Kernel Density Estimation

If $Y \in \{1, \ldots, K\}$ and g_k denotes the density for the conditional distribution of X given $Y = k$ the Bayes classifier is

$$f(x) = \operatorname*{argmax}_k \pi_k g_k(x)$$

If $\hat{g}_k$ for $k = 1, \ldots, K$ are density estimators – non-parametric kernel density estimators, say – then using the plug-in principle

$$\hat{f}(x) = \operatorname*{argmax}_k \hat{\pi}_k \hat{g}_k(x)$$

is an estimator of the Bayes classifier.

This is the non-parametric version of LDA.

Naive Bayes

High-dimensional kernel density estimation suffers from the curse of dimensionality.

Assume that the X-coordinates are independent given the Y, then

$$g_k(x) = \prod_{i=1}^{p} g_{k,i}(x_i)$$

with $g_{k,i}$ univariate densities.

$$\begin{aligned}
\log \frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)} &= \log \frac{\pi_k}{\pi_K} + \log \frac{g_k(x)}{g_K(x)} \\
&= \log \frac{\pi_k}{\pi_K} + \sum_{i=1}^{p} \underbrace{\log \frac{g_{k,i}(x_i)}{g_{K,i}(x_i)}}_{h_{k,i}(x)} \\
&= \log \frac{\pi_k}{\pi_K} + \sum_{i=1}^{p} h_{k,i}(x)
\end{aligned}$$

Naive Bayes – Continued

The conditional distribution above is an example of a generalized additive model. Estimation of $h_{k,i}$ using univariate (non-parametric) density estimators $\hat{g}_{k,i}$;

$$\hat{h}_{k,i} = \log \frac{\hat{g}_{k,i}(x_i)}{\hat{g}_{K,i}(x_i)}$$

is known as naive – or even idiot's – Bayes.

Naive Bayes – Discrete Version

If some or all of the X variables are discrete, univariate kernel density estimation can be replaced by appropriate estimation of point probabilities.

If all X_i take values in $\{a_1, \ldots, a_n\}$ the extreme implementation of naive Bayes is to estimate

$$\hat{g}_{k,i}(r) = \frac{1}{N_k} \sum_{j:y_j=k} 1(x_{ji} = a_r), \quad N_k = \sum_{j=1}^{N} 1(y_j = k).$$

This is a possible solution procedure for prac 7.

Generalized Additive Models

A generalized additive model of Y given X is given by a link function g such that the mean $\mu(X)$ of Y given X is

$$g(\mu(X)) = \alpha + f_1(X_1) + \ldots + f_p(X_p).$$

This is an extension from general linear models by allowing for non-linear but univariate effects given by the f_i-functions.

The functions are not in general identifiable – and we can face a problem similar to collinearity, which is known as concurvity.

Generalized Additive Logistic Regression

An important example arise with $Y \in \{0, 1\}$ with the logit link

$$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right), \quad \mu \in (0, 1)$$

Then

$$\mu(X) = \Pr(Y = 1|X) = \frac{\exp(\alpha + f_1(X_1) + \ldots + f_p(X_p))}{1 + \exp(\alpha + f_1(X_1) + \ldots + f_p(X_p))}$$

Like logistic regression we can use other link functions like the probit link

$$g(\mu) = \Phi^{-1}(\mu)$$

where Φ is the distribution function for the normal distribution.

Penalized Estimation

The general, penalized minus-log-likelihood function is

$$l_N(\alpha, f_1, \ldots, f_p) + \sum_{j=1}^{p} \lambda_j \int_a^b f_j''(x)\mathrm{d}x$$

with tuning parameters $\lambda_1, \ldots, \lambda_p$. The minimizer, if it exists, consists of natural cubic splines. For identification purposes we assume

$$\sum_{i=1}^{N} f_j(x_{ij}) = 0, \quad j = 1, \ldots, p.$$

This is equivalent to $\mathbf{f}_j = (f_j(x_{1j}), \ldots, f_j(x_{Nj}))^T$ being perpendicular to the column vector $\mathbf{1}$ for $j = 1, \ldots, p$. The penalization resolves overparameterization problems for the non-linear part but not the linear part of the fit.

Informal Tests for Non-Linear Effects

Using smoothing splines there is to each estimated function $\hat{f}_j$ an associated linear smoother matrix $\mathbf{S}_j$.

The effective degress of freedom for the non-linear part of the fit is

$$\mathrm{df}_j = \mathrm{trace}(\mathbf{S}_j) - 1$$

Implementations often do ad hoc χ^2-tests for the non-linear part using χ^2-distributions with df_j degress of freedom – these test are at best justified by approximations and simulation studies, and can be used as guidelines only.

Spam Email Classification

Whether an email is a spam email or a regular email is a great example of a problem where prediction is central and interpretation is secondary.

The (simplistic) example in the book deals with 4601 emails to an employee at Hewlett-Packard.

Each email is dimension reduced to a 57-dimensional vector containing

- Quantitative variables of word or special character percentages.
- Quantitative variables describing the occurrence of capital letters.

Figure 9.1 – Non-linear Email Spam Predictor Effects

CART – Classification and Regression Trees

Trees can be viewed as basis expansions of simple functions

$$f(x) = \sum_{m=1}^{M} c_m 1(x \in R_m)$$

with $R_1, \ldots, R_m \subseteq \mathbb{R}^p$ disjoint.

The CART algorithm is a heuristic, adaptive algorithm for basis function selection.

A recursive, binary partition (a tree) is given by a list of splits

$$\{(t_{01}), (t_{11}, t_{12}), (t_{21}, t_{22}, t_{23}, t_{24}), \ldots, (t_{n1}, \ldots, t_{n2^n})\}$$

and corresponding split variable indices

$$\{(i_{01}), (i_{11}, i_{12}), (i_{21}, i_{22}, i_{23}, i_{24}), \ldots, (i_{n1}, \ldots, i_{n2^n})\}$$

$$R_1 = (x_{i_{01}} < t_{01}) \cap (x_{i_{11}} < t_{11}) \cap \ldots \cap (x_{i_{n1}} < t_{n1})$$

and we can determine if $x \in R_1$ in n steps $\ll M = 2^n$.

Figure 9.2 – Recursive Binary Partitions

The recursive partition of $[0, 1]^2$ above and the representation of the partition by a tree.

A binary tree of depth n can represent up to 2^n partitions/basis functions.

We can determine which R_j an x belongs to by n recursive yes/no questions.

Figure 9.2 – General Partitions

A general partition that can not be represented as binary splits.

With M sets in a general partition we would in general need of the order M yes/no questions to determine which of the sets an x belongs to.

Figure 9.2 – Recursive Binary Partitions

For a fixed partition $R_1, \ldots, R_M$ the least squares estimates are

$$\hat{c}_m = \bar{y}(R_m) = \frac{1}{N_m} \sum_{i : x_i \in R_m} y_i$$

$$N_m = |\{i \mid x_i \in R_m\}.$$

The recursive partition allows for rapid computation of the estimates and rapid predition of new observations.

Greedy Splitting Algorithm

With squared error loss and an unknown partition $R_1, \ldots, R_M$ we would seek to minimize

$$\sum_{i=1}^{N} (y_i - \bar{y}(R_{m(i)}))^2$$

over the possible binary, recursive partitions. But this is computationally difficult.

An optimal single split on a region R is determined by

$$\min_j \min_s \underbrace{\left(\sum_{i:x_i \in R(j,s)} (y_i - \bar{y}(R(j,s)))^2 + \sum_{i:x_i \in R(j,s)^c} (y_i - \bar{y}(R(j,s)^c))^2 \right)}_{\text{univariate optimization problem}}$$

with $R(j,s) = \{x \in R \mid x_j < s\}$ The tree growing algorithm recursively does single, optimal splits on each of the partitions obtained in the previous step.

Tree Pruning

The full binary tree, T_0, representing the partitions $R_1, \ldots, R_M$ with $M = 2^n$ may be too large. We prune it by snipping of leafs or subtrees.

For any subtree T of T_0 with $|T|$ leafs and partition $R_1(T), \ldots, R_{|T|}(T)$ the cost-complexity of T is

$$C_\alpha(T) = \sum_{i=1}^{N} (y_i - \bar{y}(R_{m(i)}(T)))^2 + \alpha |T|.$$

Theorem

There is a finite set of subtrees $T_0 \supseteq T_{\alpha_1} \supset T_{\alpha_2} \supset \ldots \supset T_{\alpha_r}$ *with* $0 \leq \alpha_1 < \alpha_2 < \ldots < \alpha_r$ *such that* T_{α_i} *minimizes* $C_\alpha(T)$ *for* $\alpha \in [\alpha_i, \alpha_{i+1})$

Node Impurities and Classification Trees

Define the node impurity as the average loss for the node R

$$Q(R) = \frac{1}{N(R)} \sum_{i:x_i \in R} (y_i - \bar{y}(R))^2$$

The greedy split of R is found by

$$\min_j \min_s \left(N(R(j,s))Q(R(j,s)) + N(R(j,s)^c)Q(R(j,s)^c) \right)$$

with $R(j,s) = \{x \in R \mid x_j < s\}$ and we have

$$C_\alpha(T) = \sum_{m=1}^{|T|} N(R_m(T))Q(R_m(T)) + \alpha|T|.$$

If Y takes K discrete values we focus on the node estimate for $R_m(T)$ in tree T as being

$$\hat{p}_m(T)(k) = \frac{1}{N_m} \sum_{i:x_i \in R_m(T)} 1(y_i = k)$$

Node Impurities and Classification Trees

The loss functions for classification enter in the specification of the node impurities used for splitting an cost-complexity computations.

Examples

- 0-1 loss gives misclassification error impurity:

$$Q(R_m(T)) = 1 - \max\{\hat{p}(R_m(T))(1), \ldots, \hat{p}(R_m(T))(K)\}$$

- likelihood loss gives entropy impurity:

$$Q(R_m(T)) = -\sum_{k=1}^{K} \hat{p}(R_m(T))(k) \log \hat{p}(R_m(T))(k)$$

- The Gini index impurity:

$$Q(R_m(T)) = \sum_{k=1}^{K} \hat{p}(R_m(T))(k)(1 - \hat{p}(R_m(T))(k))$$

Figure 9.3 – Node Impurities

Spam Example

Spam Example

Sensitivity and Specificity

The sensitivity is the probability of predicting 1 given that the true value is 1 (predict a case given that there is a case).

$$\begin{aligned} \text{sensitivity} &= \Pr(f(X) = 1 | Y = 1) \\ &= \frac{\Pr(Y = 1, f(X) = 1)}{\Pr(Y = 1, f(X) = 1) + \Pr(Y = 1, f(X) = 0)} \end{aligned}$$

The specificity is the probability of predicting 0 given that the true value is 0 (predict that there is no case given that there is no case).

$$\begin{aligned} \text{specificity} &= \Pr(f(X) = 0 | Y = 0) \\ &= \frac{\Pr(Y = 0, f(X) = 0)}{\Pr(Y = 0, f(X) = 0) + \Pr(Y = 0, f(X) = 1)} \end{aligned}$$

ROC curves – reciever operating characteristic