Introduction to Data Science

Classification and nonlinear models

Zhen Zhang

Southern University of Science and Technology

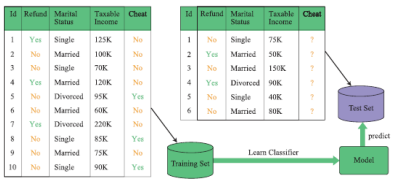
Outlines

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 $y_{pred} = f(\mathbf{x}_{pred})$



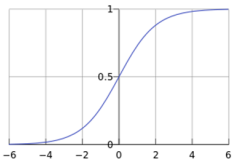
Classification Methods

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

Outlines

Logistic Regression

- Not regression, but a classification method
- Connection with linear regression: $y = w_0 + w_1 x + \epsilon$, y is binary (0 or 1); then $E(y|x) = P(y = 1|x) = w_0 + w_1 x$; but $w_0 + w_1 x$ may not be a probability
- Find a function to map it back to $[0, 1]$: Sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ with $z = w_0 + w_1 x_1 + \ldots + w_d x_d$



- Equivalently, $\log \frac{P(y=1|x)}{1-P(y=1|x)} = w_0 + w_1 x_1 + \ldots + w_d x_d$, logit transform $logit(z) = \log \frac{z}{1-z}$

MLE for Logistic Regression

- The prob. distribution for two-class logistic regression model is

$$Pr(y = 1|\mathbf{X} = \mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{x})}{1 + \exp(\mathbf{w}^T\mathbf{x})},$$
$$Pr(y = 0|\mathbf{X} = \mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^T\mathbf{x})}.$$

- Let $P(y = k|\mathbf{X} = \mathbf{x}) = p_k(\mathbf{x}; \mathbf{w})$, $k = 0$ or 1. The likelihood function is defined by $L(\mathbf{w}) = \prod_{i=1}^{n} p_{y_i}(\mathbf{x}_i; \mathbf{w})$
- MLE estimate of $\mathbf{w}$: $\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} L(\mathbf{w})$
- Solve $\nabla_{\mathbf{w}} \log L(\mathbf{w}) = 0$ by Newton-Raphson method

K-class Logistic Regression

- Extend the relative ratio of probabilities to K-class:

$$\log \frac{P(y = 1|\mathbf{X} = \mathbf{x})}{P(y = K|\mathbf{X} = \mathbf{x})} = \mathbf{w}_1^T\mathbf{x}$$
$$\log \frac{P(y = 2|\mathbf{X} = \mathbf{x})}{P(y = K|\mathbf{X} = \mathbf{x})} = \mathbf{w}_2^T\mathbf{x}$$
$$\vdots$$
$$\log \frac{P(y = K-1|\mathbf{X} = \mathbf{x})}{P(y = K|\mathbf{X} = \mathbf{x})} = \mathbf{w}_{K-1}^T\mathbf{x}$$

- Probabilistic model:

$$P(y = 1|\mathbf{X} = \mathbf{x}) = \frac{e^{\mathbf{w}_1^T\mathbf{x}}}{1 + \sum_{k=1}^{K-1} e^{\mathbf{w}_k^T\mathbf{x}}}$$
$$\vdots$$
$$P(y = K-1|\mathbf{X} = \mathbf{x}) = \frac{e^{\mathbf{w}_{K-1}^T\mathbf{x}}}{1 + \sum_{k=1}^{K-1} e^{\mathbf{w}_k^T\mathbf{x}}}$$

Outlines

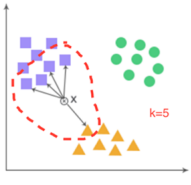
Introduction

- 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 $y_{pred} = \text{Mode}(y_{(1)}, \ldots, y_{(k)})$



Distance Metrics

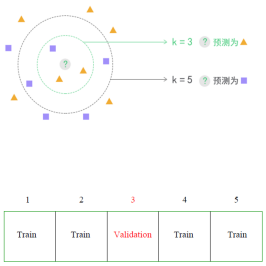
- Minkowski distance: $d_h(\mathbf{x}_1, \mathbf{x}_2) = \sqrt[h]{\sum_{i=1}^{d}(x_{1i} - x_{2i})^h}$; $h = 2$, Euclidean distance; $h = 1$, Manhattan distance
- Mahalanobis distance: $d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \hat{\Sigma}^{-1}(\mathbf{x}_1 - \mathbf{x}_2)}$, 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)

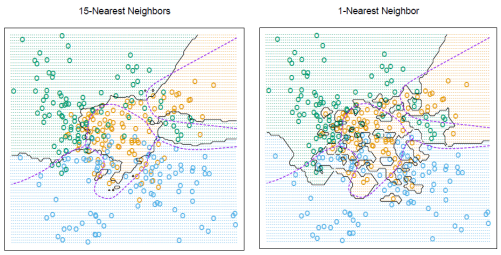
- Assume $Y \in \mathcal{Y} = \{1, 2, \ldots, C\}$, the classifier $f : \mathcal{X} \to \mathcal{Y}$ is a piecewise constant function
- For 0-1 loss $L(y, f)$, the learning problem is to minimize

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

- 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



Analysis

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

kNN Regression: Bias vs. Variance

- kNN can be used to do regression if the mode (majority vote) is replaced by mean: $\hat{f}(\mathbf{x}) = \frac{1}{k}\sum_{\mathbf{x}_{(i)}\in N_k(\mathbf{x})} y_{(i)}$
- Generalization error of kNN regression is

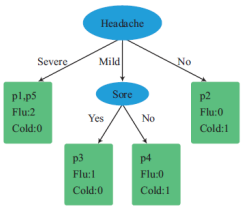
$$\mathrm{E}_{train} R_{exp}(\hat{f}(\mathbf{x})) = \sigma^2 + (f(\mathbf{x}) - \frac{1}{k}\sum_{\mathbf{x}_{(i)}\in N_k(\mathbf{x})} f(\mathbf{x}_{(i)}))^2 + \underbrace{\mathrm{E}_{train}\Big[\frac{1}{k}\sum_{\mathbf{x}_{(i)}\in N_k(\mathbf{x})}(y_{(i)} - f(\mathbf{x}_{(i)}))\Big]^2}_{\frac{1}{k}\sigma^2}$$

  where we have used the fact that $\mathrm{E}_{train} y_i = f(\mathbf{x}_i)$ and $\mathrm{Var}(y_i) = \sigma^2$.
- For small k, overfitting, bias ↘, variance ↗
- For large k, underfitting, bias ↗, variance ↘

Outlines

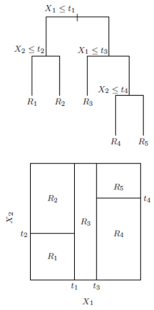
Decision Tree as Medical Diagnosis

- 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



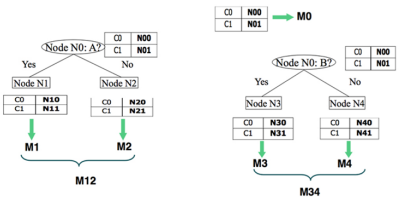
Decision Tree Algorithm

- 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



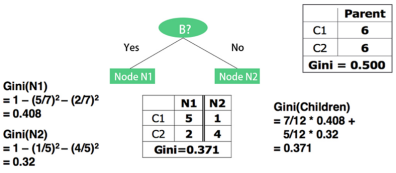
How to choose features and split points

- 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



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)

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
  - $Gini(A) = \frac{1}{2} Gini(A1) + \frac{1}{2} Gini(A2) = 2 \times \frac{1}{2}(2 \times \frac{3}{4} \times \frac{1}{4}) = \frac{3}{8}$, $Gini(B) = \frac{3}{4} Gini(A1) + \frac{1}{4} Gini(A2) = \frac{3}{4}(2 \times \frac{1}{3} \times \frac{2}{3})) = \frac{1}{3}$
  - $H(A) = 2 \times \frac{1}{2}(-\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4})) = 0.81$, $H(B) = \frac{3}{4}(-\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3})) = 0.69$
- Misclassification error at t: $Error(t) = 1 - \max_c p(c|t)$; $Error(A) = 2 \times \frac{1}{2}(1 - \max(\frac{3}{4}, \frac{1}{4})) = \frac{1}{4}$, $Error(B) = \frac{3}{4}(1 - \max(\frac{1}{3}, \frac{2}{3})) = \frac{1}{4}$
- Gini index and information gain should be used when growing the tree

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

Algorithm	Attribute Type	Impurity Measure	# Split Nodes	Target Type
ID3	Discrete	Information Gain	$k \geq 2$	Discrete
C4.5	Discrete, Continuous	Information Gain Ratio	$k \geq 2$	Discrete
C5.0	Discrete, Continuous	Information Gain Ratio	$k \geq 2$	Discrete
CART	Discrete, Continuous	GINI Index	$k = 2$	Discrete, Continuous

Table: Comparison of Different Decision Tree Algorithms

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 $F = \{\text{column variables of } X = (\mathbf{x}_1 \ldots \mathbf{x}_n)^T\}$
- Output: decision tree T

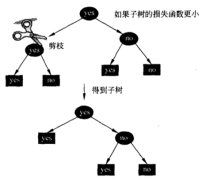
1. Create a root node
2. Check Y: if all are positive, then return a single node tree T with label "+"; if all are negative, then return a single node tree T with label "-"
3. Check F: if empty, then return a single node tree T with label as majority vote of Y
4. For each feature in F, compute information 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

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

- 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

Linear Discriminant Analysis (LDA)

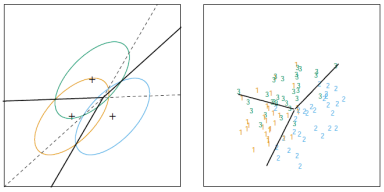
- Bayes Classifier amounts to know the class posteriors $\mathrm{P}(Y|\mathbf{X})$ for optimal classification: $k^* = \arg\max_k \mathrm{P}(Y=k|\mathbf{X})$
- Let $\pi_k = \mathrm{P}(Y=k)$ be the prior probability, $f_k(\mathbf{x}) = \mathrm{P}(\mathbf{X}=\mathbf{x}|Y=k)$ be the density function of samples in each class $Y=k$
- By Bayes theorem, $\mathrm{P}(Y|\mathbf{X}=\mathbf{x}) \propto f_k(\mathbf{x})\pi_k$ (Recall naive Bayes)
- Assume $f_k(x)$ is multivariate Gaussian: $f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T\mathbf{\Sigma}_k^{-1}(\mathbf{x}-\mu_k)}$, with a common covariance matrix $\mathbf{\Sigma}_k = \mathbf{\Sigma}$, sufficient to look at the log-ratio

$$\log\frac{\mathrm{P}(Y=k|\mathbf{X}=\mathbf{x})}{\mathrm{P}(Y=l|\mathbf{X}=\mathbf{x})} = \log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k+\mu_l)^T\mathbf{\Sigma}^{-1}(\mu_k-\mu_l) + \mathbf{x}^T\mathbf{\Sigma}^{-1}(\mu_k-\mu_l)$$

for the decision boundary between class k and l

Discriminant Rule

- Linear discriminant functions: $\delta_k(\mathbf{x}) = \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}^{-1}\mu_k + \log\pi_k$
- Then $\log\frac{\mathrm{P}(Y=k|\mathbf{X}=\mathbf{x})}{\mathrm{P}(Y=l|\mathbf{X}=\mathbf{x})} = \delta_k(\mathbf{x}) - \delta_l(\mathbf{x})$
- Decision rule: $k^* = \arg\max_k \delta_k(\mathbf{x})$
- Sample estimate of unknowns: $\hat{\pi}_k = N_k/N$, where $N = \sum_{k=1}^K N_k$, $\hat{\mu}_k = \frac{1}{N_k}\sum_{y_i=k}\mathbf{x}_i$, $\hat{\mathbf{\Sigma}} = \frac{1}{N-K}\sum_{k=1}^K\sum_{y_i=k}(\mathbf{x}_i-\hat{\mu}_k)(\mathbf{x}_i-\hat{\mu}_k)^T$

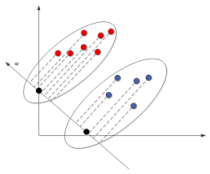


Two-class LDA

- LDA rule classifies to class 2 if

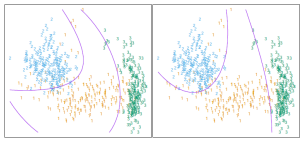
$$(\mathbf{x} - \frac{\hat{\mu}_1+\hat{\mu}_2}{2})^T\hat{\mathbf{\Sigma}}^{-1}(\hat{\mu}_2-\hat{\mu}_1) + \log\frac{\hat{\pi}_2}{\hat{\pi}_1} > 0$$

- Discriminant direction: $\beta = \hat{\mathbf{\Sigma}}^{-1}(\hat{\mu}_2-\hat{\mu}_1)$
- Bayes misclassfication rate $= 1-\Phi(\beta^T(\mu_2-\mu_1)/(\beta^T\mathbf{\Sigma}\beta)^{\frac{1}{2}})$, where $\Phi(x)$ is the Gaussian distribution function



Other Variants

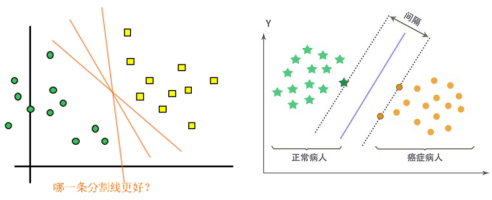
- Quadratic discriminant analysis (QDA): $\delta_k(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{\Sigma}_k| - \frac{1}{2}(\mathbf{x}-\mu_k)^T\mathbf{\Sigma}_k^{-1}(\mathbf{x}-\mu_k) + \log\pi_k$
- Regularized discriminant analysis: $\hat{\mathbf{\Sigma}}_k(\alpha) = \alpha\hat{\mathbf{\Sigma}}_k + (1-\alpha)\hat{\mathbf{\Sigma}}$
- Computations for LDA:
  1. Sphere the data with respect to $\hat{\mathbf{\Sigma}} = \mathbf{UDU}^T$: $\mathbf{X}^* = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{X}$. Then the common covariance estimate of $\mathbf{X}^*$ is $\mathbf{I}_p$
  2. Classsify to the closest class centroid in the transformed space, taking into account of the class prior probabilities π_k's
- Reduced-Rank LDA: see dimensionality reduction



Outlines

Support Vector Machine (SVM)

- Use hyperplane to separate data: maximize margin
- Can deal with low-dimensional data that are not linearly separated by using kernel functions
- Decision boundary only depends on some samples (support vectors)



Linear SVM

- Training data: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{-1, +1\}$
- Hyperplane: $S = \mathbf{w}^T\mathbf{x} + b$; decision function: $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

$$\left.\begin{array}{l} f(\mathbf{x}_i) > 0 \Leftrightarrow y_i = 1 \\ f(\mathbf{x}_i) < 0 \Leftrightarrow y_i = -1 \end{array}\right\} \Rightarrow y_i f(\mathbf{x}_i) > 0$$

- Geometric margin between a point and hyperplane: $r_i = \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|_2}$
- Margin between dataset and hyperplane: $\min\limits_i r_i$
- Maximize margin: $\max\limits_{\mathbf{w},b} \min\limits_i \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|_2}$

Formulation as Constrained Optimization

- Without loss of generality, let $\min\limits_i y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$ (multiply $\mathbf{w}$ and b by the same proper constant)
- Maximize margin is equivalent to

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2}, \quad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, i = 1, \ldots, n$$

- Further reduce to

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2, \quad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, i = 1, \ldots, n$$

- This is primal problem: quadratical programming with linear constraints, computational complexity is $O(p^3)$ where p is dimension

Method of Lagrange Multipliers

- Introduce $\alpha_i \geqslant 0$ as Lagrange multiplier of constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1$
- Lagrange function:
  $L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum\limits_{i=1}^{n} \alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$
- Since

$$\max_{\alpha} L(\mathbf{w}, b, \alpha) = \begin{cases} \frac{1}{2}\|\mathbf{w}\|_2^2, & y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geqslant 0, \forall i \\ +\infty, & y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 < 0, \exists i \end{cases}$$

- Primal problem is equivalent to the minimax problem

$$\min_{\mathbf{w},b} \max_{\alpha} L(\mathbf{w}, b, \alpha)$$

Dual problem

- When slater condition is satisfied, $\min\max \Leftrightarrow \max\min$
- Dual problem: $\max\limits_{\alpha} \min\limits_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)$
- Solve for inner minimization problem:

$$\nabla_{\mathbf{w}} L = 0 \Longrightarrow \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = 0 \Longrightarrow \sum_i \alpha_i y_i = 0$$

- Plug into L: $L(\mathbf{w}^*, b^*, \alpha) = \sum\limits_i \alpha_i - \frac{1}{2}\sum\limits_i \sum\limits_j \alpha_i \alpha_i y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$
- Dual optimization:

$$\min_{\alpha} \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) - \sum_i \alpha_i,$$

$$\text{s.t. } \alpha_i \geqslant 0, i = 1, \ldots, n, \sum_i \alpha_i y_i = 0$$

KKT conditions

- Three more conditions from the equivalence of primal and minimax problems

$$\begin{cases} \alpha_i^* \geqslant 0, \\ y_i((\mathbf{w}^*)^T\mathbf{x}_i + b^*) - 1 \geqslant 0, \\ \alpha_i^*[y_i((\mathbf{w}^*)^T\mathbf{x}_i + b^*) - 1] = 0. \end{cases}$$

- These together with two zero derivative conditions form KKT conditions
- $\alpha_i > 0 \Rightarrow y_i(\mathbf{w}^T\mathbf{x}_i + b^*) = 1$
- Index set of support vectors $S = \{i | \alpha_i > 0\}$
- $b = y_s - \mathbf{w}^T\mathbf{x}_s = y_s - \sum\limits_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s$
- More stable solution: $b = \frac{1}{|S|}\sum\limits_{s \in S}\left(y_s - \sum\limits_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s\right)$

Sequential Minimal Optimization (SMO) Algorithm

- Invented by John C. Platt (1998)
- Coordinately optimize dual problem, select two variables and fix others, then dual problem reduces to one variable quadratic programming with positivity constraint
  1. Initially, choose α_i and α_j
  2. Fix other variables, solve for α_i and α_j
  3. Update α_i and α_j, redo step 1 iteratively
  4. Stop until convergence
- How to choose α_i and α_j? choose the pair far from KKT conditions the most
- Computational complexity $O(n^3)$
- Easy to generalize to high dimensional problem with kernel functions

Soft Margin

- When data are not linear separable, introduce slack variables (tolerance control of fault) $\xi_i \geqslant 0$
- Relax constraint to $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i$
- Primal problem:

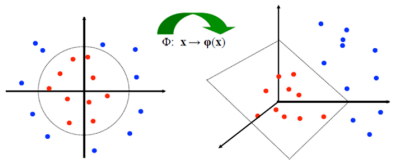
$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i, \xi_i \geqslant 0, i = 1, \ldots, n$$

- Similar derivation to dual problem:

$$\min_{\alpha} \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j(\mathbf{x}_i^T\mathbf{x}_j) - \sum_i \alpha_i,$$
$$\text{s.t. } 0 \leqslant \alpha_i \leqslant C, i = 1, \ldots, n, \sum_i \alpha_i y_i = 0$$

Nonlinear SVM

- Nonlinear decision boundary could be mapped to linear boundary in high-dimensional space
- Modify objective function in dual problem: $\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j(\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)) - \sum_i \alpha_i$
- Kernel function as inner product: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$



Kernel Methods

- Reduce effect of curse of dimensionality
- Different kernels lead to different decision boundaries
- Popular kernels:

Kernel	Definition	Parameters
Polynomial	$(\mathbf{x}_1^T\mathbf{x}_2 + 1)^d$	d is positive integer
Gaussian	$e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\delta^2}}$	$\delta > 0$
Laplacian	$e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{\delta^2}}$	$\delta > 0$
Fisher	$\tanh(\beta\mathbf{x}_1^T\mathbf{x}_2 + \theta)$	$\beta > 0, \theta < 0$

Pros and Cons

- Where it is good
  - Applications in pattern recognition: text classification, face recognition
  - Easy to deal with high-dimensional data with kernels
  - Robust (only depends on support vectors), and easy to generalize to new dataset
- Disadvantage
  - Low computational efficiency for nonlinear SVM when sample size is large
  - Poor interpretability without probability

Outlines

Confusion Matrix (Two-class)

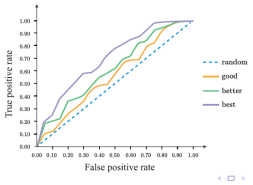
- True Positive (TP): both true label and predicted label are positive
- True Negative (TN): both true label and predicted label are negative
- False Positive (FP): true label is negative, but predicted label is positive
- False Negative (FN): true label is positive, but predicted label is negative

- *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

True Label	Prediction Result	
	1 (Positive Instance)	0 (Negative Instance)
1 (Positive Instance)	TP (True Positive)	FN (False Negative)
0 (Negative Instance)	FP (False Positive)	TN (True Negative)

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



Other metrics

- Cohen's Kappa Coefficient $\kappa \in [-1, 1]$: as large as possible
- Multiple Classes 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

Cohen's Kappa Coefficient (Optional)

- $\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}$ 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

Outlines

References

- 机器学习，周志华，2016.
- Chapters 4, 8, 9, An Introduction to Statistical Learning with Applications in Python by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani and Jonathan Taylor, Springer, 2023.
- Chapters 4, 9, 12-13, 15, The Elements of Statistical Machine Learning: Data mining, Inference and Prediction by Trevor Hastie, Robert Tibshirani, and Jerome Friedman, Springer, 2009.