Statistical Learning Condensed Notes

Daniel Lin; Professor: Prof. Guy Nason

May 19, 2023

1 Regression

Meaning of regression: Find a scientific relationship between two(or multiple) variables.

1.1 Basic Concepts

Mean-squared Error

$$\text{MSE}(\hat{\theta}) = E((\hat{\theta} - \theta)^2) = \text{var}(\hat{\theta}) + \text{bias}(\hat{\theta})^2$$

Degree of Freedom the number of pieces of information we have to estimate the population's value.
Types of variables

- Nominal: non-ordered categories
- ordered categories
- interval: numerical, 0 arbitrary
- ratio: numerical, meaningful 0

Centering, standardising data

$$x^*_{i,j} = \frac{x_{i,j} - \overline{x}_j}{s_j}$$

where s_j is jth sample standard deviation.

Sample mean, variance, covariance Suppose X, Y are data matrices,

$$\overline{X} = \frac{\mathbf{1}^T X}{n}$$

$$\text{Var}(X) = \frac{X^T X}{n}$$

$$\text{Cov}(X, Y) = \frac{X^T Y}{n}$$

if C is a constant matrix,

$$E(CX) = CE(X), \quad \text{Var}(CX) = C\text{Var}(X)C^T$$

Condition number
Measures invertibility of matrix, high condition number: difficult to invert. If A is normal (i.e. $A^*A = AA^*$),

$$\kappa(A) := \frac{\xi_{\max}(A)}{\xi_{\min}(A)}$$

where ξ means Eigenvalue.

Singular Value Decomposition
$X_{n\times p} = U_{n\times p} D_{p\times p} V^T_{p\times p}$ diagonal of D are singular values of X, in decreasing order. U, V are orthogonal.

1.2 Simple Linear Regression

Simple Linear Model
Given data $(W_i, H_i)_{i\in\{1,\cdots,n\}}$,

$$W_i = a + bH_i + \epsilon_i$$

Model assumptions:

- Errors ϵ_i are i.i.d. This implies homoscedasticity: $\text{var}(\epsilon_i)$ is constant.
- $E(\epsilon_i) = 0$

Residuals $e_i(a,b) = W_i - a - bh_i$

$$R(a,b) = \frac{\sum_i e_i(a,b)^2}{n}$$

Least square estimators $(\tilde{a}, \tilde{b})$ can be found by differentiation.

Variants

- Maximum likelihood: give distribution f to errors, if $\mathcal{D}$ represents data,

$$L((a,b)|\mathcal{D}) := f(\boldsymbol{e}|(a,b)) = \prod_{i=1}^{n} f(e_i(a,b)|(a,b))$$

  MLE $(\hat{a}, \hat{b})$ given by maximising the likelihood $L((a,b)|\mathcal{D})$.
  - When $\epsilon_i \sim N(0, \sigma^2)$, MLE equivalent to least square estimator
- Heavy-tailed, skew: assign distribution other than Gaussian to errors. e.g. Student's t.

Multivariate Linear Model

$$Y_i = \sum_{k=1}^{p} X_{i,k}\beta_k + \epsilon_i \quad \text{for } i = 1, \cdots, n$$

Matrix form:

$$Y_{n\times 1} = X_{n\times p}\beta_{p\times 1} + \epsilon_{n\times 1}$$

Model assumption: $\epsilon \sim N(0, \sigma^2)$

Residual $e = Y - X\beta$, RSS: $e^T e$, Least square estimator:

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

when X has full rank. (Proved by checking $\partial\text{RSS}(\beta)/\partial\beta = 0$, $\partial^2\text{RSS}(\beta)/\partial\beta\partial\beta^T \succ 0$)

Orthogonality of residual:

$$X^T(Y - X\hat{\beta}) = 0$$

Hat matrix $H := X(X^T X)^{-1} X^T$, fitted values $\hat{Y} := X\hat{\beta} = HY$

Inference:
$E(Y) = X\beta$, $\text{Var}(Y) = E(\epsilon\epsilon^T) = \sigma^2 I_n$, $E(YY^T) = \sigma^2 I_n + X\beta\beta^T X^T$
$E(\hat{\beta}) = \beta$, $\text{Var}(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$.
Z-score:

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}}$$

where v_j is jth diagonal entry of $(X^T X)^{-1}$, $\hat{\sigma}^2$ is sample variance

$$\hat{\sigma}^2 := \frac{\sum_i (y_i - \hat{y}_i)}{n - p}$$

Note $E(\hat{\sigma}^2) = \sigma^2$. If $\beta_j = 0$, $z_j \sim t_{n-p}$. Use the p-value on the Z-score to test whether $\beta_j = 0$.

Comparing Models:
M0: $p_0 + 1$ variables,
M1: $p_1 + 1$ variables where $p_1 > p_0$ (M1 is an extended version of M0)
F statistics:

$$F = \frac{(\mathrm{RSS}_0 - \mathrm{RSS}_1)/(p_1 - p_0)}{\mathrm{RSS}_1/(n - p_1 - 1)}$$

if M0 is true, $F \sim F_{p_1 - p_0, n-p_1-1}$.
Backward-stepwise selection: fit the full model, and delete the predictor with the least impact one by one. Stop when the p-values of all remaining predictors are less than 0.05.
Forward-stepwise selection: start with only intercept, and sequentially add the term that improves the model most. Stop when no improvement is seen.

Gauss Markov Theorem: Least square estimator $\hat{\beta}$ is the best linear unbiased estimator (BLUE). i.e. it has the smallest variance. And this is true for any linear combination of β.

Proof. If $\check{\beta} = CY$ is another unbiased linear estimator, define difference $D := C - (X^TX)^{-1}X^T$ (so $\check{\beta} = \hat{\beta} + DY$). Then by expansion $C = D + (X^TX)^{-1}X^T$, $E(\check{\beta}) = (I_p + DX)\beta$. So $DX = 0$.
Similarly, $\mathrm{Var}(\check{\beta}) = \mathrm{Var}(\hat{\beta}) + \sigma^2 DD^T \geq \mathrm{Var}(\hat{\beta})$.

If $\theta = \alpha^T\beta$ (a linear combination of β), $\hat{\theta} := \alpha^T\hat{\beta}, \check{\theta} := \alpha^T\check{\beta}$ are both unbiased and

$$\mathrm{Var}(\alpha^T\check{\beta}) = \mathrm{Var}(\alpha^T\hat{\beta}) + \sigma^2\alpha^T DD^T\alpha \geq \mathrm{Var}(\alpha^T\hat{\beta})$$

□

1.3 Ridge Regression

$\hat{\beta}^{\text{ridge}}(\lambda)$ is the minimiser of

$$\sum_i \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j\right)^2 + \lambda\sum_j \beta_j^2$$

$\lambda = 0$: $\hat{\beta}^{\text{ridge}} = \hat{\beta}$
$\lambda \to \infty$: $\hat{\beta}_j^{\text{ridge}} \to 0$, $\hat{\beta}_0^{\text{ridge}} \to \overline{Y}$.
Constraint formulation:

$$\hat{\beta}^{\text{ridge}} = \mathrm{Argmin}_\beta \sum_i \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j\right)^2, \quad (1)$$

$$\text{s.t.} \sum_j \beta_j^2 \leq t \quad (2)$$

Analytic form:

$$\hat{\beta}^{\text{ridge}} = (X^TX + \lambda I_p)^{-1}X^TY$$

This is a biased estimator of β. For $\lambda > 0$, $X^TX + \lambda I_p$ is more invertible than X^TX.

Bayes view:
If prior $\beta_j \sim N(0, \tau^2)$ independently,
$Y_i \sim N(\beta_0 + x_i^T\beta, \sigma^2)$ independently,
then $\beta|Y \sim MVN(\hat{\beta}^{\text{ridge}}(\sigma^2/\tau^2), \Sigma)$ where $\Sigma^{-1} = (X^TX + (\sigma^2/\tau^2)I)/\sigma^2$. $\hat{\beta}^{\text{ridge}}(\sigma^2/\tau^2)$ means ridge estimator with regularisation parameter $\lambda = \sigma^2/\tau^2$ Also, the mode of $\beta|Y$ is $\hat{\beta}^{\text{ridge}}(\sigma^2/\tau^2)$.

SVD view:
If $X = UDV^T$ is SVD decomposition,

$$X\hat{\beta} = UU^TY, \quad X\hat{\beta}^{\text{ridge}} = UD(D^2 + \lambda I_p)^{-1}DU^TY = \sum_j \boldsymbol{u}_j \frac{d_j^2}{d_j^2 + \lambda} \boldsymbol{u}_j^T Y$$

where $\boldsymbol{u}_j$ is jth column of U. Interpretation: transform to basis U, shrink coordinates by $d_j^2/(d_j^2 + \lambda)$.
Effective Hat matrix: $H_\lambda := X(X^TX + \lambda I_p)^{-1}X^T$. So $H_\lambda Y = \hat{Y}^{\text{ridge}} := X\hat{\beta}^{\text{ridge}}$
Effective degrees of freedom:

$$\text{tr}(H_\lambda) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$$

Note: trace is the sum of Eigenvalues of a matrix. The degree of freedom for ridge regression is defined by its shrinkage effect.

Principal Components:
If $X = UDV^T$, $X^TX = VD^2V^T$. Eigenvectors of X^TX (i.e. columns of V, v_j) are called *principal components*
Projection onto principal components: $z_j = Xv_j$, variance along principal components:

$$\frac{E(z_j^T z_j)}{n} = \frac{d_j^2}{n}$$

so ridge regression shrinks coordinates in the directions with smaller variance.

1.4 LASSO Regression

$\hat{\beta}^{\text{lasso}}(\lambda)$ is the minimiser of

$$\frac{1}{2}\sum_i \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j \right)^2 + \lambda \sum_j |\beta_j|$$

$\lambda = 0$: $\hat{\beta}^{\text{lasso}} = \hat{\beta}$
$\lambda \to \infty$: $\hat{\beta}_j^{\text{lasso}} \to 0$, $\hat{\beta}_0^{\text{lasso}} \to \overline{Y}$.
Constraint formulation:

$$\hat{\beta}^{\text{lasso}} = \text{Argmin}_\beta \sum_i \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j \right)^2, \tag{3}$$

$$\text{s.t.} \sum_j |\beta_j| \leq t \tag{4}$$

No closed form formulae in general.

LASSO has stronger shrinkage power, some coefficients are even shrunk to 0 (i.e. variables are removed by LASSO).

Penalty bound $t_0 := \sum_{j=1}^{p} |\hat{\beta}|$, For $t > t_0$ in constraint, $\hat{\beta}^{\text{lasso}} = \hat{\beta}$

Comparison of the simple linear model (ls), Ridge, LASSO when X is orthogonal:

$$\hat{\beta}^{\text{ls}} = X^TY, \quad \hat{\beta}^{\text{ridge}}(\lambda) = \frac{X^TY}{1+\lambda} = \frac{\hat{\beta}^{\text{ls}}}{1+\lambda}$$

$$\hat{\beta}^{\text{LASSO}} = \text{sign}(\hat{\beta}^{\text{ls}})(|\hat{\beta}^{\text{ls}}| - \lambda)^+$$

where $x^+ := xI(x > 0)$.
brief proof: By writing out the minimisation problem for LASSO and using orthogonality to simplify,

$$\hat{\beta}^{\text{LASSO}} = \text{Argmin}_\beta \sum_{j=1}^{p} \left(-\hat{\beta}^{\text{ls}}\beta_j + \frac{1}{2}\beta_j^2 + \lambda|\beta_j| \right)$$

Then consider cases $\hat{\beta}^{\text{ls}} > 0$ and $\hat{\beta}^{\text{ls}} < 0$: in the first case, it can be shown that $\beta_j < 0$ is not feasible, so $\beta_j \geq 0$. Then differentiation yields $(|\hat{\beta}^{\text{ls}}| - \lambda)^+$. Another case can be dealt with similarly.

1.5 Principal Component Regression

Given Eigenvectors v_m of $X^T X$, $z_m := X v_m$ is the data projection onto the ith principal component. Pick M s.t. the first M principal components are meaningful (i.e. first M Eigenvalues are large)
PC regression is

$$\hat{y}^{\text{pcr}}_{(M)} := \overline{Y} 1_n + \sum_{m=1}^{M} \hat{\theta}_m z_m = \overline{Y} 1_n + X \sum_{m=1}^{M} \hat{\theta}_m v_m$$

where regression coefficient $\hat{\theta}_m := \langle z_m, y \rangle / \langle z_m, z_m \rangle$. One can define

$$\hat{\beta}^{\text{pcr}} := \sum_{m=1}^{M} \hat{\theta}_m v_m$$

so $\hat{y}^{\text{pcr}}_{(M)} = \overline{Y} 1_n + X \hat{\beta}^{\text{pcr}}$.

$M = p$: restores least square regression.

Optimisation view
Define $S := n^{-1} X^T X$, principal components are its Eigenvectors.
Assume X is centred, suppose project to arbitrary a s.t. $\|a\| = 1$,

$$y := Xa$$

then $E(y) = 0$, sample variance $S_y(a) = n^{-1} y^T y = a^T S a$.

Solution to

$$\max_a S_y(a) \quad \text{s.t. } a^T a = 1$$

is v_1 (first principal component).

$$\max_a S_y(a) \quad \text{s.t. } a^T a = 1,\ a^T v_i = 0 \ \text{ for } j = 1, \cdots, j-1$$

is solved by v_j.

1.6 Spline Model

Linear Basis expansion:

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$$

where $\{h_m\}$ is a fixed set of functions. The set of candidates for h_m is called a dictionary.

Controlling complexity:

- Restriction: limit the number of bases:

$$f(X) = \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{j,m} h_{j,m}(X_j)$$

  for fixed integers M_j

- Selection: Sequentially put in h_m from dictionary that improves fit.

- Regularisation: include the whole dictionary, but control coefficients β_m.

Polynomial Basis
Usually, local polynomials are fitted. i.e. knots $\{\xi_k\}_{i=1,\cdots,K}$ are chosen and the space is divided into intervals $[\xi_{i-1}, \xi_i)$.

Common models
Piecewise constant polynomial:

$$h_i(X) = I(\xi_{i-1} \leq X \leq \xi_i)$$

Piece-wise linear: additional basis functions

$$h_i'(X) = I(\xi_{i-1} \leq X \leq \xi_i)X$$

Continuous piece-wise linear: additional constraints that $f(\xi_i^-) = f(\xi_i^+)$.
or use basis functions:

$$h_1(X) = 1,\ h_2(X) = X,\ h_{2+i}(X - \xi_i)_+$$

Continuous piece-wise cubic:

$$h_1(X) = 1,\ h_2(X) = X, h_3(X) = X^2, h_4(X) = X^3,\ h_{4+i}(X - \xi_i)_+^3$$

Natural cubic spline: two end regions are linear (releases 4 degrees of freedom for knots)

$$N_1(X) = 1,\ N_2(X) = X,\ N_{k+2}(X) = d_k(X) - d_{K-1}(X)$$

$$\text{where } d_k(X) := \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$$

Smoothing Spline Search among twice differentiable functions, minimise

$$\text{RSS}(f, \lambda) := \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 + \lambda \int (f''(t))^2 \, dt$$

λ penalises the curvature.
Unique minimisation: natural cubic spline with knots at unique values of x_i.
Model:

$$f(x) = \sum_{j=1}^{n} N_j(x)\theta_j$$

where $\{N_j\}$ is the n-dimensional basis for natural cubic spline. Using this f,

$$\text{RSS}(f, \lambda) = (y - N\theta)^T(y - N\theta) + \lambda\theta^T\Omega_n\theta =: \text{RSS}(\theta, \lambda)$$

where $N_{i,j} := N_j(x_i)$, $(\Omega_n)_{i,j} := \int N_i''(t)N_j''(t)\, dt$. Solution:

$$\hat{\theta} = (N^TN + \lambda\Omega_n)^{-1}N^Ty$$

Fitted function: $\hat{f} = S_\lambda y$ where $S_\lambda := N(N^TN + \lambda\Omega_n)^{-1}N^T = (I + \lambda(N^{-T}\Omega_nN^{-1}))^{-1}$ (Reinsch form). S_λ is symmetric, positive semi-definite. Using Reinsch form, define $K := N^{-T}\Omega_nN^{-1}$, then

$$\text{RSS}(f, \lambda) = (y - f)^T(y - f) + \lambda f^TKf$$

If $S_\lambda = \sum_{k=1}^{n} \rho_k(\lambda)u_ku_k^T$(Eigendecomposition) where u_k are Eigenvectors (they do not depend on λ), $\rho_k(\lambda)$ are Eigenvalues, then

$$\rho_k(\lambda) = \frac{1}{1 + \lambda d_k}$$

where d_k is corresponding Eigenvalue of K. Effective degree of freedom: $df_\lambda = \text{tr}(S_\lambda)$. When df_λ is around 2, $\hat{f}$ is close to a linear function.

choice of λ:
aims to minimise MISE $:= E\left[\int \left\{\hat{f}_\lambda(x) - f(x)\right\}^2 dx\right]$. This is estimated by cross-validation:

$$\text{CV}(\hat{f}_\lambda) := \sum_i \left(y_i - \hat{f}_\lambda^{(-i)}(x_i)\right)^2 = \frac{1}{n}\sum_i \left(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - (S_\lambda)_{i,i}}\right)^2$$

where $\hat{f}_\lambda^{(-i)}$ is smoothing spline estimator constructed with ith point ignored.

1.7 Density Estimation

Aim: given data $X_1, \cdots, X_n$, estimate pdf $f(x)$.

kernel function is smooth function $K : \mathbb{R} \to \mathbb{R}$ s.t.

- $K(x) \geq 0$
- $\int_{\mathbb{R}} K(x)\, dx = 1$
- $K(-x) = K(x)$, i.e. $\int xK(x)\, dx = 0$

KDE(kernel density estimator)

$$\hat{f}_{n,h,K}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)$$

h - bandwidth, n - number of observations, K - chosen Kernel function.
small h: high variance, low bias; large h: high bias, low variance. A good choice of h should depend on n and $h_n \to 0$ as $n \to \infty$.

Common choices of K:
Rectangular

$$K(x) = \begin{cases} 1/2, & -1 < x < 1 \\ 0, & \text{otherwise} \end{cases}$$

Triangular

$$K(x) = \begin{cases} 1 - |x|, & -1 < x < 1 \\ 0, & \text{otherwise} \end{cases}$$

Epanechnikov

$$K(x) = \begin{cases} \frac{3}{4}(1 - x^2), & -1 < x < 1 \\ 0, & \text{otherwise} \end{cases}$$

Gaussian

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$

Expectation and Variance

$$E(\hat{f}(x)) = (nh)^{-1} \sum_{i=1}^{n} E\left\{K\left(\frac{X_i - x}{h}\right)\right\}$$

conditioning on X_i, use a change of variable and Taylor expansion,

$$E\left\{K\left(\frac{X_i - x}{h}\right)\right\} = hf(x) + \frac{1}{2}C_3 h^3 f''(x) + \mathcal{O}(h^4)$$

so $E(\hat{f}(x)) \approx f(x) + \frac{1}{2}C_3 h^2 f''(x)$, bias $\approx \frac{1}{2}C_3 h^2 f''(x)$. (where $C_3 := \int v^2 K(v)\, dv$ is independent of x)

By iid of X_i,

$$\text{Var}(\hat{f}(x)) = \frac{1}{nh^2} \text{Var}\left\{K\left(\frac{X_i - x}{h}\right)\right\}$$

using $\text{Var}(W) \leq E(W^2)$ and similar techniques to above,

$$\text{Var}(\hat{f}(x)) \leq \frac{1}{nh} f(x) C_2 + \frac{1}{n} C_4 + \mathcal{O}(h/n) \to 0 \quad \text{as } n \to \infty$$

where $C_2 := \int K^2(v)\, dv,\ C_4 := f'(x) \int vK^2(v)\, dv$.

Kernel Regression
Define $K_h(x) := h^{-1}K(x/h)$, then 1-D kernel estimator is

$$\hat{f}(x) = n^{-1}\sum_{i=1}^{n} K_h(x - X_i)$$

2D kernel estimator:

$$\hat{f}(x, y) = n^{-1}\sum_{i=1}^{n} K_h(x - X_i)K_h(y - Yi)$$

Can estimate $E(Y|X = x)$ by

$$\hat{E}(Y|X = x) = \frac{\int y\hat{f}(x, y)\, dy}{\hat{f}(x)} = \frac{\sum_{i=1}^{n} Y_i K_h(x - X_i)}{\sum_{i=1}^{n} K_h(x - X_i)}$$

this is *Nadaraya-Watson Kernel Estimator*.

1.7.1 Local Polynomial Regression

Local estimator centred on x_0:

$$m_{x_0}(x) = \sum_{j=0}^{p} \beta_j(x_0)(x - x_0)^j$$

generally order p is chosen to be odd.
RSS is weighted by a Kernel function

$$\text{RSS}(x_0) = \sum_{i=1}^{m}\{Y_i - m_{x_0}(X_i)\}^2 K_h(X_i - x_0)$$

In matrix form

$$\text{RSS}(x_0) = \{Y - X\beta(x_0)\}^T W_{x_0}\{Y - X\beta(x_0)\}$$

where

$$X = \begin{pmatrix} 1 & X_1 - x_0 & \cdots & (X_1 - x_0)^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_n - x_0 & \cdots & (X_n - x_0)^p \end{pmatrix}, \quad W_{x_0} := \text{diag}\{K_h(X_1 - x_0), \cdots, K_h(X_n - x_0)\}$$

Minimiser: (solved using the square root of W_{x_0}, transform it to form of least square)

$$\hat{\beta}(X_0) = (X^T W_{X_0} X)^{-1} X^T W_{X_0} Y$$

Bias:
Let $f(x)$ be density of X_i, $g(x) := E(Y|X = x)$ is the regression function.
Bias of Nadaraya-Watson estimator:

$$h^2\left\{\frac{1}{2}g''(x) + \frac{g'(x)f'(x)}{f(x)}\right\}\int K^2(u)\, du + o(h^2)$$

The bias of local polynomial regression:

$$h^2\frac{1}{2}g''(x)\int K^2(u)\, du + o(h^2)$$

1.7.2 Orthogonal Basis

Inner product on function space: $\langle f, g\rangle = \int f(x)g(x)\,dx$.
Orthogonal basis $\{\rho_\nu(x)\}$: $\langle \rho_\nu, \rho_\mu\rangle = \delta_{\nu,\mu}$. Expansion of a function in the orthogonal basis is called *orthogonal series expansion.*
2D separable expansion:

$$f(x,y) = \sum_\nu \sum_\mu f_{\nu,\mu}\rho_\nu(x)\rho_\mu(y)$$

where

$$f_{\nu,\mu} = \int\int f(x,y)\rho_\nu(x)\rho_\mu(y)\,dx\,dy$$

f is unknown, so coefficients f_ν are estimated by

$$f_\nu := \int f(x)\rho_\nu(x)\,dx = E(\rho_\nu(X)) \approx \frac{\sum_i \rho_\nu(X_i)}{n} =: \hat{f}_\nu$$

it is unbiased. Similarly, $\hat{f}_{\nu,\mu} := n^{-1}\sum_{i=1}^n \rho_\nu(X_i)\rho_\mu(Y_i)$ is unbiased estimator of $f_{\nu,\mu}$.
e There can be infinite ρ_ν, so in practice, a linear orthogonal series estimator with truncation (at m) is used

$$\hat{f}(x) := \sum_{\nu=-m}^{m} \hat{f}_\nu \rho_\nu(x)$$

it is biased.

Fourier Basis: $\rho_\nu(x) = \exp\left(-\frac{2\pi i\nu x}{T}\right)$

$$\hat{f}_\nu = T^{-1}\hat{X}(\nu/T)$$

where $\hat{X}(\omega) := n^{-1}\sum_{j=1}^n e^{-2\pi i\omega X_j}$ is the discrete Fourier transform of $\{X_j\}$.

Fourier Transform of KDE:

$$R_h(\omega) := \int \hat{f}_h(x)\exp(-2\pi i\omega x)\,dx = \hat{X}(\omega)\tilde{K}_h(\omega)$$

where $\tilde{K}_h(\omega) := \int_{\mathbb{R}} K_h(y)\exp(2\pi i\omega y)\,dy$ (inverse Fourier transform of K_h). The inverse of the above equation:

$$\hat{f}_h(x)^{\text{KDE}} = \int_{\mathbb{R}} R_h(\omega)e^{2\pi i x\omega}\,d\omega = \int_{\mathbb{R}} \hat{X}(\omega)\tilde{K}_h(\omega)e^{2\pi i x\omega}\,d\omega$$

this gives a shortcut to find KDE: take the inverse Fourier transform of the basis function K_h, take the discrete Fourier transform of X_j and then multiply them together and take the inverse Fourier transformation. This is faster than substituting X_j to K_h for each j and summing them up.

Parseval's theorem and Plancheral's theorem If f, g are two orthogonal series expansions i.e. $f(x) = \sum_\nu f_\nu \xi_\nu(x)$, $g(x) = \sum_\nu g_\nu \xi_\nu(x)$ for orthogonal basis $\{\xi_\nu\}$ and coefficients f_ν, g_ν, then

$$\langle f, g\rangle = \langle F, G\rangle$$

where $F := \{f_\nu\}_\nu$, $G := \{g_\nu\}_\nu$. The first inner product is in the functional space $\langle f, g\rangle = \int f\overline{g}\,dx$, whereas the second one is over the vector space.

(Plancheral's theorem) $\|f\|^2 = \|F\|_\nu^2$. Energy across time is equivalent to energy across frequency.

1.8 Wavelets

Concept of Multi-Resolution Analysis(MRA): Examine a function at different scales. Approximation space $V_j \subseteq L_2$ (functions in L_2 have a finite number of discontinuities over a finite range) Approximation of f with resolution j: f_j is the projection of f onto V_j.

Defining properties of resolution spaces $\{V_j\}_{j\in\mathbb{Z}}$

- $\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots$ (monotone)
- $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L_2(\mathbb{R})$ (coverage)
- $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$ (less details for smaller j)
- $f(x) \in V_j \Rightarrow f(2x) \in V_{j+1}$ (resolution increase)
- $f(x) \in V_0 \Rightarrow f(x-k) \in V_0$ for any $k \in \mathbb{Z}$ (translation)
- Exists $\phi(x) \in V_0$ s.t. $\{\phi(x-k)\}_{k\in\mathbb{Z}}$ is orthonormal basis of V_0. This ϕ is called the father wavelet. (Existence of basis)

Haar Father Wavelet:

$$\phi(x) = \phi_H(x) := \begin{cases} 1, & x \in (0,1) \\ 0, & \text{otherwise} \end{cases}$$

Scale and translation of father wavelet:

$$\phi_{j,k}(x) := 2^{j/2}\phi(2^j x - k) \quad j,k \in \mathbb{Z}$$

j stands for resolution, $\{\phi_{j,k}\}_{k\in\mathbb{Z}}$ is basis for V_j.
Projection to V_j:

$$f_j(x) := \sum_{k\in\mathbb{Z}} c_{j,k}\phi_{j,k}(x) \quad \text{, where } c_{j,k} := \int f(x)\phi_{j,k}(x)\,dx$$

Increase of resolution

$$f_1(x) = c_{0,0}\phi(x) - d_{0,0}\psi(x)$$

where

$$\psi_H(x) := \begin{cases} 1, & x \in (0,1/2) \\ -1, & x \in (1/2,1) \\ 0, & \text{otherwise} \end{cases} \text{ (mother wavelet)}, \quad d_{0,0} := \frac{c_{1,1} - c_{1,0}}{\sqrt{2}}$$

Combination of resolution space: $V_1 = V_0 \oplus W_0$ (as ϕ is orthogonal to ψ)

General Case: starting from primary resolution j_0, wavelet representation is

$$f(x) = \sum_{k\in\mathbb{Z}} c_{j_0,k}\phi_{j_0,k}(x) + \sum_{j=j_0}^{\infty}\sum_{k\in\mathbb{Z}} d_{j,k}\phi_{j,k}(x)$$

Resolution spaces:

$$L_2 = V_{j_0} \oplus \bigoplus_{j=j_0}^{\infty} W_j$$

Relation of coefficients (this allows reconstruction of f merely from coefficients of the highest resolution)

$$c_{j-1,k} = \frac{c_{j,2k+1} + c_{j,2k}}{\sqrt{2}}, \quad d_{j-1,k} = \frac{c_{j,2k+1} - c_{j,2k}}{\sqrt{2}}$$

found using the Pyramid algorithm. In matrix form: $d = Wy$ where $y_i := c_{J,i}$ are the coefficients at resolution J.

Other father and mother wavelets may be used, but the coefficient relationship may change.

Wavelet has m vanishing moments if

$$\int x^l \phi(x)\,dx = 0$$

for $l = 0, 1, \cdots, m-1$. On smooth parts of f, wavelets have vanishing moments so this provides sparsity.

Wavelet Shrinkage
Given model $y_i = f_i + \epsilon_i$, y_i:observed values, ϵ_i:errors, f_i:unknown function.

Wavelet transform: $w = d + e$ where $w = Wy, d = Wf$, error $e = W\epsilon$ satisfies $E(e) = 0$, $\text{Var}(E) = \sigma^2 I_n$ (because $WW^T = I_n$). By Parseval's theorem, $\|d\| = \|f\|$.
Apply thresholding to w:

$$T_{\text{hard}}(w, \lambda) = wI(|w| > \lambda), \quad T_{\text{soft}}(w, \lambda) = \text{sign}(w)(|w| - \lambda)I(|w| > \lambda)$$

Bayesian Shrinkage:
Due to the sparsity of the wavelet transform, a prior(Berger-Müller prior) can be given to coefficients d_j:

$$d_{j,k} = \gamma_j N(0, \tau_j^2) + (1 - \gamma_j)\delta_0(x) \quad \forall k \in \mathbb{Z}$$

where γ_j are Bernoulli random variables with parameter p_j.
Posterior:

$$F(d|w) = r\Phi\left(\frac{d - wv^2}{\sigma v}\right) + (1 - r)I(d > 0)$$

where $v^2 = \tau^2(\sigma^2 + \tau^2)^{-1}$.

1.9 R-related

A simple linear Model can be fitted in R via `lm(y ~ x_1+x_2 + ... + x_p)` where x_i are predictors(variables). `summary(lm)` gives the quantiles of residuals, and statistical significance of each predictor in terms of the p-value of the t-test. It also gives the omnibus F test statistics.
Ridge and LASSO regression is done using `glmnet` . The best regularisation parameter λ can be found by cross-validation, this is embedded in `glmnet` .
`pcr` function in library `pls` performs principal component regression.
Smoothing spline is coded by `smooth.spline` in library MASS.

Wavelet: `wd` in library `wavethresh` , it finds wavelet coefficients

1.9.1 ANOVA (Analysis of Variance)

For regression, ANOVA uses F-statistics to test whether predictors in the linear model have enough influence on the response(dependent variable).

Degree of freedom:
$df_T (= p - 1)$: degree of freedom of the predictor
$df_R (= n - p - 1)$: degree of freedom of residuals
Sums of square:
SST: $\sum_i (\hat{W}_i - \overline{W})^2$ sum of squares error for regression,
SSR: $\sum_i (\hat{W}_i - W_i)^2$ sum of square of residuals.
Means squared errors
MST := SST/df_T, MSR := SSR/df_R.

F-statistics: F = MST/MSR
Large F: MSB large, MSE small, so predictor is strong
Small F: MSB small or MSE large, no evidence for the significance of predictor.

1.9.2 Diagonostic Plots

Code: `plot.lm()`

Four plots will be produced:
1. residuals vs fitted values: check whether residuals have zero means.
2. Standardised residuals vs fitted values: Check homoscedasticity.
3. QQ plot of residuals: check the distribution of residuals. See the QQ plot of various distributions in Figure1.
4. Standardised residuals against leverage, with Cook's distance plotted:

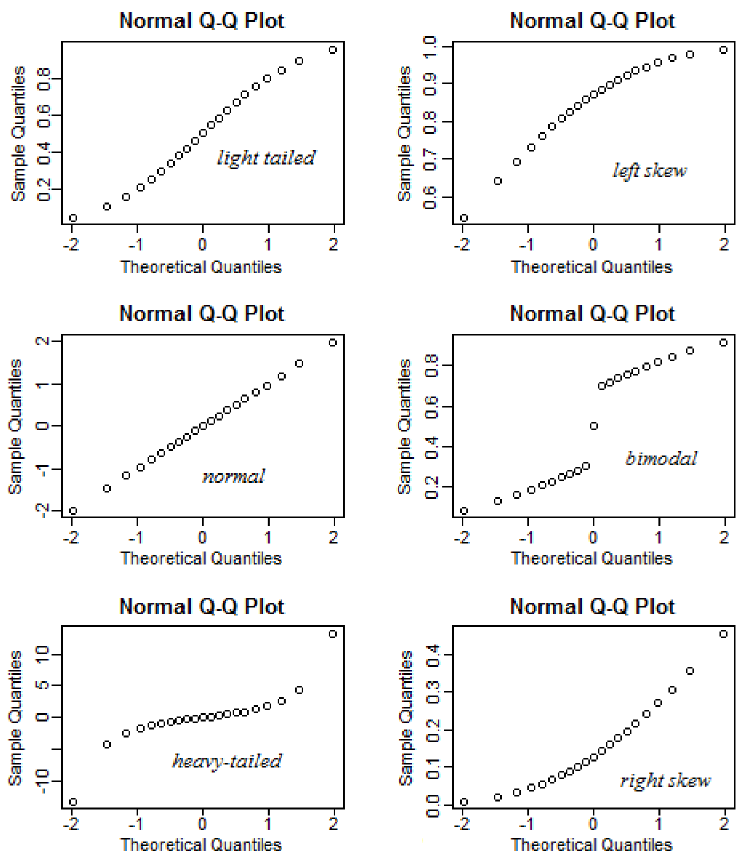


Figure 1: Possible problems reveals from QQ plot

Leverage: influence of each point on the regression
Cook's distance: change of fitted model by omitting a point.

1.9.3 Error Correlation

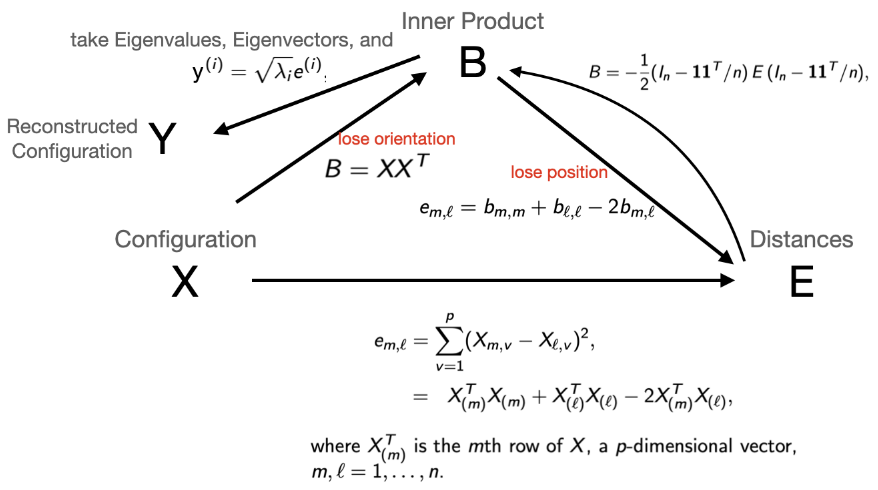
If $\text{cor}(\epsilon_i, \epsilon_j) \neq 0$, simple regression model fails.

Lagged scatter plot: $(v_t, v_{t+\tau})$: detects correlation
Durbin-Watson Statistics: tests auto-correlation ($= \text{cov}(v_t, v_{t+\tau})/\text{var}(v_t) = \text{cor}(v_t, v_{t+\tau})$).

It is also worth checking the relation(or correlation) of variables using `pairs(~x_1+x_2 + ... + x_p)` . But mind the scale of plots (you can use `eqscplot` which plots using equal scales).

2 Distance-Based Methods

2.1 Classical Scaling



Inner Product
B
take Eigenvalues, Eigenvectors, and
$y^{(i)} = \sqrt{\lambda_i} e^{(i)}$
Reconstructed Configuration
Y
lose orientation
$B = XX^T$
lose position
$B = -\frac{1}{2}(I_n - \mathbf{1}\mathbf{1}^T/n)\,E\,(I_n - \mathbf{1}\mathbf{1}^T/n)$,
$e_{m,\ell} = b_{m,m} + b_{\ell,\ell} - 2b_{m,\ell}$
Configuration
X
Distances
E
$e_{m,\ell} = \sum_{v=1}^{p}(X_{m,v} - X_{\ell,v})^2$,
$= X_{(m)}^T X_{(m)} + X_{(\ell)}^T X_{(\ell)} - 2X_{(m)}^T X_{(\ell)}$,
where $X_{(m)}^T$ is the mth row of X, a p-dimensional vector,
$m, \ell = 1, \ldots, n$.

Figure 2: Classical Scaling

Target: obtain configuration $\{X_{m,v}\}_{m=1,\cdots,n \;\; v=1,\cdots,p}$($m$th sample, vth dimension) from distances $\{E_{m,l}\}$ (distance between mth and lth point).

Requirements

- Distance is symmetric, Euclidean
- $\overline{X} = 0$

Euclidean distance:

$$e_{m,l}^2 = \sum_{v=1}^{p}(X_{m,v} - X_{l,v})^2 = X_{(m)}^T X_{(m)} + X_{(l)}^T X_{(l)} - 2X_{(m)}^T X_{(l)} = b_{m,m} + b_{l,l} - 2b_{m,l}$$

Inner product matrix: $B := XX^T$.

Information loss:
Rotation preserves inner product, so X $\rightarrow$ B loses orientation
Transition preserves distance, so X $\rightarrow$ E loses position.
Recovery requirement: $\overline{X} = n^{-1}\mathbf{1}_n^T X = 0$ (the recovery will be different if this is not satisfied), $\Leftrightarrow$ $\mathbf{1}_n$ is Eigenvector of B^T with Eigenvalue of 0.

Recover B from E:
By summing over m of $e_{m,l}$,

$$e_{\bullet,l} = b_{\bullet,\bullet} + nb_{l,l} - 2b_{\bullet,l} = b_{\bullet,\bullet} + nb_{l,l}$$

so $b_{l,l} = \frac{e_{\bullet,l} - b_{\bullet,\bullet}}{n}$. Summing over the other letter gives $b_{m,m} = \frac{e_{m,\bullet} - b_{\bullet,\bullet}}{n}$. Therefore,

$$b_{m,l} = \frac{b_{m,m} + b_{l,l} - e_{m,l}^2}{2} = -\frac{1}{2}(e_{m,l} - \frac{e_{m,\bullet}}{n} - \frac{e_{\bullet,l}}{n} + \frac{e_{\bullet,\bullet}}{n^2})$$

Matrix form:

$$B = -\frac{1}{2}(I_n - \mathbf{1}\mathbf{1}^T/n)E(I_n - \mathbf{1}\mathbf{1}^T/n)$$

Recover X from B:
B must be positive semi-definite and symmetric, so take Eigendecomposition

$$B = \sum_{i=1}^{n} \lambda_i e^{(i)} (e^{(i)})^T$$

assume WOLG first n' of λ_i are non-zero. Define

$$Y = \begin{pmatrix} \vdots & \cdots & \vdots \\ f^{(1)} & \cdots & f^{(n')} \\ \vdots & \cdots & \vdots \end{pmatrix}, \qquad \text{where } f^{(i)} := \sqrt{\lambda_i} e^{(i)}$$

$YY^T = B$.

Classical Scaling in Practice:

- Some Eigenvalues of B may be negative, means that distance E may not be Euclidean distance. In practice, only take the largest few positive Eigenvalues. This also indicates the underlying dimensions of data.
- Value of the $n' + 1$th dimension indicates the error.
- Choices of breakpoint n':
  - Use $n' = 2$ which is convenient for plotting
  - Place n' at a large drop of Eigenvalues
  - make $\sum_{i=1}^{n'} \lambda_i \approx \mathrm{tr}(B)$
  - pick all $\lambda_i > |\lambda_n|$.

2.2 Distances, Dissimilarities

For entries that are attributes instead of numbers, dissimilarities are used.

Metric $d(x, y)$ (good dissimilarity measure)

- $d(x, y) \geq 0$, and $d(x, x) = 0$.
- $d(x, y) = d(y, x)$ (symmetric)
- $d(x, y) + d(y, z) \geq d(x, z)$ (triangular inequality)

$\{d_\alpha\}_{\alpha \in \mathcal{A}}$ is family of metrics $\Rightarrow \sum_{\alpha \in \mathcal{A}} d_\alpha$ is metric.

Hamming distance: number of disagreements/mismatch of attributes. It is a metric.

Contingency table: Count the number of times each object has an attribute

		Attributes of x	
		Yes	No
Attributes of y	Yes	a	b
	No	c	d

Dissimilarity/Similarity measures

- (Hamming distance) $b + c$
- (Simple matching Coefficient) $(a + d)/(a + b + c + d)$
- (Jaccard Distance) $d_J(x, y) := (b + c)/(a + b + c)$.
- Maximum difference (L_∞ norm)
- Manhattan distance (L_1 norm)
- Canberra distance, weighted L_1 norm,

$$\sum_q \frac{|x_q - u_q|}{|x_q| + |u_q|}$$

- if the variable is ordinal/continuous:
$$S_{i,j} = 1 - \frac{|x_i - x_j|}{r}$$
  where r is the range of values
- if the variable is nominal $S_{i,j} = \delta_{x_i,x_j}$

Combination of metrics: Gower's similarity coefficients
$$S_{i,j} = \frac{\sum_q w_{i,j,q} S_{i,j,q}}{\sum_q w_{i,j,q}}$$
$w_{i,j,q}$ is confidence in variable q, $S_{i,j,q}$ is the similarity between i, j on variable q.

2.3 Non-metric Scaling

Given dissimilarities $\{\delta_{m,l}\}$, try to find a good configuration so that Euclidean distances $d_{m,l}$ is close to $\{\delta_{m,l}\}$.

Steps:

1. Re-order the data so that the dissimilarities $\{\delta_{m,l}\}$ are in increasing order.
2. Find a good initial configuration $x_{i,k}^{(0)}$
   - Random: each entry is taken from Unif($[0,1]$)
   - Kruskal: L-shaped
   - Classical Scaling: convert $\{\delta_{m,l}\}$ to numerical, use classical scaling to give a "rough" solution.

   Choice of dimensions of initial configuration: start with high dimensions, obtain the optimal configuration, then project to lower dimensions and keep iterating. Final dimensions cannot be 1, usually, $2, 3$ are chosen. But one can also look at the plot of optimal stress versus dimension and find the elbow.
3. Calculate distances $d_{m,l}$ from the initial configuration, and use least square monotone regression to obtain $\hat{d}_{m,l}$ which is ordered in the same way as $\{\delta_{m,l}\}$ (i.e. monotone increasing).
4. Calculate stress function: (measures how different the distance of created configuration is to the dissimilarities)
$$S(X) := \sqrt{\frac{S^*}{T^*}}, \quad S^* := \sum_{m<l} (d_{m,l} - \hat{d}_{m,l})^2, \; T^* := \sum_{m<l} d_{m,l}^2$$
5. Minimise stress function by an iterative algorithm. e.g. gradient descent

Monotone Regression: fix y_i, aims to minimise $\sum_i (y_i - z_i)^2$ subjected to $z_1 \le z_2 \le \cdots$ (monotone increasing)
Miles Algorithm: Put y_i into singleton blocks, scan through all blocks if there is a decreasing/constant jump, merge the two blocks and replace all of them with their mean. Stop when all jump between blocks are strictly increasing.
Young's Boundary Search Algorithm: Start with first two y_i, perform one scan of Miles Algorithm, then add next y_i and scan again.

Stress function: By the chain rule,
$$\frac{\partial S}{\partial x_{i,k}} = \frac{S}{2} \left(\frac{1}{S^*} \frac{\partial S^*}{\partial x_{i,k}} - \frac{1}{T^*} \frac{\partial T^*}{\partial x_{i,k}} \right)$$
Using definitions of S^*, T^* and chain rule again,
$$\frac{\partial T^*}{\partial x_{i,k}} = 2 \sum_{m<l} (x_{m,k} - x_{l,k}) \left(\frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{l,k}}{\partial x_{i,k}} \right), \quad \frac{\partial S^*}{\partial x_{i,k}} = \sum_{m<l} 2(d_{m,l} - \hat{d}_{m,l}) \frac{x_{m,k} - x_{l,k}}{d_{m,l}} \left(\frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{l,k}}{\partial x_{i,k}} \right)$$
For the derivative of S^*, $\partial \hat{d}_{m,l} / \partial x_{i,k}$ is involved, but since $\hat{d}$ is piece-wise constant by its design, this part turns out to be 0.

S is exactly once differentiable (higher derivatives not suitable for analysis)

2.4 Procrustes Analysis

Purpose

- Assess the performance of scaling methods
- compare two configurations produced by scaling
- match natural specimens to identify species

Given configurations $X, Y \in \mathbb{R}^{n\times K}$ where n is the number of samples, K is the number of features. Target: compare the two configurations. i.e. minimise

$$G(X,Y) = \sum_{k=1}^{K}\sum_{i=1}^{n}(X_{i,k} - Y_{i,k})^2$$

under translation, rotation, reflection, scaling or similarity operations.

Translation: Write out $G(X,Y)$ by bridging of $\overline{X}_k$ and $\overline{Y}_k$,

$$G(X,Y) = \sum_{i,k}(X_{i,k} - \overline{X}_k)^2 + \sum_{i,k}(Y_{i,k} - \overline{Y}_k)^2 + \sum_{i,k}(\overline{X}_k - \overline{Y}_k)^2 + \cdots + 2\sum_{i,k}(X_{i,k} - \overline{X}_k)(Y_{i,k} - \overline{Y}_k)$$

minimise w.r.t Y yields $\overline{Y}_k = \overline{X}_k$.

Rotation: if $A = YP$ where P is rotation matrix

$$G(X,A) = \sum_{i,k}(X_{i,k} - YP)^2 = \mathrm{tr}\{(X-A)^T(X-A)\} = \langle YP - X, YP - X\rangle_F$$

expand and use properties of trace (invariant under circulation),

$$G(X,A) = -2\langle P, YPY^TX\rangle_F = -2\langle U^TPV, \Sigma\rangle_F = -\sum_{i,i} S_{i,i}\Sigma_{i,i}$$

where $Y^TX = U\Sigma V^T$ is the SVD. $S := U^TPV$ is orthogonal so $|S_{i,i}| \leq 1$. Therefore, G is minimised when $S = I$, i.e. $P = UV^T$.

When this P is taken, distance $G(X,YP) = \|Y\|_F^2 + \|X\|_F^2 - 2\sum_i \Sigma_{i,i}$.

Scale: Minimise $G(X,\alpha Y) = a\alpha^2 + b\alpha + c$ where a, b, c can be found in terms of $X_{i,k}, Y_{i,k}$. Then differentiation yields

$$\alpha^* = \frac{\sum_{i,k} X_{i,k}Y_{i,k}}{\sum_{i,k} Y_{i,k}^2}$$

2.4.1 General Procrustes Analysis

Minimise

$$S_{\{X_l\}_{l=1}^L}(\{R_l\}_{l=1}^L, M) = \sum_{l=1}^{L}\|X_lR_l - M\|_F^2$$

Solution:

- Pick i, let $M := X_i$
- match all X_l to M, say the matched configurations are Y_l.
- update M by $M = L^{-1}\sum_{l=1}^L Y_l$
- repeat above two steps
- stop until convergence of $S_{\{X_l\}_{l=1}^L}(\{R_l\}_{l=1}^L, M)$.

2.5 R-related

Classical MDS: `cmdscale(dist, k, eig=TRUE)` where k is the number of Eigenvalues to keep. Return: the recovered matrix Y, Eigenvalues (of the recovered B matrix), the doubly centred distance matrix.

Kruskal's non-metric MDS is `isoMDS(d = distance, y=initial_configuration)` . Use `dist(t(data), method=...)` to construct various dissimilarity measures. Missing samples can be dealt (they are ignored when calculating stress function)

Procrustes Analysis: `Procrustes(X, Y)` in library `smacof` .

3 Supervised, Unsupervised Learning

3.1 Clustering Algorithms

K-mean Algorithm: Given configuration $X \in \mathbb{R}^{n\times p}$, the target is to divide n observations into K clusters.

- Start with chosen k initial centroids(mean vectors) $m_i^{(1)}$.
- (Iteration s) Assign each observation to the closest mean vector regarding Euclidean distance.

$$C_j^{(s)} := \left\{ X_i \,:\, \|X_i - m_j^{(s)}\|^2 \leq \|X_i - m_r^{(s)}\|^2 \forall r = 1, \cdots, k \right\}$$

- Update means from observations in the new clusters.

$$m_j^{(s+1)} := \left| C_j^{(s)} \right|^{-1} \sum_{X_i \in C_j^{(s)}} X_i$$

- Iterate until convergence

Self-Organising Maps: Target is the same as K-mean Clustering. Simply speaking, SOM forms a rectangular grid in the beginning. The grid points m_l are called prototypes. (the number of prototypes represents the number of clusters). Iterate through all data X_i, and for each data point, select the closest (winner) prototype, and pull the winner prototype and its neighbours together towards X_i via

$$m_l \mapsto m_l + \alpha(X_i - m_l)$$

α is the learning rate. The choice of the learning rate, neighbourhood region and initial prototypes are aspects of SOM that should be considered before implementation.

3.2 Projection Pursuit

Recall projection $y = Xa$, Instead of maximising variance $a^T Sa$ in PCA, PP maximises the degree of non-Gaussian (i.e. how multimodal the distribution is) to perform clustering.

K-L divergence of g from f: measures the difference between two distributions

$$D_{KL}(g||f) = \int_{\mathbb{R}} g(x) \log\left(\frac{g(x)}{f(x)}\right) dx = E_{X\sim g(x)}(\log(g/f))$$

Entropy: measures non-Gaussian

$$H(g) := -\int_{\mathbb{R}} g(x) \log(g(x))\, dx$$

Multivariate version:

$$H(g) = -\int_{\mathbb{R}^p} g(\boldsymbol{x}) \log(g(\boldsymbol{x}))\, d^p\boldsymbol{x}$$

Note $D_{KL}(g||f) = -H(g) - \int_{\mathbb{R}} g(x)\log(f(x))\,dx$.
If f is PDF of $\mathrm{N}(0,\sigma^2)$, then for any distribution g with variance σ^2, $H(f) \geq H(g)$.

Sphering making variance of individual variables 1 and make them uncorrelated. Recall sample variance $S = n^{-1}X^TX$, find $R = S^{-1/2}$, $W = XR$ is the sphered data. (i.e. $S_W := n^{-1}W^TW = I_p$). Also, $\mathrm{Var}(Wa) = a^TS_W = a^Ta = 1$, variance of projection in any direction is the same.

Projection Pursuit process:

- Begin with centred and sphered data W. (Usually have multiple random starts)
- pick initial projection direction a, find $u_a := Wa$
- Solve
$$\min_{a\,:\,a^Ta=1} H(\hat{f}_{U,a}(u))$$
where $\hat{f}_{U,a}(u)$ is a density estimate of f_U formed by $u_1, \cdots, u_n$

Note PCA uses centred and standardised data, but PP must use centred and sphered data.

3.3 Independent Component Analysis

If there are p individuals formed by p components, i.e. $X = SA^T$ where columns of $S \in \mathbb{R}^{p\times p}$ are the components.
If $X = UDV^T$, let $S = \sqrt{n}U$, $A^T = DV^T/\sqrt{n}$, then Cov(S)$= n^{-1}S^TS = I_p$ so the components are uncorrelated. But this decomposition is not unique.
When there are more factors than the number of samples, the above decomposition does not work.

Only if S is assumed to be statistically independent, and non-Gaussian, the independent component decomposition $X = AS$ is unique. And ICA begins with sphered data X so that A must be orthogonal.
Mutual Information $Y = (Y_1, \cdots, Y_p)$, Mutual information

$$I(Y) := \sum_{j=1}^{p} H(Y_j) - H(Y)$$

when Y_i are independent, $I(Y) = 0$.

In the case of ICA, let $Y := S = A^TX$, then

$$I(Y) = \sum_{j=1}^{p} H(Y_j) - H(X)$$

this is minimised over all orthogonal A.

3.4 Projection Pursuit Regression

Use the model

$$f(X) = \sum_{m=1}^{M} g_m(\omega_m^TX)$$

where both projection directions ω_m and functions g_m are to be fitted.
Error

$$E = \sum_{i=1}^{n}\left(y_i - \sum_{m=1}^{M} g_m(\omega_m^Tx_i)\right)^2$$

Suppose ω_m are already found, minimisation of E is a smoothing problem (e.g. use smoothing spline)
After finding g, use linear approximation

$$g(\omega^Tx_i) \approx g(\omega_{\text{old}}^Tx_i) + g'(\omega_{\text{old}}^Tx_i)(\omega - \omega_{\text{old}})^Tx_i$$

The error can be written as

$$E \approx \sum_{i=1}^{n} g'(\omega_{\text{old}}^T x_i)^2 \left[\left\{ \omega_{\text{old}}^T x_i + \frac{y_i - g(\omega_{\text{old}}^T x_i)}{g'(\omega_{\text{old}}^T x_i)} \right\} - \omega^T x_i \right]^2$$

which is weighted least square regression.

PPR is an alternating algorithm performed by repeating the above two steps.

3.5 Neural Networks

Single Layer Neural Network: Input $\{X_i\}_{i=1,\cdots,p}$, hidden units $\{Z_m\}_{m=1,\cdots,M}$, $\{Y_k\}_{k=1,\cdots,K}$
Forward propagation:

$$Z_m = \sigma(\alpha_{0,m} + \alpha_m^T X)$$
$$T_k = \beta_{0,k} + \beta_k^T Z$$
$$\hat{Y}_k = f_k(X) = g_k(T_k)$$

where activation function

$$\sigma(v) := \frac{1}{1 + e^{-v}}$$

for regression: $g_k(T) = T$ for all k
for classification, use Softmax

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}$$

Neural Network is similar to PPR, but PPR uses free g_m for estimation, whereas activation function σ is fixed for the neural network.

Residual:

$$R(\theta) = \sum_{i=1}^{n} R_i(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} (y_{i,k} - f_k(x_i))^2$$

let $z_{m,i} = \sigma(\alpha_{0,m} + \alpha_m^T x_i)$, let $z_i = (z_{1,i}, \cdots, z_{M,i})$. Neural network fitted by gradient descent: assume $\alpha_{0,m}, \beta_{0,k} = 0$ (no intercept), $\beta_k = (\beta_{k,m})_{m=1,\cdots,M}$, $\alpha_m = (\alpha_{m,l})_{l=1,2,\cdots,p}$

$$\frac{\partial R_i(\theta)}{\partial \beta_{k,m}} = \underbrace{-2(y_{i,k} - f_k(x_i)) g_k'(\beta_k^T z_i)}_{=:\delta_{k,i}} z_{m,i}$$

$$\frac{\partial R_i(\theta)}{\partial \alpha_{m,l}} = \underbrace{-2 \sum_{k=1}^{K} (y_{i,k} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{k,m} \sigma'(\alpha_m^T x_i)}_{=:s_{m,i}} x_{i,l}$$

note

$$s_{m,i} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \delta_{k,i} \beta_{k,m}$$

Fitting Neural Networks:

- Could be helpful to centralise and standardise data first.
- In practice, the number of hidden layers could be higher
- Choose initial parameter values, usually random, small (close to 0)
- go through forward propagation, obtaining $\hat{f}_k(x_i)$

- Back-propagation: use the fitted values to find errors $\delta_{k,i}$, then use

$$s_{m,i} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \delta_{k,i}\beta_{k,m}$$

- Find the partial derivatives and use gradient descent to update parameters

$$\beta_{k,m} \mapsto \beta_{k,m} - \gamma_r \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial \beta_{k,m}}$$

$$\alpha_{k,m} \mapsto \beta_{k,m} - \gamma_r \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial \alpha_{k,m}}$$

  where γ_r is the learning rate at iteration r.

- repeat forward propagation and backward propagation.

3.6 Tree

Split the parameter space into rectangles $R_1, \cdots, R_m$, and predict output based on these rectangles. e.g. for regression,

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

where c_m are constants.
Error:

$$\text{SSQ} = \sum_{i=1}^{n} (Y_i - f(X_i))^2$$

Minimiser for c_m:

$$\hat{c}_m = \frac{1}{n_i} \sum_{i\,:\,X_i \in R_m} Y_i$$

Search for best split: For simplicity, only binary split along explanatory variables are considered, i.e. split by $X_i \leq t_i$ or $X_i > t_i$. The best split is searched by the greedy algorithm.
Define $R_1(j,s) := \{x \,|\, X_j < s\}, R_2(j,s) := \{x \,|\, X_j > s\}$, seek j, s that minimises

$$\min_{c_1} \sum_{X_i \in R_1(j,s)} (Y_i - c_1)^2 + \min_{c_2} \sum_{X_i \in R_2(j,s)} (Y_i - c_2)^2$$

note the two minimisation problems are solved by taking c_1, c_2 as the average value over the region. Fix j, the sum only changes when s crosses a data X_i. So only have to search among the $n+1$ constant values. Do this for every j and find the minimum point.

Stopping usually stop when all leaf nodes contains 5 points X_i or less.
Cost-complexity Pruning
Let $|T|$ be the number of leaf nodes of a tree,

$$Q_m(T) := n_m^{-1} \sum_{X_i \in R_m} (Y_i - \hat{c}_m)$$

where $\hat{c}_m$ is average of Y_i over region R_m and n_m is number of points X_i in R_m.
Cost-Complexity:

$$C_\alpha(T) = \sum_{m=1}^{|T|} n_m Q_m(T) + \alpha|T|$$

Over-fitting: $|T|$ too large.
Under-fitting: error $\sum_{m=1}^{|T|} n_m Q_m(T)$ too large.

Given a grown large tree T_0, search among all sub-trees T (formed by merging nodes together) of T_0 for minimiser of cost-complexity. α controls the balance between fitting data and the complexity of the tree, the best value of α can be found by cross-validation.

Weakest Link Pruning Sequentially collapse two internal nodes with the smallest increase in $\sum_{m=1}^{|T|} n_m Q_m(T)$, until the tree is merged into a single tree. Then search along this sequence of trees for the minimiser of $C_\alpha(T)$.

Classification Tree Prediction of the region R_m replaced by

$$k(m) = \text{argmax}_k \hat{p}_{m,k}$$

where proportion of class k in node R_m $\hat{p}_{m,k} := n_m^{-1} \sum_{X_i \in R_m} I(Y_i = k)$.

Measures of Classification error:
Mis-classification

$$n_m^{-1} \sum_{i\,:\,X_i \in R_m} I(Y_i \neq k(m)) = 1 - \hat{p}_{m,k(m)}$$

Gini index

$$\sum_{k \neq k'} \hat{p}_{m,k}\hat{p}_{m,k'} = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k})$$

Cross-entropy

$$-\sum_{k=1}^{K} \hat{p}_{m,k} \log(\hat{p}_{m,k})$$

3.7 Random Forest

Bootstrap
Suppose $\{X_i\}_{i=1,\cdots,n}$ are sample from distribution F, empirical distribution is

$$\hat{F}_n(x) := n^{-1} \sum_{i=1}^{n} I(X_i \leq x)$$

it has derivative

$$d\hat{F}_n(x) = n^{-1} \sum_{i=1}^{n} \delta(x - X_i)\, dx$$

where $\delta(x)$ is Dirac delta function. i.e. Each point X_i is given equal weight $1/n$. Bootstrap sample: can sample uniformly n times from $\{X_i\}_{i=1,\cdots,n}$. Multiple bootstrap samples: $\{X_i^{(b)}\}_{i=1,\cdots,n}$ where $b = 1, \cdots, B$.

Bagging For regression, can take Bootstrap samples $Z^{(b)}$ for $b = 1, \cdots, B$, and use them to train models $\hat{f}^{(b)}$. Aggregation/Bagging:

$$\hat{f}_{\text{bag}}(x) = B^{-1} \sum_{b=1}^{B} \hat{f}^{(b)}(x)$$

True bagging estimator: $\hat{f}_{\text{ag}}(x) := E_{\hat{P}}(\hat{f}^*(x))$ where $\hat{P}$ is uniform distribution on $\{(x_i, y_i)\}_{i=1,\cdots,n}$ and $\hat{f}$ is the model trained from $\{(x_i^*, y_i^*)\}_{i=1,\cdots,n}$ samples from $\hat{P}$. $\hat{f}_{\text{bag}}(x) \to \hat{f}_{\text{ag}}(x)$ as $B \to \infty$.

Bagging works because

$$E_P\left[(Y - \hat{f}^*(x))^2\right] \geq E_P\left[(Y - f_{\text{ag}}(x))^2\right]$$

(this is proved by bridging using f_{ag}) so bagging decreases MSE.

Bootstrap correlation Suppose $W_1, \cdots, W_B$ are i.i.d. with variance σ^2. Then

$$\text{Var}(\overline{W}) = \frac{\sigma^2}{B^{-1}} \to 0 \quad \text{as } B \to \infty$$

but when $\mathrm{Cov}(W_b, W_d) = \sigma^2\rho > 0$ when $b \neq d$, then

$$\mathrm{Var}(\overline{W}) = B^{-2}\left\{\sum_{b=1}^{B}\mathrm{Cov}(W_b, W_b) + \sum_{b=1}^{B}\sum_{d=1,\, d\neq b}^{B}\mathrm{Cov}(W_b, W_d)\right\} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

does not tend to 0 as $B \to \infty$.
To reduce correlation, select m of p variables and pick the best split from the m selected variables. Random forest is the combination of bagging and this variable selection technique.

Out-of-Bag Sample
For each observation, (x_i, y_i), can construct a predictor by averaging only the trees constructed from bootstrap samples without z_i. Out-of-bag errors refer to the errors of these predictors.
A $N \times N$ proximity matrix can be accumulated for the training data, i.e. for each tree, increase proximity by 1 if they are in the same leaf node.
Matrix of dissimilarities: $d = 1 - p/\max p$, then pass D to MDS to recover a configuration called *Proximity Maps*.

3.8 Boosting

Boosting combines weak classifiers (e.g. weak trees are trees that have only one split) to make a strong classifier. Only the Adaboost.M1 will be introduced. Consider a binary classification problem where the only two classes are $\{\pm 1\}$.
Step 1. First give equal weights $w_i = n^{-1}$ to observations $\{x_i\}$. Sample weights indicate how much the classifiers should focus on classifying these samples right.
Step 2. Fit weak classifiers $G_m(x)$ according to the weights. (e.g. for classification tree, weighted Gini index or weighted entropy is used)
Step 3. Find the error rate for classifier G_m:

$$e_m = \frac{\sum_{i=1}^{n} w_i I(y_i \neq G_m(x_i))}{\sum_i w_i}$$

Find the importance of G_m:

$$\alpha_m := \log\left(\frac{1-e_m}{e_m}\right)$$

- $e_m < 0.5$: $\alpha_m > 0$, and classifier with lower error gets higher weight
- $e_m = 0.5$: $\alpha_m = 0$, the classifier is not different from the random classifier, does not give weight
- $e_m > 0.5$: $\alpha_m < 0$, inverting this classifier would make a good classifier, so let the weight be negative

Step 4. Update sample weights:

$$w_i \mapsto w_i e^{\alpha_m I(y_i \neq G(x_i))}$$

- G_m with positive α_m: up-weigh the samples that are not correctly classified by e^{α_m}.
- G_m with negative α_m (inverted classifiers): down-weigh the samples that are not correctly classified. equivalent to up-weighing the correctly classified ones.

Repeat the above steps until convergence, the output (strong) classifier is

$$G^*(x) := \mathrm{sign}\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right)$$

3.9 R-related

K-mean: `kmean(x=data, centers=..., nstart=100)` where `centers` takes the initial guess of mean vectors.

SOM: `som(data, somgrid(...))` in library `kohonen` . `somgrid` constructs a grid of prototypes.

Projection pursuit: `PP3many(data, nrandstarts=100)` it performs PP using 100 random starts, and picks the best one.
Or use the simple function `ppr(formulae, data, nterms, max.terms)` . `nterms` indicates the number of terms in the PPR expansion.

ICA: `fastICA(X, p, ...)` in library `fastICA` where p is the number of components.

Neural Network: `neuralnet(f=formulae, data, hidden=c(5, 3), linear.output=TRUE)` hidden specifies number of neurons in each hidden layer. If `linear.output` is false, smoothing will be applied to the output.

Build Tree: `rpart(formulae, data=...)` from library `rpart` . The returned value is an object in the class `rpart` , which can be plotted by `plot(tree)` or `fancyRpartPlot(tree)` . Prediction: `rpart.predict(object=tree, newdata=...)` .

4 Data Ethics

- Think things through
- makes no harm to others, also protect yourself
- ask for permission/authorisation
- be transparent, open
- Do the right things

5 Basic Math Techniques

Jacobin of transformation: if $Y = AX$

$$g_Y(y) = f_X(x_1(y), \cdots, x_p(y)) \left| \frac{\partial x}{\partial y} \right|$$

Change of variable of multivariate integral:

$$\int_{\mathbb{R}^p} f(X)\, d^p x = \int_{\mathbb{R}^p} f(A^T y) |\det(A^T)|\, d^p y$$

Inequality $\log(r) \leq r - 1$

The sum of Eigenvalues is the trace, product of Eigenvalues is the determinant.
$\mathrm{Tr}(ABC) = \mathrm{Tr}(CAB) = \mathrm{Tr}(BCA)$ (trace invariant under circular shift.

Orthogonal matrix:

- The product of orthogonal matrices is orthogonal,
- transpose of an orthogonal matrix is orthogonal
- determinant is ± 1
- Eigenvalues are ± 1
- 2-norm of every column is 1
- always invertible
- rotation gives rise to a unique orthogonal matrix
- magnitude of all entries bounded by 1