3. Linear regression

3.1. Simple linear regression.

$Y \approx \beta_0 + \beta_1 X$ (3.1)

$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ (3.2) coefficient / parameter.

least squares: choosing $\hat{\beta}_0, \hat{\beta}_1$ to minimize RSS.

RSS = residual $= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

$$RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \dots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2 \quad (3.3)$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad (3.4)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i,\ \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

3.1.2 Accuracy of coefficients estimate

true relation: $Y = \beta_0 + \beta_1 X + \varepsilon$ (3.5)

$Y = 2 + 3X + \varepsilon$ (3.6) Simulated.

How accurate is the sample mean $\hat{\mu}$ as an estimate of μ?

$$\mathrm{Var}(\hat{\mu}) = SE(\hat{\mu})^2 = \frac{\sigma^2}{n} \quad (3.7)$$

$SE(\hat{\beta}_0)^2$, $SE(\hat{\beta}_1)^2$ (3.8)

$\sigma^2 = \mathrm{Var}(\varepsilon)$ residual standard error

$RSE = \sqrt{RSS/(n-2)}$ ~~RSE~~ → estimate of population var using sample.

95% confidence intervals: $\hat{\beta}_1 \pm 2 \cdot SE(\hat{\beta}_1)$ (3.9)

Hypothesis testing:

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)} \quad (3.14)$$

3.1.3 Accuracy of the Model.

- RSE: residual ~~squared~~ standard error
- R^2

RSE:
- an estimate of standard deviation of ε.
- Average amount that the response deviate from the true regression line.
- An absolute measure of lack off of fit of model to the data.

$$RSE = \sqrt{\frac{1}{n-2} RSS} = \sqrt{\frac{1}{n-2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (3.15)$$

R^2: proportion of variability in Y that can be explained ~~by~~ using X.

$$R^2 = 1 - \frac{RSS}{TSS} \quad (3.17)$$

$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$ total sum of squares.

R^2: a measure of linear relationship between X and Y.

$$\mathrm{Cor}(X, Y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \quad (3.18)$$

$\mathrm{Cor}(X,Y) = r$

$\boxed{R^2 = r^2}$ for simple linear regression.

3.2 Multiple Linear Regression

p distinct preditors

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots \beta_p X_p + \varepsilon \quad (3.19)$$

3.2.1 Estimating the regression coefficients

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \dots \hat{\beta}_p X_p \quad (3.21)$$

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (3.22)$$

F statistics: at least one ~~factor~~ predictor is effective.

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

F-distribution → p value

- individual p value: report the partial effect of adding that variable to the model.

ch.6 feature selection

- forward selection: starting with null, add variables, find lowest RSS.
- Backward selection: starting with all. remove variable with largest p-value.
- Mixed selection: add variable, which has best fit, remove variable with p-value greater than threshold.

$$R^2 = Cor(Y, \hat{Y})^2$$

correlation between response and fitted model.

$$RSE = \sqrt{\frac{1}{n-p-1} RSS} \quad (3.25)$$

confidence interval: reducible

prediction interval = irr + re

- hierachical principles

if we include the interaction term, we should also include the main effects, even if p-value is large.

- correlated ε: give unwanted confidence.
- leverage statistics: outlier: increase RSS, reduce R^2; high leverage points: affect lsf.
- non-linearity: reduce prediction accuracy.
- Collinearity: VIF — Variance inflation factor.

reduce estimate accuracy of coefficients, cause $\hat{\beta}_j \uparrow$

4. Classification

probability → regression

- logistic regression
- linear discriminant analysis
- k-nearest neighbors,

$X\hat{\beta}$ is an estimate of $Pr(G \mid X)$ ...

4.3 Logistic regression

the propability that response Y belongs to a particular category.

To make $p \in [0,1]$

logistic function: $$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \quad (4.2)$$

To fit this model, we use maximum likelihood.

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X} \quad (4.3)$$

odds

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0 + \beta_1 X \quad (4.4)$$

$$l(\beta_0, \beta_1) = \prod_{i: y_i = 1} p(x_i) \prod_{i': y_{i'} = 0} [1 - p(x_{i'})]$$

4.3.2 Estimating the coefficients.

$$\mathcal{L}(\beta_0,\beta_1)=\prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0}(1-p(x_i))$$

* Use gradient descent to solve for coefficients.

multi-class classifiation — use linear discriminate analysis.

4.4. Linear discriminate analysis.

①. Model the distributions of X seperately in each response class (Y), then use Baye's theorem to flip these around into estimates for $Pr(Y=k|X=x)$.

- class are well-seperated.
- n is small, X is normal
- more than 2 classes.

π_K = prior ~~[illegible]~~ probability that a random observation comes from k.

$f_k(x) \equiv Pr(X=x|Y=k)$ density function of X given k.

$$Pr(Y=k|X=x)=\frac{f_k(x)\pi_k}{\sum_{l=1}^{K}\pi_l f_l(x)} \quad (4.10)$$

linear in X.

assume $f_k(x)$ to be Gaussian distributed,

$$\Rightarrow \hat{\delta}_k(x)=x\cdot\frac{\hat{\mu}_k}{\hat{\sigma}^2}-\frac{\hat{\mu}_k^2}{2\hat{\sigma}^2}+\log(\hat{\pi}_k)$$

Multivariate Gaussian:

$$f(x)=\frac{1}{(2\pi)^{p/2}|\Sigma|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \quad (4.18)$$

$$\hat{\delta}_k(x)=x^T\Sigma^{-1}\mu_k-\frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k+\log(\pi_k)$$

· logistic = model directly $P(Y=y|X=x)$

LDA: first get distribution of X for each class k, then use Bayes' theorem to flip it. (try to estimate Bayes classifier).

by estimating: $\hat{\pi}_k$, $\hat{f}_k(x)$, $\hat{\sigma}^2$

credit card company: low tolerance for false negtive, (0.5)

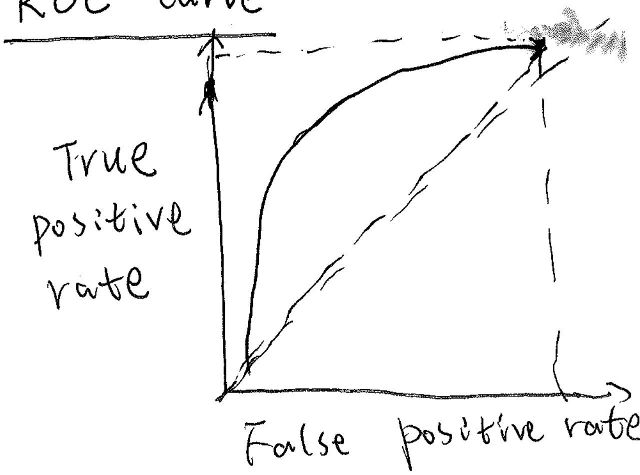
(~~treat~~ default as non-default).

lower the threshold $Pr(\text{default}|X=x) \geq 0.2$.

· LDA vs QDA (Σ_k for each k)

bias-variance tradeoff.

ROC curve



5. Resampling method.

~~Repetedly~~ Repetedly draw samples from a training set and refit a model of interest on each sample in order to ~~estimate the variability of a linear regression~~ obtain additional information about the fitted model.

- cross-validtion : measure error rate / ~~select~~ select flexibility
- bootstrap : accuracy of parameter estimate.

5.1 Cross validation.

Validation set

~~LOOCV~~

high variability

half amount of data — overestimate test MSE

(highly correlated)
LOOCV — non-biased, high variance, high computational expense.

(5,10)
k-fold — moderate bias and variance

(may underestimate test MSE)

~~5.2~~

5.2 Bootstrap.

widely applicable

quantify uncertainty with a statistical method.

- Repeatly sampling observations from the orignal data set to emulate sampling from the population.

6. Linear model selection and regularization.

6.1. Subset Selection

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots \beta_p X_p + \varepsilon \quad (6.1)$$

least squares

↓ improve linear models using alternative fitting approaches.

↓ for better prediction accuracy and model-interpretability

By constraning ~~and~~ or shrinking the estimated coefficients, substantiall reduce variance at a cost of negligible increase in bias

~~generalize to~~

~~deviance~~ ↓RSS

deviance : $-2 \log \mathcal{L}$

$$C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2) \quad (6.2)$$

— if $\hat{\sigma}^2$ is unbiased, then

C_p is an unbiased estimator of test MSE.

$$AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2), \text{ for least squares.}$$

$$BIC = \frac{1}{n\hat{\sigma}^2}(RSS + \log(n) d\hat{\sigma}^2) \quad (6.3)$$

$$\text{Adjusted } R^2 = 1 - \frac{RSS(n-d-1)}{TSS(n-1)} \quad (6.4)$$

(foward/ backward) (stepwise

- subset selection : computationally expensive.
- shrinkage.
- Dimension reduction.

6.2. Shrinkage method.

- ridge regression. (biased)
- Lasso

elastic net: $\lambda \sum_{j=1}^{p} [(1-\alpha)\beta_j^2 + \alpha|\beta_j|]$.

least squares fitting

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij}\right)^2.$$

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \quad ①$$

minimize ① subject to $\sum_{j=1}^{p} \beta_j^2 \le s$

now λ — tuning parameter.

change of scale affects the estimation of coefficients...

Standardize:

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}} \quad (6.6)$$

Ridge > least squares is rooted in bias-variance tradeoff.

- works best in situations LS has high variance.

$\lambda \uparrow$ less flexibility, less variance, more bias,

6.2.2 Lasso

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j| \quad (6.7)$$

- minimise (6.7) subject to $\sum_{j=1}^{p} |\beta_j| \le s$.

6.2.3 Choosing tuning parameter λ.

Use cross-validation.

which gives the smallest cross-validation error.

6.3 Dimension reduction methods.

using transformed variables

$$Z_m = \sum_{i=1}^{p} \phi_{jm} X_j \quad (6.16)$$

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \varepsilon_i, \; i=1,2,\dots,n \quad (6.17)$$

6.3.1. $\boxed{\text{PCA}}$ $\boxed{\text{unsupervised}}$

M

- standalize before constructing PC.

6.3.2. partial least squares

$\boxed{\text{PLS}}$ $\boxed{\text{supervised}}$

- standlize before implementing.

PCR assumption: the directions in which $X_1, X_2, \dots, X_p$ show the most variation are the directions that are associated with Y.

ridge regression : continous version of PCR.

①. $Z_1 = \sum_{j=1}^{p} \phi_{j1} X_j$

simple linear regression coefficients.

② regessing each variable on Z_1 and take residuals, repeat ① on these residuals.

can reduce bias, but increase ~~can~~ variance.

6.4 High dimensions

$\boxed{\text{features related to response}}$

Cp, AIC, BIC, adjusted R^2 not applicable, since $\hat{\sigma}^2 = 0$

~~not accurate~~ problematic.

7. Beyond linearity.

- polynomial regression
- step functions.
- splines
- basis function
- local regression (dimension reduction?)
- generalized additive models.

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \dots \beta_K b_K(x_i) + \varepsilon_i \quad (7.7)$$

7.4 Regression splines.

Smoothing splines:

$$\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt \quad (7.11)$$

$$y_i = \beta_0 + \sum_{i=1}^{p} f_j(x_{ij}) + \varepsilon_i$$

$$= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \dots f_p(x_{ip}) + \varepsilon_i \quad (7.15)$$

trunked power basis function per knot;

$$h(x, \xi) = \begin{cases} (x-\xi)^3, & x > \xi \\ 0, & \text{else} \end{cases}$$

- natural spline: linear at boundary

shrink compared to basis function approach.

λ controls effective degrees of freedom.

effective degrees of freedom:

$$\hat{g}_\lambda = S_\lambda y$$

$$df_\lambda = \sum_{i=1}^{n} \{S_\lambda\}_{ii}$$

Comparison

- regression splines: choose number of knots, fit to minimize RSS and require continuity (y, y', y'').
- a knot at each observation.

$$\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2$$

minimize, with df shrikun by λ.

7.6 Local regression. (Compared to KNN not good in high dimension.)

7.7 GAM. (Con additivity)

8. Tree-Based methods.

- stratifying or segmenting the predictor space into a number of simple regions. - decision tree.
- bagging, random forests, boosting produce multiple trees combined to generate a single consensus prediction.
- recursive binary splitting:
  - greedy
  - only split one region each step.

minimize $\sum_{i=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$

- To avoid overfitting. Prune the tree.

① choose subtree which has smallest:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \quad (8.4)$$

② Use cross-validation to find minimum test error w.r.t α.

classification tree:

instead of minimizing error rate, use

- Gini index:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \quad (8.6)$$

- entropy:

$$D = - \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

8.2 Bagging, random forests and Boosting.

- Bagging: $\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$.

help to find most important predictors.

8.2.2 Random forests

- decorrelate the trees.

using $m = \sqrt{p}$ predictors at each split.

8.2.3 Boosting.

- Growing trees sequentially...

learns slowly, fit the residuals.

① $\hat{f}(x) = 0$, $r_i = y_i$

$$\begin{cases} \hat{f}(x) \leftarrow \hat{f}(x_i) + \lambda \hat{f}^b(x_i) \\ r_i \leftarrow r_i - \lambda \hat{f}^b(x) \end{cases}$$

② $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$.

- large $B \to$ overfitting, use CV to choose B.

- λ: learning rate.

9. Support vector machine.

- maximal margin classifier. (linear boundary)
- support vector classifier
- support vector machine.

hyperplane: $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots \beta_p X_p = 0$.

support vectors: margin observations.

$y_i(\beta_0 + \beta_1 x_1 + \cdots \beta_p X_p) \geq M$ (Maximize)

subject to $\sum \beta_j^2 = 1$

soft margin $\to$ support vector classifier.

9.2 Support vector classifier

maximize M

subject to $\sum_j \beta_j^2 = 1$

$y_i(\beta_0 + \beta_1 X_1 + \cdots \beta_p X_p) \geq M(1 - \varepsilon_i)$

$\varepsilon_i \geq 0$, $\sum_{i=1}^{n} \varepsilon_i \leq C$.

- slack variable: ~~also~~ allow some observations to be on the wrong side of the hyperplane.

$\varepsilon_i > 0$: violate the margin

$\varepsilon_i > 1$: wrong side of hyperplane.

Only support vectors (on or violate margin) can affect classifier!

9.3 Support vector machines.

SVM: enlarge feature space with efficient computations.

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(X, X_i)$$

8. Tree-based Methods.

bagging, random forests &
boosting.

- construct the regions

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \quad (8.1)$$

classification:

① classification error rate

② Gini index:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \quad (8.6)$$

③ entropy.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

node purity.

8.2.2 Random forests.

m predictors $m \sim \sqrt{p}$.

decorrelating.

- not overfit with large B.
- better for correlated predictors.

boosting of decision trees.

B, d, λ

learn slowly

9. Support vector machine.

— Classification.

9.1 maximal margin classifier.

hyperplane: 2-d

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \quad (9.1)$$

Support vectors: on the margin or violate the margin.

Support vector machine

kernal:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(X, X_i)$$

$K(x_i, x_{i'})$ polynomial kernal
radial kernal

9.5 relation to logistic regression

- Similar loss functions

well seperated : choose SVM
much ovelap : choose logistic

10.

10. Unsupervised learning.

proportion of variance explained (PVE)

scree plot.

10.3 Clustering.

PCA: a low-dimentional representation that could ~~exp~~ ~~have a good explan~~ explain a good fraction of variance.

Clustering: find homogeneous subgroups

K means clustering.
hierachical clustering.
(dendrogram)

- minimize ~~total~~ in-cluster variation.

K means:
- assign randomly observations to 1 to K,
- centroid as means of clusters.
- assign observation to closest centroid.

10

10.3.2 hierachiccal clustering.

Kinds of measuring dissimilarity
Kinds of dissimilarity.
linkage

issues
- standardize.

K means clustering.

How to choose K:

①. "Elbow ~~method~~ method"

plot RSS vs K, if we observe ~~we expect~~ that ~~first~~ RSS ~~will~~ dropssharply and then smoothly, we can choose the turning point K.

②. Silhouette coefficient

$\{b[i]-a[i]\}/\max(a[i],b[i])$

a[i] = average dissimilarity within cluster.

b[i] = lowest dissimilarity with other clusters.

-1: bad misclassification

0 = ~~OK~~

1 = Good.

③. GAP statistics.

For each cluster, ~~simulate~~ calculate W_K, simulate randomly points and calculate W_{Kb},

$$GAP(K)=\frac{1}{B}\sum_{i=1}^{B}(\log W_{Kb}-\log W_K)$$

Compute sd(K), which is the standard deviation of $\log W_{Kb}$, $b=1,2\ldots B$

Find smallest K,

$$Gap(K)\geq Gap(K+1)-S_{K+1}$$

11. Neural Networks

- Central idea: extract linear combinations of the inputs as derived features, and then model the target as a non-linear function of these features.

11.2. Projection Pursuit Regression.

$$f(X)=\sum_{m=1}^{M} g_m(W_m^T X) \quad (11.1)$$

(PPR)

To fit, minimise error function:

$$\sum_{i=1}^{N}\left[y_i-\sum_{m=1}^{M} g_m(W_m^T X_i)\right]^2 \quad (11.2)$$

- Universal approximator: for large M PPR can approximate any continuous function in $\mathbb{R}^p$.

- Back propagation:

$$\begin{cases} Z_m = \sigma(\alpha_{0m}+\alpha_m^T X),\ m=1,2,\dots M \\ T_k = \beta_{0k}+\beta_k^T Z,\ k=1,2,\dots K \\ f_k(X) = g_k(T),\ k=1,2,\dots,K \end{cases}$$

$$R(\theta)=\sum_{i=1}^{N}\sum_{k=1}^{K}[y_{ik}-f_k(X_i)]^2$$

$$R(\theta)=-\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log f_k(X_i) \quad \text{(cross-entropy)}$$

$$\begin{cases} \frac{\partial R_i}{\partial \beta_{km}} = -2[y_{ik}-f_k(X_i)]\,g_k'(\beta_k^T Z_i)\,Z_{mi} = \delta_{ki} Z_{mi} \\ \frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2[y_{ik}-f_k(X_i)]\,g_k'(\beta_k^T Z_i)\,\sigma'(\alpha_m^T X_i)\,X_{il} = S_{mi} X_{il} \end{cases}$$

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N}\frac{\partial R_i}{\partial \beta_{km}}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N}\frac{\partial R_i}{\partial \alpha_{ml}^{(r)}}$$

(batch learning)

- Backpropagation: current weights are use to calculate $\hat{y}_{ik}$ (forward process). then the errors δ_{ki} and S_{mi} are calculated using $\hat{f}_k(X_i)$ (backward process). Then δ_{ki} and S_{mi} are used to calculate $\beta_{km}^{(r+1)}$, $\beta_{km}^{(r)}$.

pro: local, easy to implement in parallel architecture computer.

11.5 Some issues in training NN

- start around zero weights.
  ~~zero~~: never move
  large: poor solutions.

- regularization (weight decay)

$$J(\theta)=\sum_{km}\beta_{km}^2+\sum_{ml}\alpha_{ml}^2$$

$$J(\theta)=\sum_{km}\frac{\beta_{km}^2}{1+\beta_{km}^2}+\sum_{ml}\frac{\alpha_{ml}^2}{1+\alpha_{ml}^2}$$

- standardizing the inputs
  - affects scaling of weights.
  - regularization
  - ~~meo~~ choose meaning for range for starting weights.

- Choose large number of hidden layers with regularization.

- Nonconvex
  - different starting weights, choose minimum $R(\theta)+J(\theta)$.
  - bagging.

6

1.36 to 4.26

① Naive Bayes' : ~~[illegible]~~ Applies Bayes' rule to construct a posterior probability based on likelihood and prior, for e.g.

$p(C_k|X) \propto p(X|C_k)\, p(C_k)$

Naive means: $p(X|C_k) = \prod_j p(X_j|C_k)$

In order for prediction, choosing C_k which has maximum $p(C_k|X)$

优点: ①. Simple to implement
②. Handle "wide data" well, $p >> k$
③. Fast to train and predict, and good for online learning

缺点: ①. ~~[illegible]~~ Can be hampered by irrelevant features
②. Probabilistic estimates are not reliable because of naive assumption.
③. Outperformed by other models.
④. choice of prior affects the model performance.

②. Linear regression:

• Assumptions:
①. Linear relationship: $y = \beta_0 + \beta X + \varepsilon$
②. $E[\varepsilon|X] = 0$, $Var[\varepsilon|X] = \sigma^2$ is constant.
③. Distribution of X is arbitrary.
④. ε is independent.

$MSE = E[(\cancel{Y} - (\beta_0 + \beta_1 X))^2]$

$\hat{\beta}_1 = \frac{cov[X,Y]}{Var[X]}$, $\hat{\beta}_0 = E[Y] - \hat{\beta}_0 E[X]$

$MLE = $

$\log MLE = \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[y_i - (\beta_0 + \beta_1 x_i)]^2}{2\sigma^2}}$

$RSS = (Y - X\beta)^T (\cancel{Y} - X\beta)$

$\frac{\partial RSS}{\partial \beta} = -2X^T Y + 2X^T X \beta$

$\Rightarrow \hat{\beta} = (X^T X)^{-1} X^T Y$

优点: ①. Easy to interpret.
②. Fast to train and predict

缺点: ① Non-linearity, multicollinearity,
②. ~~Out~~ Sensitive to outliers
③. Can not handle $p >> n$.

③. Logistic regression:

Used for classification by modeling the probability of a class given observation. Because $p \in [0, 1]$, needs a functional form to map p to be $[0, 1]$.

$h_\theta(X) = \frac{1}{1+e^{-\theta^T X}} = p$

$\log \frac{p}{1-p} = \theta^T X$

• Assumptions:
①. Linear relationship between log odds and predictors
②. No multicolinearity

优点: ① Easy to interpret
② Fast to train and predict

缺点: ① Sensitive to outliers.
②. Non linearity, multicolinearity.

④、~~Tre~~ Decision tree:

that performs iterative binary split on the feature space. In order for prediction, we use ~~the~~ average of the training observations in the region for which it belongs to for regression, ~~and~~ or majority class in this region for classification.

- For each split, interate all the features and possible cut on that feature, choose the cut that causes largest decrease of RSS for regression, or largest information gain for classification.

- Feature importance: accumulate improvement on split-criterion at each split on that feature.

优点: ①. Easily handle Mixed predictors.
②. Needs little preprocessing of features.
③. Robust to Outliers
④. Hande multicolinearit, non-linearity
⑤. Small trees are easy to interpret.

缺点: ①. Deep trees hard to interpret.
~~②. Store complex models for future use.~~
③. Decision boundries are parrallel to the axis, not flexible.

⑤. Bagging.

- Bootstrap: Sampling with replacement. Average to reduce variance without ~~reduce~~ increasing bias by much.

$1-\left(1-\frac{1}{n}\right)^n$

- OOB error = used ~~for~~ validtion.

- Random forests: Decorrelate the tree. At each split, choose only a subset of the features.

优点: ①. Easy to tune (than boosting)
②. run parallel
③. give feature importance.
④. OOB error for validation.
⑤. Robust to missing data

缺点: ①. long time to train
②. overfitting
③. Store complex models for future use.
④. Hard to interpret.

turing parameters: (sklearn)
n_estimators
max_depth
max-features

⑦. SVM:

A classification machine learning algorithm usually for binary classes.

- In order to talk about SVM, first we need to know:
- hyperplane: An affine $(p-1)$-dimension plane in p-dimension space.
- margin: minimum distance from observation points to the seperating hyperplane.

- maximal margin classifer:

$$\underset{\beta_0,\beta_1,\dots\beta_p}{\text{maximize}}\ M$$

subject to $\sum_{j=1}^{p}(\beta_j)^2 = 1$

$$y_i\cdot(\beta_0+\beta_1 X_{i1}+\dots\beta_p X_{pi}) \geq M \quad \forall\, i=1,2,\dots n$$

- Support vector classifer

- observation points can be on the wrong side of margin or hyperplane. (soft margin)

$$\underset{\beta_0,\beta_1,\dots\beta_p,\ \varepsilon_1,\varepsilon_2\dots\varepsilon_i}{\text{maximize}}\ M \quad \text{subject to } \sum_{j=1}^{p}\beta_j^2=1$$

$$y_i\cdot(\beta_0+\beta_p X_{1i}+\dots\beta_p X_{pi}) \geq M(1-\underline{\varepsilon_i})$$

$$\forall\, i=1,2,\dots p$$

$$\sum_{i=1}^{n}\varepsilon_i = C.$$

- only depends on support vectors.

- hinge Loss: <u>hinge loss</u>

minimize:

$$\frac{1}{n}\sum_{i=1}^{n}\max[1,\ 1-y_i(\beta_0+\beta_1 X_{i1}+\dots\beta_p X_{ip})] + \lambda\sum_{j=1}^{p}\beta_j^2$$

Solve using quatratic programming, only need to solve the inner product between observations.

Use kernels to expand feature space, which is <u>computationally efficient</u>.

- polynomial kernal:

$$K(X_i, X_{i'}) = \left(1+\sum_{j=1}^{p} X_{ij}X_{i'j}\right)^d$$

- radial kernal:

$$K(x_i, x_{i'}) = \exp\left[-\gamma\sum_{j=1}^{p}(X_{ij}-X_{i'j})^2\right]$$

- Gaussial kernal:

$$K(X_i, X_{i'}) = \exp\left[-\frac{1}{2\sigma^2}\sum_{j=1}^{p}(X_{ij}-X_{i'j})^2\right]$$

---

maximum margin classifier

$$\text{minimize} \sum_{j=1}^{p}\beta_j^2$$

subject to:

$$y_i\cdot(\beta_0+\beta_1 X_{1i}+\dots\beta_p X_{pi}) \geq 1$$

⑥. Boosted decision tree (GBDT)

Start with a weak learner, a shallow tree which has low variance but high bias, iterately learn the residuals from the previous model in order to reduce the bias without increasing the variance by much.

- Adaboost

The idea is to apply weights to the observations, and for each iteration, by updating weights, assign more weights to the misidentified observations.

①. First assign all the events the weight $w_i = \frac{1}{N}$.

②. For each iteration,

a. Fit a classifier $G_m(x)$ based on w_i from last step.

b. Calculate an weighted error rate:

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(X_i))}{\sum_{i=1}^{N} w_i}$$

c. $\alpha_m = \log[(1-err_m)/err_m]$

d. update the weights:

$$w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(x))]$$

③. 输 $\mathrm{sign}[\sum_{i=1}^{m} \alpha_m G_m(X_i)]$

- Gradient boosting:

learn pseudo-residuals from previous step

①. First, initialize

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

② For each iteration, calculate the pseudo-residuals,

$$r_{im} = -\left.\frac{\partial L[y_i, F_m(X_i)]}{\partial F(X_i)}\right|_{F(x)=F_{m-1}(X)}$$

b. Fit a base learner to r_{im}, $h_m(X)$

c. $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L[y_i, F_m(X) + \gamma_m h_m(X_i)]$

d. Update the model:

$$F_m(x) = F_{m-1}(X) + \underbrace{\alpha}_{\text{learning rate}} \gamma_m h_m(X)$$

③. Output $F_M(x)$

优点: ① strong prediction power.
②. multicolinearity, non-linearity
③ robust to outliers

缺点: ① slow to train
② overfitting
③ Hard to interpret.

$\boxed{\text{Random Forest}}$

n_estimators :
max_depth :
min_samples_leaf :
max_features = 'auto'.

$$\boxed{L = \exp[-yf(x)]}$$

- AdaBoost

1. Initialize : $w_i = \frac{1}{N}$

2. For $m = 1$ to M :

(a) Fit $G_m(x)$ to the training data using w_i.

(b) complete error rate :

$$err_m = \frac{\sum_i w_i I[y_i \neq G_m(x_i)]}{\sum_i w_i}$$

(c) update the weights :

$$\alpha_m = \log[(1-err_m)/err_m]$$

$$w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$$

3. $G(x) = \text{sign}\left[\sum_{m=1}^{M} m G_m(x)\right]$.

$\boxed{\text{Boosting}}$

$$f(x) = \sum_{m=1}^{M} \beta_m b(x, \gamma_m)$$

$$\min_{(\beta_m, \gamma_m)} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i, \gamma_m)\right)$$

$\boxed{\text{GBDT}}$

1. $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$

2. For $m = 1$ to M :

① pseudo-residual $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x)}$

② base learner $h_m(x)$ to $[(x_i, r_{im})]$

③ $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$.

④ Update :

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

3. Output $F_m(x)$.