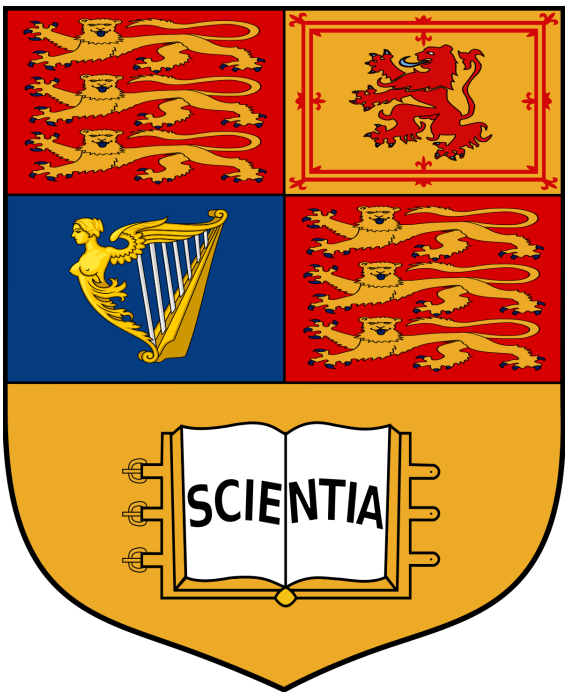
Intro to Statistical Learning Concise Notes

MATH60049

Arnav Singh



Content from prior years assumed to be known.

Mathematics
Imperial College London
United Kingdom
April 28, 2023

Contents

3 Linear Methods for Regression

3.2 Linear Regression Models and Least Squares

Definition 3.2.1. *The **linear regression model** is given by*

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{1}$$

where $X^T = (X_1, \ldots, X_p)$ is the input vector, $\beta^T = (\beta_0, \ldots, \beta_p)$ is the unknown parameter vector, and $f(X)$ is the output.

Linear model assumes that regression function $E(Y \mid X)$ is linear, or it is a reasonable approximation.

Definition 3.2.2. *Multivariate Linear Model Given p explanatory variables, $X = (X_1, \ldots, X_p)$, we have the multivariate linear model*

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \ldots + \beta_p X_{i,p} + \epsilon_i = \beta_0 + \sum_{j=1}^{p} X_{ij}\beta_j + \epsilon_i \tag{2}$$

Given $\beta = \left(\beta_1, \ldots, \beta_p\right)$ and $X = (X_{i,j})$ the $n \times p$ matrix of $X_{i,j}$ values of X on ith individual and jth variable.

$$Y = X\beta + \epsilon$$

ϵ the $n \times 1$ vector of errors, with mean 0 and variance σ^2.

Definition 3.2.3. *(Residuals)*

$$e = Y - X\beta$$

(Residual Sum of Squares)

$$RSS(\beta) = e^T e = (Y - X\beta)^T (Y - X\beta) = Y^T Y - 2\beta^T X^T Y + \beta^T X^T X \beta$$

(Least Squares Estimator)

$$\hat{\beta} = \arg\min_{\beta} \; RSS(\beta) = \underbrace{(X^T X)^{-1} X^T Y}_{H \text{ - Hat matrix}}$$

We have the fitted values of the model as

$$\hat{Y} = X\hat{\beta} = HY$$

Proposition 3.2.4. *(Expectation and Variance of β)*

$$E(\hat{\beta}) = \beta$$

$$Var(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$$

$Var\hat{\beta}$ a $p \times p$ covariance matrix. If X an orthogonal matrix then $X^T X = I_p$ so $Var\hat{\beta}$ a diagonal matrix so for $i \neq j, \hat{\beta}_i, \hat{\beta}_j$ are uncorrelated

Proposition 3.2.5. *(Distribution of $\hat{\beta}$) If $\{e_i\}$ are normally distributed, then $\hat{\beta}$ is also normal*

$$\hat{\beta} \sim N_p(\beta, \sigma^2 (X^T X)^{-1})$$

And if X orthogonal then $\hat{\beta}_i$ and $\hat{\beta}_j$ are independent for $i \neq j$

Definition 3.2.6. *(Z-score)*

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

where v_j the jth diagonal element of $(X^T X)^{-1}$ and so var $\hat{\beta}_j = \sigma^2 v_j$

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \quad E(\hat{\sigma}^2) = \sigma^2$$

Used to test the hypothesis that $\beta_j = 0$, we have that z_j distributed as a Student's t distribution with $n - p$ degrees of freedom, approximated as $N(0, 1)$ for large n larger than p

Definition 3.2.7. *(F-test)*

$$F = \frac{(RSS_0 - RSS_1)/(p_1 - p_0)}{RSS_1/(N - p_1 - 1)}$$

where RSS_0 the residual sum of squares for the smaller model with $p_0 + 1$ parameters, RSS_1 the residual sum of squares for the larger model with $p_1 + 1$ parameters, and N the number of observations.

$$F \sim F_{p_1 - p_0, n-p-1-1}$$

The F statistic measures the change in residual sum-of-squares per additional parameter in the bigger model, and it is normalized by an estimate of σ^2

3.2.2 The Gauss-Markov Theorem

Theorem 3.2.8. *(Gauss-Markov Theorem) If the errors ϵ_i have mean 0, variance σ^2, and are uncorrelated, then the least squares estimator $\hat{\beta}$ is the best linear unbiased estimator of β (BLUE)*

3.3 Subset Selection

Unsatisfied with the least squares estimates due to

- *prediction accuracy* - the least squares estimates often have low bias but large variance.
- *interpretation* - With a large number of predictors, we often would like to determine a smaller subset that exhibit the strongest effects. In order to get the "big picture," we are willing to sacrifice some of the small details

3.4 Shrinkage Methods

3.4.1 Ridge Regression

Definition 3.4.1. *(Ridge Regression)*

$$\begin{aligned}\hat{\beta}_R &= \arg\min_{\beta}\left\{\sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p}X_{i,j}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\} \\ &= \arg\min_{\beta}\sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p}X_{i,j}\beta_j\right)^2 \quad \textit{subject to } \sum_{j=1}^{p}\beta_j^2 \le t\end{aligned}$$

λ a tuning parameter that controls the amount of shrinkage
When variables in least squares are highly correlated it can cause instability in the least squares estimators - ridge regression can help with this.

Proposition 3.4.2. *(Ridge Regression RSS in matrix form)*

$$RSS(\beta) = (Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta$$

Differentiating, setting to 0 and solving this for β gives

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

Definition 3.4.3. *(Condition number)*

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

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of A
$\kappa(A)$ is a measure of how much the solution of a linear system of equations changes when the coefficients of the equations are slightly changed.

Proposition 3.4.4. *(Bayesian interpretation of ridge regression)*
If prior $\beta_j \sim N(0, \tau^2)$ independently for $j = 1, \ldots, p$, and $Y_i \sim N(\beta_0 + x_i^T\beta, \sigma^2)$ Then $\beta \mid Y \sim N$ with posterior mean $\hat{\beta}_R$ with $\lambda = \sigma^2/\tau^2$

Singular Value Decomposition

Definition 3.4.5. *(Singular Value Decomposition) The SVD of a $n \times p$ (centred) matrix X is*

$$X = UDV^T$$

where U, V are $n \times p, p \times p$ orthogonal matrices - columns of U spanning the column space of X and V spanning the row space
D is an $p \times p$ diagonal matrix with non-negative entries - called the singular values of X
If one or more of $d_j = 0$ then X is singular.

$$\begin{aligned} \text{if } X &= UDV^T \\ X^TX &= VD^2V^T \\ (X^TX)^{-1} &= (V^T)^{-1}D^{-2}V^{-1} \end{aligned}$$

Can write the least squares fitted vector as

$$X\hat{\beta}_{ls} = UU^TY$$

And for ridge regression

$$\begin{aligned} X\hat{\beta}_R &= (UDV^T)(VD^2V^T + \lambda I_p)^{-1}(VD^TU^TY) \\ &= UD(D^2 + \lambda I_p)^{-1}DU^TY \\ &= \sum_{j=1}^{p} \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda}\mathbf{u}_j^TY \quad \mathbf{u}_j \text{ cols of } U \end{aligned}$$

Like linear regression, ridge regression computes the coordinates of y with respect to the orthonormal basis U . It then shrinks these coordinates by the factors $\frac{d_j^2}{d_j^2+\lambda}$ This means that a greater amount of shrinkage is applied to the coordinates of basis vectors with smaller d_j^2

Ridge regression shrinks coordinates in directions with smaller variance.

Definition 3.4.6. *(Effective) Degrees of freedom*

$$H_\lambda = X(X^TX + \lambda I_p)^{-1}X^T$$

is the "effective" hat matrix for ridge regression.

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

$$df(0) = p$$

which is the actual degrees of freedom of the least squares fit.

Proposition 3.4.7. *(Bias and variance of ridge regression)*
We have that

$$E(\hat{\beta}_R) = (X^TX + \lambda I_p)^{-1}X^TX\beta$$

3.4.2 The Lasso

Definition 3.4.8. *(The Lasso)*

$$\begin{aligned} \hat{\beta}_L &= \arg\min_\beta \left\{ \frac{1}{2}\sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \right\} \\ &= \arg\min_\beta \sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j\right)^2 \quad \textit{subject to } \sum_{j=1}^{p}|\beta_j| \le t \end{aligned}$$

λ a tuning parameter that balances between fidelity of the fit and size of the coefficients
Note

$$\hat{\beta}_L(0) = \hat{\beta} \quad \hat{\beta}_L(\lambda) \to \overline{Y} \textit{ as } \lambda \to \infty \quad \beta_j \to 0 \textit{ for } j \neq 0$$

3.4.3 Discussion: Subset Selection, Ridge Regression and the Lasso

Lasso v. Ridge

$$\text{Different penalties } \sum \beta_j^2 \text{ v. } \sum |\beta_j|$$

If β_j small then β_j^2 tiny, but $|\beta_j|$ can be considerable larger so is penalised more.
Lasso has the ability to select variables and shrink coefficients.

X orthogonal

$$\begin{aligned} \hat{\beta}_{ls} &= (X^TX)^{-1}X^TY = X^TY \\ \hat{\beta}_R &= \hat{\beta}_{ls}/(1+\lambda) \\ \hat{\beta}_L &= (\hat{\beta}_{ls} - \lambda) \\ &= (\hat{\beta}_{ls} - \lambda)^+ \\ &= \text{sign}(\hat{\beta}_{ls})(\left|\hat{\beta}_{ls}\right| - \lambda)^+ \end{aligned}$$

3.5 Methods using Derived Input Directions

3.5.1 Principal Components Regression

Definition 3.5.1. *(Principal Components Regression)*

1. *Compute the first M principal components $Z_1, \ldots, Z_M$ of X*

$$z_m = Xv_m \quad m = 1, \ldots, p$$

Where V from the SVD - the eigenvectors of the sample covariance matrix

2. *Regress Y onto $Z_1, \ldots, Z_M$ by least squares*

PCR can be written as sum of univariate regressions

$$\begin{aligned} \hat{y}_M^{pcr} &= \overline{Y}\mathbf{1}_n + \sum_{m=1}^{M} \hat{\theta}_m z_m \\ &= \overline{Y}\mathbf{1}_n + X\sum_{m=1}^{M} \hat{\theta}_m v_m \end{aligned}$$

where $\hat{\theta}_m = \frac{\langle z_m, y \rangle}{\langle z_m, z_m \rangle}$
Can think of

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

Usually operate pcr on scaled inputs
If $M = p$ then back to least squares (same col. space, just a rotation)
Works by discarding the $p - M$ smallest components

14 Cluster Analysis

14.1 Scaling

Definition 14.1.1. *(Scaling Matrices)*
Define the following

$$\begin{aligned} e_{m,l}^2 &= \sum_{v=1}^{p} (X_{m,v} - X_{l,v})^2 \\ E &= (e_{m,l}) \quad \text{a } n \times n \text{ matrix} \\ e_{m,l} &= X_{(m)}^T X_{(m)} + X_{(l)}^T X_{(l)} - 2X_{(m)}^T X_{(l)} \end{aligned}$$

We form the inner product matrix

$$\begin{aligned} B_X &= XX^T \\ e_{m,l} &= b_{m,m} + b_{l,l} - 2b_{m,l} \quad m, l = 1, \ldots, n \end{aligned}$$

Proposition 14.1.2. *(Loss of information) - Rotation*
For $Y = XP$, P an orthogonal $p \times p$ matrix - a rotation matrix

$$B_Y = YY^T = XP(XP)^T = XX^T = B_X$$

Going from $X \to B$ we lose orientation information.

Proposition 14.1.3. *(Loss of information) - Position*
Suppose $W_{(m)} = X_{(m)} - \mu$ *for* μ *a arbitrary p-vector*

$$W_{(m)}^T W_{(l)} = X_{(m)}^T X_{(l)} - X_{(m)}^T \mu - \mu^T X_{(l)} + \mu^T \mu$$

Forming distances using W

$$\begin{aligned} e_{m,l}^{(W)} &= W_{(m)}^T W_{(m)} + W_{(l)}^T W_{(l)} - 2W_{(m)}^T W_{(l)} \\ &= e_{m,l} \end{aligned}$$

So going from B *to* E *we lose position information.*

Theorem 14.1.4. *(Recovering* B *from* E *)*

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

We have that $\boldsymbol{1}_n$ *an eigenvector of* B *with eigenvalue* 0

Theorem 14.1.5. *(Recovering* X *from* B *)*
B *a* $n \times n$ *matrix,* $B = XX^T$*, so* B *is positive semi-definite and symmetric - so we can form the following eigen-decomposition*

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

For $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{n'}$ *and* $\lambda_{n'+1}, \ldots, \lambda_n = 0$ *and* $\{e^{(i)}\}$ *are the eigenvectors of* B*.*
Define the following n-vectors f

$$f^{(i)} = \sqrt{\lambda_i} e^{(i)} \quad i = 1, \ldots, n'$$

Then we have that

$$Y_{n \times n'} = \begin{pmatrix} \vdots & \vdots & \ldots & \vdots \\ f^{(1)} & f^{(2)} & \ldots & f^{(n')} \\ \vdots & \vdots & \ldots & \vdots \end{pmatrix}$$

Where $YY^T = XX^T = B$

14.3 Proximity Matrices

14.3.1 Proximity Matrices

Definition 14.3.1. *(Distance Matrix)*

$$\mathbf{D} = (d_{ij}) = d(x_i, x_j)$$

A $N \times N$ *matrix* $\mathbf{D}$ *where* N *is the number of observations.*
Assume $d_{ij} \geq 0$ *and* $d_{ii} = 0$*.*
Assume $\mathbf{D}$ *also symmetric, if not consider* $(\mathbf{D} + \mathbf{D}^T)/2$ *instead for analysis.*

14.3.2 Dissimilarites Based on Attributes

Definition 14.3.2. *(Dissimilarity)*
For measurements x_{ij} *for* $i = 1, 2 \ldots, N$ *on variables* $j = 1, 2, \ldots, p$ *(attributes) - define the dissimilarity*

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j})$$

where d_j *is the dissimilarity on the* j*th attribute.*
Often taken as

$$d_j(x_{ij} x_{i'j}) = (x_{ij} - x_{i'j})^2$$

14.3.3 Object Dissimilarity

Definition 14.3.3. *(Object Dissimilarity)*

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot d_j(x_{ij}, x_{i'j}); \quad \sum_{j=1}^{p} w_j = 1$$

where d_j is the dissimilarity on the jth attribute.
w_j is the weight assigned to the jth variable.

$$\overline{D} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot \overline{d}_j \qquad \overline{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_j(x_{ij}, x_{i'j})$$

Setting $w_j \sim \overline{d}_j$ would give all attributes equal influence.
In the Euclidean Case

$$D_I(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot (x_{ij} = x_{i'j})^2$$

$$\overline{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} (x_{ij} - x_{i'j})^2 = 2 \cdot var_j = \text{ sample estimate of } Var(X_j)$$

14.3.6 K-means

A popular iterative descent clustering method - used where all variables quantitative and Euclidean distance used.

Definition 14.3.4. *(Within-point scatter)*

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2$$

$$= \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \overline{x}_k||^2$$

Where $\overline{x}_k = \left(\overline{x}_1 k, \ldots, \overline{x}_n k\right)$ the mean vector of cluster k, $N_k = \sum_{i=1}^{N} I(C(i) = k)$

Algorithm 14.1. *(K-means Clustering)*

- *For a given cluster assignment C , the total cluster variance (33) is minimized with respect to $\{m_1, \ldots, m_K\}$ yielding the means of the currently assigned clusters (32)*

$$\min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - m_k||^2 \tag{33}$$

$$\overline{x}_S = \arg\min_{m} \sum_{i \in S} ||x_i - m||^2 \tag{32}$$

- *Given a current set of means $\{m_1, \ldots, m_K\}$,(33) is minimized by assigning each observation to the closest cluster mean. That is,*

$$C(i) = \arg\min_{1 \le k \le K} ||x_i - m_k||^2 \tag{34}$$

- *Iterate 1 and 2 until the assignments stop changing.*

14.8 Multidimensional Scaling

Definition 14.8.1. *Given configuration and set of dissimilarities $\{\delta_{m,l}\}$ we can form the distances $\{d_{m,l}\}$ and least squares monotone regression fit $\{\hat{d}_{m,l}\}$ to the d using the δs*
Let the residual sum of squares be

$$S^* = \sum_{m<l} \left(d_{m,l} - \hat{d}_{m,l}\right)^2$$

And

$$T^* = \sum_{m<l} d_{m,l}^2$$

Then the stress of the configuration is

$$S = \sqrt{\frac{S^*}{T^*}}$$

Theorem 14.8.2. *(Differentiation of Stress)*
The stress function S is differentiable and the partial derivatives are

$$\begin{aligned}
\frac{\partial S}{\partial x_{i,k}} &= \frac{1}{2S}\left\{\frac{T^*\frac{\partial S^*}{\partial x_{i,k}} - S^*\frac{\partial T^*}{\partial x_{i,k}}}{(T^*)^2}\right\} \\
&= \frac{S}{2}\left\{\frac{1}{S^*}\frac{\partial S^*}{\partial x_{i,k}} - \frac{1}{T^*}\frac{\partial T^*}{\partial x_{i,k}}\right\} \\
&= \frac{S}{2}\left[\frac{1}{S^*}\left\{\sum_{m<l} 2(d_{m,l} - \hat{d}_{m,l})\frac{\partial d_{m,l}}{\partial x_{i,k}}\right\} - \frac{2}{T^*}\left\{\sum_{m<l}(x_{m,k} - x_{l,k})\left(\frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{l,k}}{\partial x_{i,k}}\right)\right\}\right]
\end{aligned}$$

Theorem 14.8.3. *(Derivative of T^*)*

$$\frac{\partial T^*}{\partial x_{i,k}} = 2\sum_{m<l}(x_{m,k} - x_{l,k})\left\{\frac{\partial x_{m,k}}{\partial x_{i,k}} - \frac{\partial x_{l,k}}{\partial x_{i,k}}\right\}$$

Theorem 14.8.4. *(Derivative of S^*)*

$$\frac{\partial S^*}{\partial x_{i,k}} = \sum_{m<l} 2(d_{m,l} - \hat{d}_{m,l})\frac{\partial d_{m,l}}{\partial x_{i,k}} - \sum_{m<l} 2\left(d_{m,l} - \hat{d}_{m,l}\right)\frac{\partial \hat{d}_{m,l}}{\partial x_{i,k}}$$

14.4 Self-Organizing Maps

Definition 14.4.1. *(SOM)*
Consider a SOM with 2-dimensional rectangular grid of K prototypes $m_j \in \mathbb{R}^p$ - each prototype parametrized w.r.t an integer coordinate pair $\ell_j \in \mathcal{Q}_1 \times \mathcal{Q}_2$ where $\mathcal{Q}_i = \{1, \ldots, q_i\}$

Algorithm 14.2. *(SOM Algorithm)*

1. *Sample $X_i \in \mathbb{R}^p$*
2. *Find the prototype m_j that is closest to X_i and the closest neighbours m_ℓ of m_j - determined by distance r*
3. *move neighbours m_k towards X_i via*

$$m_j \leftarrow m_j + \alpha(X_i - m_j)$$

Definition 14.4.2. *(Closeness of configurations)*
Given X, Y measure their closeness using

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

Aim to minimise G under the following group actions: translation group, Euclidean group and similarity group.
Done in order

1. *Match translation - by matching the centroids.*
2. *Match Rotation*
3. *Match Scale change*

Definition 14.4.3. *(Frobenius Norm)*

$$\langle A, B\rangle_F = \sum_{i,j}\overline{A}_{i,j}B_{i,j} = \mathrm{tr}(\overline{A}^T B)$$

Consider only real-valued matrices we have

$$||A||_F^2 = \langle A, A\rangle_F$$

Proposition 14.4.4. *(Norm Form)*
Given X and $A = YP$ configurations, where P a rotation matrix.

$$G(X,A) = ||X - A||_F^2$$

So we have to minimise G we find P^*

$$\begin{aligned} P^* &= \arg\max_P \left\langle P, Y^T X\right\rangle_F \\ U\Sigma V^T &= Y^T X \ \textit{(the SVD of } Y^T X) \\ P^* &= VU^T \end{aligned}$$

Definition 14.4.5. *(Procrustes Distance)*
The minimised distance when $P = P^*$

$$||YP^* - X||_F^2 = \|Y\|_F^2 + \|X\|_F^2 - 2\underbrace{\langle I, \Sigma\rangle_F}_{=\mathrm{tr}(\Sigma)}$$

Definition 14.4.6. *(Generalised Procrustes Analysis Algorithm)*

1. *Choose arbitrary configuration X_i and set $M = X_i$*
2. *Use Procrustes Analysis to match all configurations to M - giving matched configurations $\{Y_i\}_{i=1}^L$ and matching matrices $\{R_\ell\}_\ell^L$*
3. *Compute mean shape of all matched configurations*

$$M \leftarrow L^{-1}\sum_{i=1}^{L} Y_i$$

4. *Compute $S_{\{X_\ell\}_{\ell=1}^L}(\{R_{\ell=1}^L\}, M)$ and check convergence, if not go to 2.*

$$\textit{Defined:} \quad S_{\{X_\ell\}_{\ell=1}^L}(\{R_{\ell=1}^L\}, M) = \sum_{\ell=1}^{L}\|X_\ell R_\ell - M\|_F^2$$

5 Basis Expansions and Regularizations

5.1 Introduction

Definition 5.1.1. *(Linear basis expansion)*
Denote by

$$h_m(X) : \mathbb{R}^p \mapsto \mathbb{R}$$

the mth transformation of X, $m = 1, \ldots, M$
Then model

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

Examples

- $h_m(X) = X_m$ - *linear regression*
- $h_m(X) = X_m^2$ - *quadratic regression*
- $h_m(X) = \log(X_m), \sqrt{X_j}$
- $h_m(X) = \mathbb{I}(L_m \leq X_m \leq U_m)$ - *can use to construct piecewise constant functions*

5.1.1 Complexity Control

- Restriction Methods Where we limit class of functions before hand - e.g. insist on additive models of the form

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

  Size of model limited by number of basis functions M_j for each component function f_j.

- Selection Methods: which continually look at the dictionary and put in members (basis functions) that improve the fit or remove those which are not contributing. For example, variable selection methods such as forward stepwise.
- Regularization methods: where the whole dictionary is included,but the coefficients are restricted - e.g. ridge regression or the lasso

5.2 Piecewise Polynomials and Splines

Definition 5.2.1. *(Piecewise Constant)*

$$f(X) = \sum_{m=1}^{M} \beta_m \mathbb{I}(L_m \leq X \leq U_m)$$

For some partition of the input space $\{L_m, U_m\}_{m=1}^{M}$
Least squares will yield $\hat{\beta}_m = \overline{X}_m$ *the mean of* X *in the* m*th region.*

Definition 5.2.2. *(Piecewise Linear)*
We require additional basis functions - for each slice of our partition we have the polynomial: $\beta_m + \beta_{m+M} X$

$$f(X) = \sum_{m=1}^{M} \beta_m \mathbb{I}(L_m \leq X \leq U_m) + \beta_{m+M} \mathbb{I}(L_m \leq X \leq U_m) X$$

Definition 5.2.3. *(Continuous Piecewise linear)*
We start with $2M$ free parameters, but the continuous constraint means we now have instead $M-1$ degrees of freedom
Can alternatively build the constraints into the basis functions

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = (X-\xi_1)_+, \quad h_4(X) = (X-\xi_2)_+, \quad \dots$$

Definition 5.2.4. *(Piecewise cubic polynomials)*
As above but with a cubic on each piece - giving us continuity and continuity of first and second derivatives - total function called a ***cubic spline*** *An order-M spline with knots $\xi_j, j = 1, \dots, K$ a piecewise polynomial of order M*
A truncated-power basis set would be

$$h_j(X) = X^{j-1} \quad j = 1, \dots, M$$
$$h_{M+j}(X) = (X-\xi_j)_+^{M-1} \quad j = 1, \dots, K$$

Definition 5.2.5. *A natural cubic spline with K knots represented by K basis functions*

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

where

$$d_k(X) = \frac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k}, \quad k = 1, \dots, K-2$$

5.4 Smoothing Splines

Definition 5.4.1. *(Smoothing Splines)*
A smoothing spline is a function f that minimizes

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

where $\lambda \geq 0$ is a smoothing parameter.
First term measures fidelity of fit to the data, second term measures roughness of f - penalising curvature.

- $\lambda = 0$ - *interpolating spline, $f \in \mathcal{F}$ can be any function*
- $\lambda \to \infty$ - *the simple least squares line fit, since no second derivative can be tolerated*

The solution is a natural spline, can write it as

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

where $N_j(x)$ a n-dimensional set of basis functions.
Criterion simplifies to

$$RSS(\theta, \lambda) = (y - N\theta)^T(y - N\theta) + \lambda \underbrace{\theta^T \Omega_n \theta}_{=\int f''(t)^2 dt}$$

Where matrix $\{N\}_{i,j} = N_j(x_i)$ and $(\Omega_n)_{i,j} = \int N_i^{''}(t) N_j^{''}(t) dt$

Fitted smoothing spline is then

$$\hat{f}(x) = \sum_{j=1}^{n} N_j(x)\hat{\theta}_j, \quad \hat{\theta} = (N^T N + \lambda \Omega_n)^{-1} N^T y$$

5.4.1 Degrees of freedeom and Smoother Matrices

If λ prechosen - gives linear smoother spline. This is because the estimated parameters $\hat{\theta}$ are a linear combination of the yi.
Then

$$\hat{f} = N(N^T N + \lambda\Omega_n)^{-1} N^T y = S_\lambda y$$

Where S_λ the **smoother matrix** - only dependent on x and λ

Definition 5.4.2. *(Effective degrees of freedom for splines)*

$$df_\lambda = \mathrm{tr}(S_\lambda)$$

Proposition 5.4.3. *(Smoother matrix)*
S_λ is symmetric and positive semidefinite
Can write it as

$$S_\lambda - (I + \lambda K)^{-1}$$

Where K independent of λ
Sine $\hat{f} = S_\lambda y$ solves

$$\min_f (y - f)^T (y - f) + \lambda f^T K f$$

*call K the **penalty matrix***
Eigen-decomposition of S_λ is

$$S_\lambda = \sum_{k=1}^{n} \rho_k(\lambda) u_k u_k^T$$

Where u_k the eigenvectors and $\rho_k(\lambda)$ its eigenvalues.
Can show that

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

d_k the corresponding eigenvalue of K

Proposition 5.4.4. *(Facts about smoothing spline)*

- *The eigenvectors are not affected by λ , the whole family of smoothing splines (for a particular x) indexed by λ have the same eigenvectors.*
- $S_\lambda y = \sum_{k=1}^{n} u_k \rho_k(\lambda) \langle u_k, y \rangle$
  So, the smoothing spline operation decomposes y into the basis u and then shrinks the components by $\rho_k(\lambda)$ (whereas selection does 0-1).

5.5 Automatic Selection of the Smoothing Parameters

Definition 5.5.1. *(LOOCV) - **leave one out cross-validation** to select λ.*
Aim to minimise

$$\mathbb{E}\left[\int \left\{\hat{f}_\lambda(x) - f(x)\right\}^2 dx\right] = MISE$$

Estimate MISE by

$$\mathrm{CV}(\hat{f}_\lambda) = \sum_{i=1}^{N} \left(y_i - \hat{f}_\lambda^{(i)}(x_i)\right)^2$$

where $\hat{f}_\lambda^{(i)}(x_i)$ is the fitted value at x_i with the ith observation deleted from the data set used to fit the smoother.
Can show that

$$\mathrm{CV}(\hat{f}_\lambda) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i - \hat{f}(x_i)}{1 - S_\lambda(ii)}\right)^2$$

where $S_\lambda(ii)$ is the ith diagonal element of S_λ
*This is a **generalized cross-validation** (GCV) score.*

6 Kernel Smoothing Methods

Definition 6.0.1. *Kernel function*
$K : \mathbb{R} \to \mathbb{R}$ *satisfying*

- $K(x) \geq 0, \forall x$
- $\int_{\mathbb{R}} K(x) = 1$
- *Usually:* $K(-x) = K(x)$ *and* K *smooth*

Definition 6.0.2. *(Kernel density estimator)*
KDE for $X_1, \ldots, X_n$ *with kernel function* K *and bandwidth* h *given by*

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

Choice of bandwidth is important.

- h *smaller* $\to$ *higher variance, lower bias*
- h *larger* $\to$ *higher bias, lower variance*

Proposition 6.0.3. *(Bias and variance - rectangular kernel)*
Given observation X *what is probability of landing in interval* $(x - \frac{h}{2}, x + \frac{h}{2})$

$$\sum_{i=1}^{n} K(\frac{X_i - x}{h}) \text{ - number of } X_i \text{ in interval}$$

$$\hat{f}_{n,h,K}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{X_i - x}{h}) \text{ avg num of points/h}$$

$$\underbrace{nh\hat{f}_{n,h}(x)}_{N^*} \sim Bin(n, p)$$

$$\mathbb{E}[N^*] = np, \quad var(N^*) = np(1-p)$$

$$\mathbb{E}\left[\hat{f}_{n,h}(x)\right] = \frac{F(x + \frac{h}{2}) - F(x - \frac{h}{2})}{h} \xrightarrow[h \to 0]{} f(x)$$

$$var[nh\hat{f}(x)] = \frac{1}{nh} \frac{F(x - \frac{h}{2}, x + \frac{h}{2})}{h} \left\{1 - F\left(x - \frac{h}{2}, x + \frac{h}{2}\right)\right\}$$

$$var(\hat{f}x) \approx \frac{1}{nh} f(x)$$

Proposition 6.0.4. *(Finding optimum h)*
Use taylor approximations with assumption that f *is twice differentiable*

$$F(x + \frac{h}{2}) = F(x) + \frac{h}{2} f(x) + \frac{1}{2} f'(x) \left(\frac{h}{2}\right)^2 + \frac{1}{6} f''(x) \left(\frac{h}{2}\right)^3 + \mathcal{O}(h^4)$$

$$F(x - \frac{h}{2}) = F(x) - \frac{h}{2} f(x) + \frac{1}{2} f'(x) \left(\frac{h}{2}\right)^2 - \frac{1}{6} f''(x) \left(\frac{h}{2}\right)^3 + \mathcal{O}(h^4)$$

$$bias(\hat{f}(x)) = \frac{F(x - h/2, x + h.2)}{h} - f(x) \approx \frac{1}{24} h^2 f''(x)$$

$$MSE(\hat{f}_{n,h,k}(x)) \approx C_1 \frac{1}{nh} f(x) + C_2 (f''(x))^2 h^4, \quad C_1 = 1, C_2 = \frac{1}{24}^2$$

$$h^* = C^* n^{-\frac{1}{5}} \text{ where } C^* = \left[\frac{C_1}{4C_2} \frac{f(x)}{f''(x)^2}\right]^{1/5} \text{ the MSE optimal } h$$

Proposition 6.0.5. *(Properties of optimal bandwidth)*

- $h_n \xrightarrow[n\to\infty]{} 0$
- $nh_n \propto n^{\frac{4}{5}} \xrightarrow[n\to\infty]{} \infty$

Proposition 6.0.6. *(Expectation with general kernel)*

$$\begin{aligned}\mathbb{E}[\hat{f}(x)] &= \frac{1}{nh}\sum_{i=1}^{n}\mathbb{E}[K(\frac{X_i - x}{h})] \\ &= hf(x) + \frac{1}{2}C_3h^3f^{''}(x) + \mathcal{O}(h^4) \\ bias\{\hat{f}_{n,h,K}(x)\} &\approx \frac{1}{2}C_3h^2f''(x)\end{aligned}$$

Proposition 6.0.7. *(Bounding variance w/ general kernel)*

$$\begin{aligned}var\{\hat{f}(x)\} &= \frac{1}{nh^2}var\left[K\left(\frac{X_i - x}{h}\right)\right] \\ &\leq \frac{1}{nh}f(x)C_1 + \frac{1}{n}C_4 + \mathcal{O}(\frac{h}{n}) \xrightarrow[n\to\infty]{} 0 \\ C_4 &= f'(x)\int vK^2(v)dv\end{aligned}$$

6.6 Kernel density estimation and Classification

Definition 6.6.1. *(Kernel Regression)*
Given variables X, Y wish to find $\mathbb{E}[Y \mid X]$

$$\begin{aligned}\mathbb{E}[Y \mid X] &= \frac{\int yf(x,y)dy}{f(x)} \\ \hat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n}K_h(x - X_i) \\ \hat{\mathbb{E}}(Y \mid X = x) &= \frac{\sum_{i=1}^{n}K_h(x - X_i)\int yK_h(y - Y_i)dy}{\sum_{i=1}^{n}K_h(x - X_i)} \\ &= \frac{\sum_{i=1}^{n}Y_iK_h(x - X_i)}{\sum_{i=1}^{n}K_h(x - X_i)}\end{aligned}$$

Called the Nadaraya-Watson estimator

6.6.1 (Local polynomial regression)

Definition 6.6.2. *(Local polynomial estimator)*

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

Centred on x_0 or local to x_0
We have weighted RSS

$$RSS(x_0) = \sum_{i=1}^{n}\{Y_i - m_{x_0}(X_i)\}^2K_h(X_i - X_0)$$

And the design matrix and weight matrix,

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

Rewriting the RSS and minimising we get

$$RSS(x_0) = (Y - X\beta(x_0))^T W_{x_0}(Y - X\beta(x_0)), \quad \hat{\beta}(x_0) = (X^T W_{x_0} X)^{-1} X^T W_{x_0} Y$$

Proposition 6.6.3. *(Bias of NW)*
Take $f(x)$ density of $\{X_i\}$ and $g(x) = \mathbb{E}[Y \mid X = x]$ the unknown regression function

$$bias_{NW} : h^2 \left\{ \frac{1}{2} g''(x) + \frac{g'(x)f'(x)}{f(x)} \right\} \int K^2(u)du + \mathcal{O}(h^2)$$

Bias of local linear is

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

Generally choose odd polynomial order

6.6.2 Orthogonal Series Regression

Given $\{\rho_v(x)\}$ an orthogonal series basis for some function space
So for all f in this space

$$f(x) = \sum_v f_v \rho_v(x), \quad f_v = \langle f, \rho_v \rangle = \int f(x)\rho_v(x)dx$$

Orthogonality gives

$$\int \rho_v(x)\rho_w(x)dx = \delta_{vw}(\text{ The Kronecker delta})$$

2D case

$$f(x,y) = \sum_v \sum_u f_{v,u}\rho_v(x)\rho_u(y), \quad f_{v,u} = \int\int f(x,y)\rho_v(x)\rho_u(y)dxdy$$

Set of basis functions is *complete* for the function space

$$\lim_{m\to\infty} \int \left\{ f(x) - \sum_{r=-m}^{m} f_r \rho_r(x) \right\}^2 dx = 0, \quad \forall f$$

Definition 6.6.4. *(Orthogonal series estimator)*

$$\begin{aligned} f_v &= \int f(x)\rho_v(x)dx = \mathbb{E}[\rho_v(X)] \\ &\approx \frac{1}{n}\sum_{i=1}^{n} \rho_v(X_i) \\ f_{v,u} &= \mathbb{E}[\rho_v(X)\rho_u(Y)] = \hat{f}_v \\ &\approx \frac{1}{n}\sum_{i=1}^{n} \rho_v(X_i)\rho_u(Y_i) = \hat{f}_{v,u} \end{aligned}$$

Proposition 6.6.5. *(Properties of OS estimator)*

$$\mathbb{E}[\hat{f}(x)] = \frac{1}{n}\sum_{i=1}^{n} f(x) = f(x)$$

Unbiased - but cant use this to estimate $f(x)$ as v often infinite.
Estimating only for $v = -m, \ldots, m$ and $\hat{f}(x) = \sum_{v=-m}^{m} \hat{f}_v \rho_v(x)$

$$\begin{aligned} bias(\hat{f}(x)) &= \mathbb{E}[\hat{f}(x) - f(x)] \\ &= -\sum_{|v|>m} f_v \rho_v(x) \end{aligned}$$

Not necessarily 0.

5 Basis Expansion

5.9 Wavelet Smoothing

5.9.1 Multiresolution Analysis (MRA)

Definition 5.9.1. *(MRA)*
MRA used to examine functions at different scales - indexed by j
Spaces $\{V_j\}_{j\in\mathbb{Z}}$ form a ladder

$$\ldots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots, \quad \overline{\bigcup_{j\in\mathbb{Z}} V_j} = L_2(\mathbb{R})$$

Where L_2 functions are those which, over a finite range, have a finite number of discontinuities.
Functions get progressively less detailed as $j \to -\infty$ so

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$$

the zero function

$$f(x) \in V_j \Rightarrow f(2x) \in V_{j+1}, \forall j \in \mathbb{Z}$$

and

$$f(x) \in V_0 \Rightarrow f(x-k) \in V_0, \forall k \in \mathbb{Z}$$

Integer translates of a function in V_0 are also in V_0

$$\exists \phi(x) \in V_0 \text{ s.t } \{\phi(x-k)\}_{k\in\mathbb{Z}} \text{ is an orthonormal basis for } V_0$$

If $\{V_j\}_{j\in\mathbb{Z}}$ and ϕ satisfy the above then they form a MRA

Definition 5.9.2. *(Haar father wavelet)*

$$\phi_H(x) = \begin{cases} 1, & \text{if } x \in (0,1); \\ 0, & \text{otherwise .} \end{cases}$$

Given inner product $\langle f, g\rangle = \int f(x)g(x)dx$

$$\|\phi_H\|^2 = \langle \phi_H, \phi_H\rangle = \int \phi_H^2(x)dx = \int_0^1 1dx = 1$$

And

$$\langle \phi_H(x-\ell), \phi_H(x-m)\rangle = \delta_{l,m}$$

Different integer translates do not overlap.
$\{\phi_H(x-\ell)\}_{\ell\in\mathbb{Z}}$ form an orthonormal set and we can set $V_0 = span\{\phi_H(x-\ell)\}_{\ell\in\mathbb{Z}}$

Proposition 5.9.3. *Dyadically scale and translate the wavelet by*

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

$\{\phi_{j,k}(x)\}_{k\in\mathbb{Z}}$ *is an orthonormal basis for* V_j
Show that for function $f(x)$

$$P_j f = \sum_{k\in\mathbb{Z}} c_{j,k}\phi_{j,k}(x)$$

For $j = 5, 3, 1$ *; the projection of* $f(x)$ *onto* V_j*, where*

$$c_{j,k} = \int f(x)\phi_{j,k}(x)dx$$

Definition 5.9.4. *(Haar mother wavelet)*

$$\psi_H(x) = \begin{cases} 1, & \text{if } x \in (0, \frac{1}{2}); \\ -1, & \text{if } x \in (\frac{1}{2}, 1); \\ 0, & \text{otherwise} \end{cases}$$

Call space spanned by $\psi(x-k)$ W_0 *and note that* $\psi(x)$ *orthonormal to* $\phi(x)$ *with their respective spaces being orthonormal too.*
Extend the idea across all scales and locations

$$f(x) = \sum_{k\in\mathbb{Z}} c_{j_0,k}\phi_{j_0,k}(x) + \sum_{j=0}^{\infty}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x)$$

For function $f(x)$ *at scale* $j_0 \ldots$
Infinite scales yield

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

Call the set $\{d_{j,k}\}$ *the* ***wavelet coefficients***

Proposition 5.9.5. *(Wavelet coefficients)*
Given that we have $c_{1,0}, c_{1,1}$ *can obtain*

$$c_{0,0} = \frac{1}{\sqrt{2}}(c_{1,1} + c_{1,0})$$
$$d_{0,0} = \frac{1}{\sqrt{2}}(c_{1,1} - c_{1,0})$$

Generalising to

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

Definition 5.9.6. *(Wavelet transform)*
A wavelet has m *vanishing moments if*

$$\int x^j \psi(x)dx = 0, \quad \forall j = 0, 1, \ldots, m-1$$

5.9.2 Wavelet Shrinkage

Suppose we have model

$$y_i = f_i + \epsilon_i, \quad i = 1, \ldots, n$$

Where we observe $\{y_i\}_{i=1}^n$. Assume ϵ_i are IID random variables with mean 0 and var σ^2 - ϵ vector then has covariance matrix $\sigma^2 I_n$
$\{f_i\}_{i=1}^n$ are unknown functions and we want to estimate them.
Wavelet transform $e = W\epsilon$

$$\mathbb{E}[e] = W\mathbb{E}[\epsilon] = 0 \quad var(e) = \mathbb{E}[ee^T] = \sigma^2 I_n$$

Applying W to our model we get

$$w = d + e$$

The noise e uncorrelated - gets spread evenly across all coefficients.
The signal d is sparse - we have that $\|d\| = \|f\|$. So f is sparse in the wavelet domain - improving the signal to noise ratio greatly.

5.9.3 Thresholding Types

Definition 5.9.7. *(Hard Thresholding)*

$$T_{hard}(w, \lambda) = w\mathbb{I}(\|w\| > \lambda)$$

Definition 5.9.8. *(Soft Thresholding)*

$$T_{soft}(w, \lambda) = sign(w)(\|w\| - \lambda)\mathbb{I}(\|w\| > \lambda)$$

Definition 5.9.9. *(Bayesian Wavelet Shrinkage)*
Natural to think about wavelet coefficients having the prior distribution

$$d_{j,\cdot} = \gamma_j N(0, \tau_j^2) + (1 - \gamma_j)\delta_0(x)$$

$\delta_0(x)$ is the Dirac delta function at 0, γ_j a Bernoulli random variable with parameter p_j
Likelihood comes from $w = d + e$, if $e \sim N(0, \sigma^2 I_n)$ then $w \mid d \sim N(d, \sigma^2)$
Can show

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

where Φ the CDF of the standard normal, $\nu^2 = \tau^2(\sigma^2 + \tau^2)^{-1}$ and $r \in (0, 1)$

14 Cluster Analysis

14.7 Independent Component Analysis and Exploratory Projection Pursuit

14.7.1 Latent Variables and Factor Analysis

Saw the SVD of a $n \times p$ matrix X as

$$X = \underbrace{U}_{n\times p}\ \underbrace{D}_{p\times p, d_{ii}\geq 0}\ \underbrace{V^T}_{p\times p}$$

With the $p \times p$ sample covariance matrix $S = n^{-1}X^T X$ as

$$S = n^{-1}X^T X = n^{-1}VD^2V^T$$

an eigen-decomposition, with eigenvectors v_j (principal components) with eigenvalues d_j^2
Can project the X onto principal components forming

$$z_{n\times 1} = X_{n\times p}v_{p\times 1}$$

With $Var(z_i) = \frac{d_i^2}{n}$ once projected onto v_i

Proposition 14.7.1. *(Projection onto arbitrary vector)*
Suppose we project onto arbitrary p vector a

$$y_{n\times 1} = X_{n\times p}a$$

Then

$$n^{-1}y^T\mathbf{1} = n^{-1}a^TX^T\mathbf{1} = 0, \quad Var(y_i) = S_y(a) = n^{-1}y^Ty = a^TSa$$

Proposition 14.7.2. *(Optimisation formulation of PCA)*

$$\textit{First PC: } \max_a S_y(a) \textit{ given } a^Ta = 1$$

$$i\textit{th PC}: \max_a S_y(a) \textit{ given } a^Ta = 1, a^Tv_j = 0, \forall j = 1, \ldots, i-1$$

Definition 14.7.3. *(Kullback-Liebler Divergence of g from f)*

$$D_{KL}(f \mid g) = \int g(x) \log\left\{\frac{g(x)}{f(x)}\right\} dx$$

Easy to see that $D_{KL}(g \mid f) \geq 0$ with equality if and only if $f = g$

Definition 14.7.4. *Entropy of density g*

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

Proposition 14.7.5. *(Gaussian maximises entropy)*
$f(x)$ density of $N(0, \sigma^2)$ and g arbitrary any other density with mean 0 and variance σ^2

$$H(g) \leq H(f)$$

Definition 14.7.6. *(Sphering)*
Transform matrix so its variance is the identity matrix.

$$S = n^{-1}X^TX \quad R = S^{-1/2} \quad \underbrace{W}_{sphered} = XR$$

This makes PC redundant as

$$var(Wa) = a^TS_Wa = a^Ta = 1$$

Algorithm 14.1. *Process and Optimisation*

1. *Start with centred and sphered matrix W*
2. *Choose initial unit projection vector a*
3. *Form projected data $u_a = Wa$*
4. *Form density estimate, $\hat{f}_{U,a}(u)$ from $u_1, \ldots, u_n$*
5. *Compute entropy $H\{\hat{f}_{U,a}(u)\}$*
6. *Solve $\arg\min_{a:a^Ta=1} H\{\hat{f}_{U,a}(u)\}$*
7. *Build up multidimensional solutions by optimising over unit b orthogonal to a*

Definition 14.7.7. *(Factor Analysis Model)*
$X = UDV^T$ has latent variable representation
Writing $S = \sqrt{n}U$ and $A^T = DV^T/\sqrt{N}$ we have $X = SA^T$

$$\begin{aligned} X_{i,1} &= a_{1,1}S_{i,1} + \cdots + a_{1,p}S_{i,p} \\ X_{i,2} &= a_{2,1}S_{i,1} + \cdots + a_{2,p}S_{i,p} \\ &\vdots \\ X_{i,p} &= a_{p,1}S_{i,1} + \cdots + a_{p,p}S_{i,p} \\ X &= AS \end{aligned}$$

X will be correlated, but want S to be uncorrelated.
Assume X centered (so is S)

Proposition 14.7.8. *(Covariance of S)*

$$n^{-1}S^TS = n^{-1}U^TU = I$$

So S uncorrelated
But for R orthogonal, write

$$X = SA^T = SRR^TA^T = S^*(A^*)^T$$

with $\mathrm{Cov}\,[S^*] = I_p$
So there is no unique decomposition into uncorrelated factors $S_1, \ldots, S_p$

Definition 14.7.9. *(Classical Factor Analysis)*

$$\begin{aligned} X_{i,1} &= a_{1,1}S_{i,1} + \cdots + a_{1,q}S_{i,q} + \epsilon_{i,1} \\ X_{i,2} &= a_{2,1}S_{i,1} + \cdots + a_{2,q}S_{i,q} + \epsilon_{i,2} \\ &\vdots \\ X_{i,p} &= a_{p,1}S_{i,1} + \cdots + a_{p,q}S_{i,q} + \epsilon_{i,p} \end{aligned}$$

Here there are $q < p$ factors, ϵ are zero mean and uncorrelated and S assumed to be Gaussian

14.7.2 Independent Component Analysis

Definition 14.7.10. *(Independent Component Analysis)*
$X = AS$ with A unknown and S independent
X is observed, A is unknown, S is unknown
Aim to find orthogonal A and S independent
If Y has pdf g we say that the entropy of Y is

$$H(Y) = H(g) = -\int g(y)\log g(y)dy$$

Definition 14.7.11. *(Mutual Information)*

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

Call $I(Y)$ the Kullback-Lieber distance between the density $g(y)$ of Y

Proposition 14.7.12. *(Mutual Information)*
If X has covariance matrix I and $Y = A^TX$ then

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

Minimising this over A.

11 Neural Networks

11.2 Projection Pursuit Regression

Definition 11.2.1. *(Projection Pursuit Regression)*
Here the model is

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

this is an additive model on the derived directions $V_m = \omega_m^T X$ and not the X directly, ω_m are unit vectors
Call $g_m(\omega_m^T X)$ the ridge function in $\mathbb{R}^p$ and only varies in direction ω_m
V_m is the projection of X onto ω_m and want to find ω_m so the model fits will; like projection pursuit.
If M is taken arbitrarily large, for appropriate choice of g_m the PPR model can approximate any continuous function in $\mathbb{R}^p$ - called universal approximation property

Proposition 11.2.2. *(Fitting PPR models)*
Suppose we have training data $(x_i, y_i), i = 1, \ldots, ; x_i \in \mathbb{R}^p$
Want to find approximate minimisers of error

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

over $\{g_m\}$ and direction vectors $\{\omega_m\}$

- *Considering the one term case $M = 1$- given direction vector ω can form the derived variable $v_i = w^T x_i$. We have a 1D smoothing problem, can apply smoothing spline to obtain an estimate of g*
- *Or given g can minimise the error ω using a Gauss-Newton Search.*

Proposition 11.2.3. *(Gauss-Newton Search)*
Let ω_{old} be the current estimate for ω, then

$$g(\omega^T x_i) \approx g(\omega_{old}^T x_i) + g'(\omega_{old}^T x_i)(\omega - \omega_{old})^T x_i$$

So that

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

Minimising the RHS requires we carry out a least squares regression with target $w_{old}^T x_i + \frac{y_i - g(\omega_{old}^T x_i)}{g'(w_{old}^T x_i)}$ and predictor x_i with weights $g'(w_{old}^T x_i)^2$ and no intercept - gives ω_{new}

Algorithm 11.1. *PPR algorithm iterates as follows*

- *finding good g using current ω*
- *using g to update current ω to ω*
- *Iterate until convergence - only need to examine changes in ω*

Can decide optimal M by cross validation

11.3 Neural Networks

Definition 11.3.1. *(Neural Network Model)*
The model has $X_1, \ldots, X_p$ explanatory variables at the bottom, with K units at the top - for regression $K = 1$, for classification we use K units for each class.
The model is

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

where $Z = (Z_1, \ldots, Z_M)$ and $T = (T_1, \ldots, T_K)$
Call $\sigma(v)$ the activation function,

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

Output function $g_k(T)$ permits final transformation of vector of outputs - for regression $g_k(T) = T_k$ and for classification $g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^K e^{T_l}}$ the softmax function.
Z_m are the hidden units not directly observed - often many hidden layers.
If σ the identity then back to linear model.

Proposition 11.3.2. *(ANN vs. PPR)*
PPR model uses non-parametric functions $g_m(v)$ whereas ANN uses simpler function based on $\sigma(v)$
Can write

$$g_m(\omega_m^T X) = \beta_m \sigma(\alpha_{0,m} + \|\alpha_m\| (\omega^T X))$$

where $\omega_m = \alpha_m / \|\alpha_m\|$ the mth unit vector.
Since σ less complex than g_m need more layers to get same approximative power.

11.4 Fitting Neural Networks

Proposition 11.4.1. *(Fitting Neural Networks)*
Given training data $(x_i, y_i), i = 1, \ldots, n$ want to find

$$\begin{aligned} \{\alpha_{0,m}, \alpha_m : m = 1, \ldots, M\}, &\quad M(p+1) \text{ weights} \\ \{\beta_{0,k}, \beta_k : k = 1, \ldots, K\}, &\quad K(M+1) \text{ weights} \end{aligned}$$

to minimise the error, the usual sum of squares

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

Typically need constraints to avoid overfitting.

- *Backpropagation - use gradient descent to minimise $R(\theta)$*

Proposition 11.4.2. *(Backpropagation)*
Need to compute the gradient of $R(\theta)$ with respect to each weight.

$$\frac{\partial R_i(\theta)}{\partial \beta_{k,m}} = \underbrace{-2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_j + \beta_{0,k})}_{\delta_{k,i}} z_{m,i} = \delta_{k,i} z_{m,i}$$

$$\frac{\partial R_i(\theta)}{\partial \alpha_{m,l}} = \underbrace{-2 \sum_{k=1}^K (y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i + \beta_{0,k}) \beta_{k,m} \sigma'(\alpha_{0,m} + \alpha_m^T x_i)}_{s_{m,i}} x_{il} = s_{m,i} x_{il}$$

Use these to form the gradient descent algorithm by

$$\beta_{k,m}^{(r+1)} = \beta_{k,m}^{(r)} - \gamma_r \frac{\partial R_i(\theta)}{\partial \beta_{k,m}^{(r)}}$$
$$\alpha_{m,l}^{(r+1)} = \alpha_{m,l}^{(r)} - \gamma_r \frac{\partial R_i(\theta)}{\partial \alpha_{m,l}^{(r)}}$$

Where γ_r the learning rate.
In total we get

$$s_{m,i} = \sigma'(\alpha_{0,m} + \alpha_m^T x_i) \sum_{k=1}^{K} \delta_{k,i} \beta_{k,m}$$

We have the algorithm in 2 stages

1. *Forward pass - fixed current weights - and predicted values $\hat{f}_k(x_i)$ computed*
2. *Backward pass - errors $\delta_{k,i}$ and $s_{m,i}$ computed to form gradients for the updates.*

Proposition 11.4.3. *(Issues with fitting ANNs)*

- *Starting values - usually chosen to be random, but small, near 0.*
- *Overfitting - need to use cross-validation to choose number of hidden units and layers.*
- *Scaling of inputs - sensitive to input scales, usually centre and standardise inputs*

9 Additive Models, Trees and Related Methods

9.2 Tree-Based Methods

9.2.2 Regression Trees

Proposition 9.2.1. *(Growing a tree)*
Given data of p inputs and a response for each N observations; $(x_i, y_i), i = 1, \ldots, N$ with $x_i = (x_{i1}, \ldots, x_{ip})$
Need to decide splitting variables, and split points, and topology of tree.
Suppose we have M regions in our partition, $R_1, \ldots, R_M$ model the response as a constant c_m in each region

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

Easy to see that the best $\hat{c}_m$ minimising the sum of squares is the mean of the response in each region

$$\hat{c}_m = ave(y_i \mid x_i \in R_m) = \frac{1}{n_m} \sum_{i:x_i \in Rm} y_i$$

Apply a greedy algorithm to find the best partitioning of the data.

1. *Start with all data in one region*
2. *Find best split of data into 2 regions*
3. *Repeat for each region until stopping criterion met*

Consider splitting variable $j = 1, \ldots, p$ and split point s, define the pair of half-planes

$$R_1(j,s) = \{x \mid X_j < S\} \text{ and } R_2(j,s) = \{x \mid X_j \geq S\}$$

Seek splitting variable j and split point s that solves

$$\min_{j,s} \left\{ \underbrace{\min_{c_1} \sum_{X_i \in R_1(j,s)} (y_i - c_1)^2}_{=\hat{c}_1} + \underbrace{\min_{c_2} \sum_{X_i \in R_2(j,s)} (y_i - c_2)^2}_{=\hat{c}_2} \right\} = \min_{j,s} \left(\sum_{X_i \in R_1(j,s)} (y_i - \hat{c}_1)^2 + \sum_{X_i \in R_2(j,s)} (y_i - \hat{c}_2)^2 \right)$$

Can now simply compute all the different values choosing the minimum s, repeating over all j. We then repeat after partitioning the data into the 2 regions.
Stopping criterion

- *Maximum tree depth*
- *Minimum number of observations in a region*
- *Minimum reduction in RSS from splitting*

Definition 9.2.2. *Cost-complexity pruning*
Define a subtree T_0 of T by collapsing any number of its internal nodes.
Let $|T|$ the number of terminal nodes of tree T
Let n_m number of observations in R_m
$\hat{c}_m = n_m^{-1} \sum_{x_i \in R_m} y_i$ and $Q_m(T) = n_m^{-1} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$
The cost-complexity criterion is

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

Idea is to find for each α the subtree $T_\alpha \subseteq T_0$ that minimises $C_\alpha(T)$ and then choose the subtree with the smallest α that is within one standard error of the minimum.

Definition 9.2.3. *(Weakest link pruning)*
Start with the full tree T_0 and find the weakest link, the pair of nodes that when removed gives the smallest increase in $\sum_{m=1}^{|T|} n_m Q_m(T)$. Until we end up with a single node tree.
Can show that this sequence has tree T_α that minimises $C_\alpha(T)$
Estimate of α is given by 5 or 10 fold cross-validation.

Definition 9.2.4. *(Classification trees)*
Similar to regression trees, but instead of minimising SSQ, we use

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

the proportion of class k observations in node m.
Classify the observations in node m to class $k(m) = \arg\max_k \hat{p}$ the majority class in node m
Can use the following alternatives for $Q_m(T)$

- *Misclassification error; $\frac{1}{n_m} \sum_{x_i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{m,k(m)}$*
- *Gini index; $\sum_{k=1}^K \hat{p}_{m,k}(1 - \hat{p}_{m,k}) = \sum_{k=1}^K \hat{p}_{m,k}(1 - \hat{p}_{m,k})$*
- *Cross-entropy; $-\sum_{k=1}^K \hat{p}_{m,k} \log \hat{p}_{m,k}$*

8 Model Inference and Averaging

8.2 The Bootstrap and Maximum Likelihood Methods

8.2.1 Bootstrap

Definition 8.2.1. *Bootstrap*
Let $X_1, \ldots, X_n$ be a random sample from a distribution with unknown distribution function F.
Let $\hat{\theta} = \hat{\theta}(X_1, \ldots, X_n)$ be an estimator of statistic $\theta = \theta(F)$ a functional of F
Can estimate $\theta(F)$ replacing F by empirical distribution function:

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

Can simulate from $\hat{F}_n(x)$ as it puts mass $1/n$ at each X_i
Let $X_1^{(1),\ldots,X_n^{(1)}}$ be a random sample from $X_1, \ldots, X_n$ with replacement
Call $\{X_i^{(b)}\}_{i=1}^n$ the bth bootstrap sample, and let their empirical distribution function be $\hat{F}^{(b)}, b = 1, \ldots, B$

8.7 Bagging

Definition 8.7.1. *(Bagging)*
Suppose we fit model to training data $Z = \{(x_1, y_1), \ldots, (x_n)\}$, and obtain $\hat{f}(x)$
Can produce B bootstrap samples $Z^{(b)}, b = 1, \ldots, B$ and give each an estimate $\hat{f}^{(b)}(x)$
Bootstrap aggregation, or Bagging, estimate averages these predictions over many bootstrap samples by

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

Definition 8.7.2. *(Actual Bagging)*
Let $\hat{P}$ be distribution that puts mass of n^{-1} onto each of the (x_i, y_i)
The 'true' bagging estimate defined by $\mathbb{E}_{\hat{P}}\{\hat{f}^*(x)\}$ where $\hat{f}^*$ is obtained from a set $Z^* = \{(x_i^*, y_i^*)\}_{i=1}^n$ where each $(x_i^*, y_i^*) \sim \hat{P}$
Then $\hat{f}_{bag}(x)$ an estimate of the true bagging estimate approaching as $B \to \infty$
Bagged estimate will only differ when estimator is a non-linear or adaptive estimator
If linear then $\hat{f}_{bag}(x) \to \hat{f}(x)$ as $B \to \infty$

Proposition 8.7.3. *(Why bagging works)*
Assume training observations are drawn independently from distribution P
Ideal bagging estimator $f_{ag}(x) = \mathbb{E}_P \hat{f}^*(x)$
Consider

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

Hence true population aggregate never increases MSE, suggests bagging will often decrease MSE.

Definition 8.7.4. *(Bagging Trees)*
Bagging trees is a special case of bagging, where we use regression trees as the base learner.
Suppose $W_1, \ldots, W_B$ a set of independent random variables with variance σ^2

$$\overline{W} = B^{-1} \sum_{b=1}^{B} W_b, \quad var(\overline{W}) = \sigma^2 / B \xrightarrow[B \to \infty]{} 0$$

Suppose that $\mathrm{Cov}\,[W_b, W_d] = \sigma^2 \rho > 0$ *for $b \neq d$ then*

$$var(\overline{W}) = \rho \sigma^2 + \frac{1 - \rho}{B} \sigma^2 \xrightarrow[B \to \infty]{\not} 0$$

Correlated trees limit the effectiveness of bagging.

Algorithm 8.1. *(Random Forests)*
Random forests are a modification of bagging that builds a large collection of de-correlated trees, and then averages them.

1. *For* $b = 1, \ldots, B$
   (a) *Draw a bootstrap sample* $Z^{(b)}$ *of size* n *from the training data*
   (b) *Grow a random-forest tree* T_b *to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size* n_{min} *is reached.*
   (c) *Select* $m \leq p$ *variables at random from the* p *variables*
   (d) *Pick the best variable/split-point among the* m
   (e) *Split the node into two daughter nodes*
2. *Output the ensemble of trees* $\{T_b\}_{b=1}^{B}$

For a prediction at new point x

- *Regression:* $\hat{f}_{rf}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
- *Classification: Let* $\hat{C}_b(x)$ *be the class prediction of the bth random-forest tree.*
  Then $\hat{C}_{rf}(x) =$ majority vote$\{\hat{C}_b(x)\}_{b=1}^{B}$

10 Boosting and Additive Trees

10.1 Boosting Methods

Definition 10.1.1. *Boosting*
We build sequence of classifiers $G_m(x), m = 1, \ldots, M$ *where* $G_1(x) = G(x)$ *and* $G_m(x)$ *depends on previous classifiers.*
Combining them to give

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

Where the $\{\alpha_m\}_{m=1}^{M}$ *are computed by the boosting algorithm.*

Proposition 10.1.2. *(Boosting weights)*
Apply weights $w_1, \ldots, w_n$ *to each of the training observations* $(x_i, y_i), i = 1, \ldots, n$
Initialise all weights to equal $w_i = 1/n$
Each successive iteration, weights modified and classifier applied to weighted observations
Incorrectly classified observations get their weight increased

Algorithm 10.1. 1. *Initialise weights* $w_i = 1/n$ *for* $i = 1, \ldots, n$

2. *For* $m = 1, \ldots, M$
   (a) *Fit classifier* $G_m(x)$ *to training data using weights* $\{w_i\}$
   (b) *Compute*

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

   (c) *Compute* $\alpha_m = \log\{(1 - e\mathrm{rm})/e\mathrm{rm}\}$
   (d) *Set* $w_i \leftarrow w_i \cdot \exp\{\alpha_m I(y_i \neq G_m(x_i))\}$ *for* $i = 1, \ldots, n$
3. *Output*

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

11 Ethics

lol