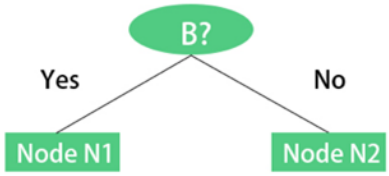
- Impurity : choose the feature and split point so that after each slit the impurity should decrease the most
- Impurity(M0)-Impurity(M12) $>$ Impurity(M0)-Impurity(M34), choose A as split node ; otherwise choose B





Impurity Measures - GINI Index

- Gini index of node t : $Gini(t) = 1 - \sum_{c=1}^{C}(p(c|t))^2$ where $p(c|t)$ is the proportion of class-c data in node t
- Maximum at $1 - \frac{1}{C}$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some c
- Gini index of a split : $Gini_{split} = \sum_{k=1}^{K} \frac{n_k}{n} Gini(k)$ where n_k is the number of samples in the child node k, $n = \sum_{k=1}^{K} n_k$
- Choose the split so that $Gini(t) - Gini_{split}$ is maximized



	Parent
C1	6
C2	6
Gini = 0.500	

	N1	N2
C1	5	1
C2	2	4
Gini=0.371		

Gini(N1)
= 1 – (5/7)² – (2/7)²
= 0.408

Gini(N2)
= 1 – (1/5)² – (4/5)²
= 0.32

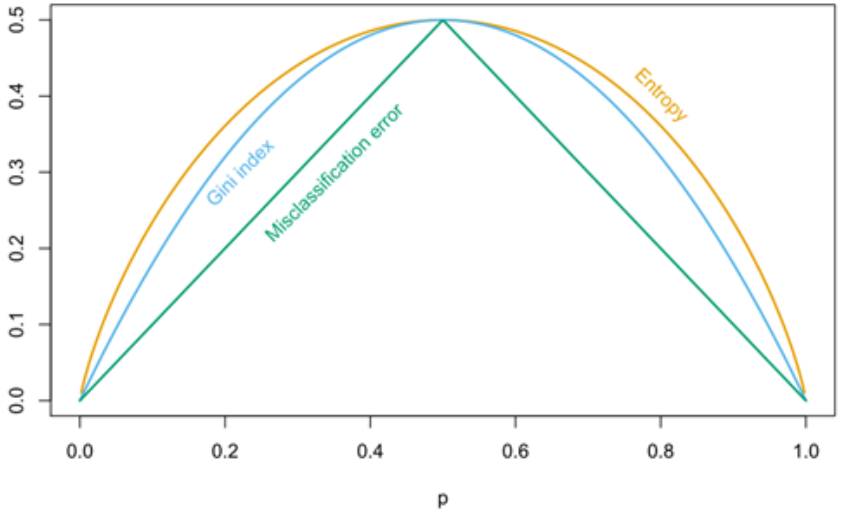
Gini(Children)
= 7/12 * 0.408 +
5/12 * 0.32
= 0.371

Impurity Measures - Information Gain

- Entropy at t : $H(t) = -\sum_{c=1}^{C} p(c|t) \log_2 p(c|t)$
- Maximum at $\log_2 C$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some c
- Information gain : $InfoGain_{split} = H(t) - \sum_{k=1}^{K} \frac{n_k}{n} H(k)$ where n_k is the number of samples in the child node k, $n = \sum_{k=1}^{K} n_k$
- Choose the split so that $InfoGain_{split}$ is maximized (ID3 algorithm)
- Drawback : easy to generate too many child nodes and overfit
- Introduce information gain ratio : $SplitINFO = -\sum_{k=1}^{K} \frac{n_k}{n} \log_2 \frac{n_k}{n}$, $InfoGainRatio = \frac{InfoGain_{split}}{SplitINFO}$ (C4.5 algorithm)

Impurity Measures - Misclassification Error

- Misclassification error at t : $Error(t) = 1 - \max_c p(c|t)$; use majority vote
- Maximum at $1 - \frac{1}{C}$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some c
- For two-class classification, $Gini(p) = 2p(1-p)$, $H(p) = -\frac{1}{2}p\log_2 p - (1-p)\log_2(1-p)$ (up to a factor $\frac{1}{2}$), $Error(p) = 1 - \max(p, 1-p)$



Comparing Three Impurity Measures

- Information gain and Gini index are more sensitive to changes in the node probabilities than the misclassification error
- Consider a two-class problem with 400 observations in each class, (400, 400); two possible splits, A : (300, 100) + (100, 300), and B : (200, 400) + (200, 0); B should be preferred
  - $Gini(A) = \frac{1}{2}Gini(A1) + \frac{1}{2}Gini(A2) = 2 \times \frac{1}{2}(2 \times \frac{3}{4} \times \frac{1}{4}) = \frac{3}{8}$, $Gini(B) = \frac{3}{4}Gini(A1) + \frac{1}{4}Gini(A2) = \frac{3}{4}(2 \times \frac{1}{3} \times \frac{2}{3})) = \frac{1}{3}$
  - $H(A) = 2 \times \frac{1}{2}(-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4})) = 0.81$, $H(B) = \frac{3}{4}(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3})) = 0.69$
  - $Error(A) = 2 \times \frac{1}{2}(1 - \max(\frac{3}{4}, \frac{1}{4})) = \frac{1}{4}$, $Error(B) = \frac{3}{4}(1 - \max(\frac{1}{3}, \frac{2}{3})) = \frac{1}{4}$
- Gini index and information gain should be used when growing the tree
- In pruning, all three can be used (typically misclassification error)

Algorithms

- Iterative Dichotomiser 3 (ID3) : by Ross Quinlan (1986), based on Occam's Razor rule (be simple) ; information gain, choose feature values by enumeration
- C4.5 and C5.0 : by R. Quinlan (1993), use information gain ratio instead, choose split thresholds for continuous features
- Classification and Regression Tree (CART) : by Leo Breiman etc. (1984) ; for classification, use Gini index ; for regression, use mean square error ; binary split

算法	属性类型	不纯度度量	分割的子节点数量	目标属性类型
ID3	离散型	信息增益	$k \geq 2$	离散型
C4.5	离散型、连续型	信息增益率	$k \geq 2$	离散型
C5.0	离散型、连续型	信息增益率	$k \geq 2$	离散型
CART	离散型、连续型	GINI指数	$k = 2$	离散型、连续型

ID3 Algorithm

- Input : training set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $Y = \{y_1 \ldots, y_n\}$, set of features $F = \{$column variables of $X = (\mathbf{x}_1 \ldots \mathbf{x}_n)^T\}$
- Output : decision tree T

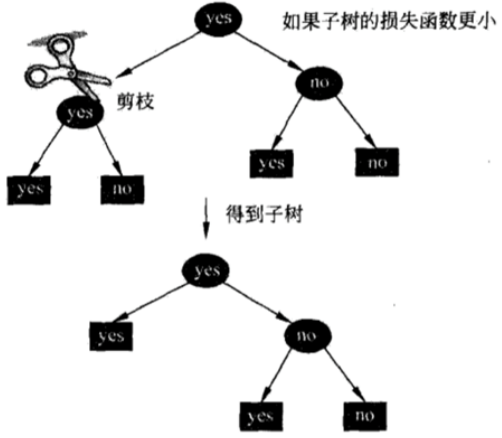
1. Create a root node
2. Check Y : if all are positive, then return a single node tree T with label "+" ; if all are negative, then return a single node tree T with label "-"
3. Check F : if empty, then return a single node tree T with label as majority vote of Y
4. For each feature in F, compute information gain, choose the feature $A \in F$ which maximizes information gain as root
5. For $A = i$, let $D(i) = \{(\mathbf{x}_j, y_j) \in D | x_{jA} = i\}$:
   - 5.1 If $D(i) = \emptyset$, then create a leaf node and make majority vote of D as the label
   - 5.2 Else, let $D = D(i)$, go back to step 1 iteratively

Tree Pruning

- Too complex tree structure easily leads to overfitting
- Prepruning : set threshold δ for impurity decrease in splitting a node ; if $\Delta Impurity_{split} > \delta$, do slitting, otherwise stop
- Postpruning : based on cost function

$$Cost_\alpha(T) = \underbrace{\sum_{t=1}^{|T|} n_t Impurity(t)}_{\text{data fidelity}} + \underbrace{\alpha |T|}_{\text{model complexity}}$$

- Input : a complete tree T, α
- Output : postpruning tree T_α
  1. Compute $Impurity(t)$ for $\forall t$
  2. Iteratively merge child nodes bottom-up : T_A and T_B are the trees before and after merging, do merging if $Cost_\alpha(T_A) \geqslant Cost_\alpha(T_B)$



Pros and Cons

- Advantages
  - Easy to interpret and visualize : widely used in finance, medical health, biology, etc.
  - Easy to deal with missing values (treat as new data type)
  - Could be extended to regression : decision tree is a rectangular partition of the domain, the predictor can be written as

    $f(x) = \sum_{m=1}^{M} c_m \mathrm{I}(\mathbf{x} \in R_m)$; for regression problems

    $c_m = \bar{y}_m = \frac{1}{n_m} \sum_{i=1}^{n} y_i \mathrm{I}(\mathbf{x}_i \in R_m)$ where $n_m = \sum_{i=1}^{n} \mathrm{I}(\mathbf{x}_i \in R_m)$
- Drawbacks :
  - Easy to be trapped at local minimum because of greedy algorithm
  - Simple decision boundary : parallel lines to the axes

Outlines

Introduction

- Based on Bayes Theorem and conditional independency assumption on features
- Widely used in text analysis, spam filtering, recommender systems, and medical diagnosis
- Bayes Theorem : let X and Y be a pair of random variables having joint probability $P(X = x, Y = y)$; by definition, the condition probability of Y given X is $P(Y|X) = \frac{P(X,Y)}{P(X)}$; then by symmetry, $P(X|Y) = \frac{P(X,Y)}{P(Y)}$; upon eliminating $P(X, Y)$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- $P(Y)$ is prior prob. distribution, $P(X|Y)$ is likelihood function, $P(X)$ is evidence, $P(Y|X)$ is posterior prob. distribution

Naive Bayes

- The core problem of machine learning is to estimate $P(Y|X)$ (or its moments $\mathrm{E}[Y|X] = \arg\min_f \mathrm{E}[\|Y - f(X)\|^2]$)
- Let $X = \{X_1, \ldots, X_d\}$, for fixed sample $X = x$, $P(X = x)$ is independent of Y, by Bayes Theorem

$$P(Y|X = x) \propto P(X = x|Y)P(Y)$$

- Assume conditional independency of $X_1, \ldots, X_d$ given $Y = c$:

$$P(X = x|Y = c) = \prod_{i=1}^{d} P(X_i = x_i|Y = c)$$

- Naive Bayes model :

$$\hat{y} = \arg\max_c P(Y = c) \prod_{i=1}^{d} P(X_i = x_i|Y = c)$$

Maximum Likelihood Estimate (MLE)

- Estimate $P(Y = c)$ and $P(X_i = x_i | Y = c)$ from the dataset $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
- MLE for $P(Y = c)$: $P(Y = c) = \frac{\sum_{i=1}^{n} \mathrm{I}(y_i = c)}{n}$
- When X_i is discrete variable with range $\{v_1, \ldots, v_K\}$, MLE for $P(X_i = v_k | Y = c) = \frac{\sum_{i=1}^{n} \mathrm{I}(x_i = v_k, y_i = c)}{\sum_{i=1}^{n} \mathrm{I}(y_i = c)}$
- When X_i is continuous variable
  1. Do discretization, and go back to the above formula
  2. Assume X_i follows some distribution (e.g., $N(\mu, \sigma^2)$) :

$$P(X_i = x | Y = c) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Then use MLE to estimate μ and σ^2

Pros and Cons

- Where it is good
  - Spam filter : compute the posterior prob. distribution of frequently used words (convert to vector by word2vec)
  - Stable : for outliers and miss values
  - Robust : for uncorrelated features; $P(X_i|Y)$ is independent of Y and thus has no effect on posterior probability
  - May outperform far more sophisticated alternatives even if conditional independency assumption is not satisfied
- Disadvantage
  - However, when conditional independency assumption is violated, performance of Naive Bayes can be poorer
  - Depends heavily on how well the parameter estimates are

Outlines

Confusion Matrix

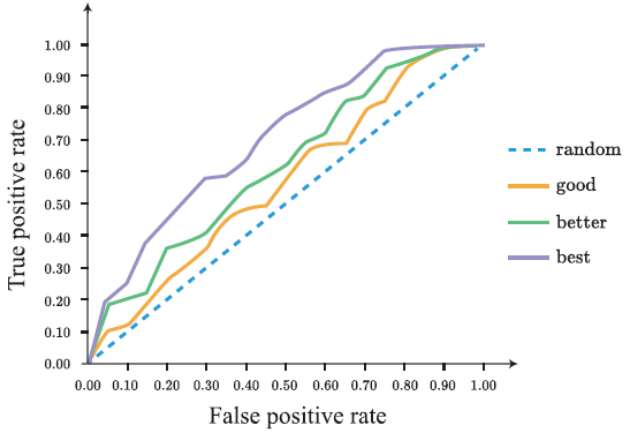
- For two-class classification :
  - True Positive (TP) : both true label and predicted label are positive
  - True Negative (TN) : both true label and predicted label are negative
  - False Positive (FP) : true label is negative, but predicted label is positive
  - False Negative (FN) : true label is positive, but predicted label is negative

真实标签	预测结果	
	1（正例）	0（反例）
1（正例）	TP（真正例）	FN（假反例）
0（反例）	FP（假正例）	TN（真反例）

- $Accuracy = \frac{TP+TN}{TN+FN+FP+TP}$; not a good index when samples are imbalanced
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$; important in medical diagnosis (sensitivity)
- F score : $F_\beta = \frac{(1+\beta^2)Precision \times Recall}{\beta^2 \times Precision + Recall}$; $\beta = 1$, F_1 score
- $Specifity = \frac{TN}{TN+FP}$; recall for negative samples

Receiver Operating Characteristic (ROC) and AUC

- Aim to solve class distribution imbalance problem
- Set different threshold t for continuous predicted values (probability), e.g., if $P(Y = 1|X = x_i) > t$, then $\hat{y}_i = 1$
- Compute TPR ($= \frac{TP}{TP+FN}$, or recall) vs. FPR($= \frac{FP}{FP+TN}$) for different t and plot ROC curve
- The higher the ROC, the better the performance
- AUC : area under ROC, the larger the better, the more robust of the method for the change of t ; very good if > 0.75



Cohen's Kappa Coefficient

- $\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}$ measures the agreement between two raters
- p_o is the accuracy (or the relative observed agreement)
- p_e is the hypothetical probability of chance agreement, $p_e = \sum_{c=1}^{C} \frac{n_c^{pred}}{N} \frac{n_c^{true}}{N}$, where n_c^{pred} is the number of samples predicted in class c, n_c^{true} is the true number of samples in class c, N is the total number of samples
- Eg : $p_o = \frac{20+15}{50} = 0.7$, $p_e = \frac{25}{50} \times \frac{20}{50} + \frac{25}{50} \times \frac{30}{50} = 0.5$, $\kappa = 0.4$

		Predicted Label		
		1	0	Total
True Label	1	20 TP	10 FN	30 C
	0	5 FP	15 TN	20 D
	Total	25 A	25 B	50 N

The Values of Kappa Coefficient

- $\kappa \in [-1, 1]$
- $\kappa = 1$: perfect agreement between two raters
- $\kappa = -1$: completely disagreement
- $\kappa = 0$: no agreement among the raters other than what would be expected by chance
- $\kappa < 0$: worse than random
- $\kappa > 0$: the result is meaningful, agree more as κ gets larger
- $\kappa \geqslant 0.75$: good performance
- $\kappa < 0.4$: bad performance

Multiple Class Problem

- ROC and AUC are not well-defined
- Confusion matrix : $C \times C$, each entry means the number of samples in the intersection of the predicted class i and the true class j
- Positive sample is the sample belonging to the class i, negative sample is the sample not belonging to the class i, so every sample could be positive or negative
- Convert to multiple 0-1 classification problems
- Precision and recall are the averages of that in the each 0-1 classification problem
- $F1$ score is still defined as the harmonic average of precision and recall

Outlines

References

- 数据分析导论，博雅大数据学院
- 周志华，机器学习，2016
- T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning : Data mining, Inference, and Prediction, 2nd Edition, 2009