Quantitative Research Notes

A short complation of knowledge

Pratim Chowdhary

Last updated: February 21, 2025

Contents

※ Introduction

This is a short compilation of notes on various topics in quantitative research. The notes are based on various sources and are intended to be a quick reference guide for students as well as for myself to brush up on concepts.

※ Decision Theory

2.1 Framework

1. A **statistical model** is a framily of distributions, formally defined as:

$$\mathcal{P} = \{P_\theta : \theta \in \Theta\} \tag{1}$$

   where Θ is the parameter space and P_θ is the distribution of the data given θ. The parameter space Θ is the set of all possible values of θ.

2. A **decision procedure** is a function $\delta : \mathcal{X} \to \Theta$ that maps the data space $\mathcal{X}$ to the parameter space Θ. For example if we take the example of a weighted coin flips, we have:

$$\mathcal{X} = \{0,1\}^n \quad \text{and} \quad \Theta = [0,1] \tag{2}$$

If we are interested in estimating the parameter θ of the coin, we can define a decision space as $\Theta = [0, 1]$ and a decision procedure as,

$$\delta(x) = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3}$$

where $x \in \mathcal{X}$ is the data and $\delta(x)$ is the estimate of the parameter θ.

3. A **loss function** is a function $L : \Theta \times \Theta \to \mathbb{R}$ that measures the loss incurred by choosing θ when the true parameter is θ'. For example in many situtations we use the squared loss function:

$$L(\theta, \theta') = (\theta - \theta')^2 \tag{4}$$

4. The **risk** of a decision procedure δ is the expected loss incurred by the decision procedure:

$$R(\theta, \delta) = \mathbb{E}_\theta[L(\theta, \delta(X))] \tag{5}$$

where $\mathbb{E}_\theta$ denotes the expectation with respect to the distribution P_θ.

2.2 Data Reduction

The idea is that not all data is relevant for making decisions. We can hence reduce the data to a smaller set of and not lose any information.

1. A **statistic** is a function $T : \mathcal{X} \to \mathcal{T}$ that maps the data space $\mathcal{X}$ to a smaller space $\mathcal{T}$.
2. A statistic T is **sufficient** for a parameter θ if the distribution of the data given the statistic T does not depend on θ. Formally if for all t,

$$P_\theta(X|T = t) = P_{\theta'}(X|T = t) \quad \forall \theta, \theta' \tag{6}$$

3. For any matrix $X \in \mathbb{R}^{n \times p}$, $X^T X$ must be at least positive semi-definite, this is because:

$$v^T X^T X v = (Xv)^T Xv = ||Xv||^2 \geq 0 \tag{7}$$

where $v \in \mathbb{R}^p$.

※ Linear Algebra

3.1 Eigenvalues and Eigenvectors

1. The eigenvalues of a matrix $A \in \mathbb{R}^{n \times n}$ are the roots of the characteristic polynomial:

$$\det(A - \lambda I) = 0 \tag{8}$$

where λ is the eigenvalue and I is the identity matrix. The eigenvectors are the vectors v such that:

$$Av = \lambda v \tag{9}$$

2. The trace of a matrix is the sum of its eigenvalues and the determinant is the product of its eigenvalues.
3. The eigenvectors of a matrix are orthogonal if the matrix is symmetric.
4. The eigenvectors of a matrix are linearly independent if the matrix is diagonalizable.

3.2 (Semi) Positive Definite Matrices

1. A matrix $A \in \mathbb{R}^{n \times n}$ is **positive semi-definite** if for all $x \in \mathbb{R}^n$:

$$x^T A x \geq 0 \tag{10}$$

   Positive definite matrices are defined similarly with the inequality replaced by a strict inequality.

2. If a matrix is positive semi-definite, then all its eigenvalues are non-negative and if it is positive definite, then all its eigenvalues are positive.

3.3 Covariance Matrix

1. The covariance matrix of a random vector $X \in \mathbb{R}^n$ is defined as:

$$\Sigma = \mathbb{E}[(X - \mu)(X - \mu)^T] \tag{11}$$

   where $\mu = \mathbb{E}[X]$ is the mean of the random vector X. The covariance matrix is a symmetric positive semi-definite matrix. With a real set of data $X \in \mathbb{R}^{n \times p}$, the empircal covariance matrix is defined as:

$$\hat{\Sigma} = \frac{1}{n-1} X^T X \in R^{p \times p} \tag{12}$$

   where X is the data matrix with n samples and p features.

2. Note the covariance matrix can only be full rank if $N \geq p$ and none of the features are linearly dependent, this is because:

$$\text{rank}(\Sigma) \leq \min(n, p) \tag{13}$$

   and since $n \geq p$, the rank of the covariance matrix is at most p. Some of the useful properties of the covariance matrix are:

   - The covariance matrix is symmetric and positive semi-definite.
   - The covariance matrix is diagonal $\iff$ the features are uncorrelated.
   - It is related to the correlation matrix by:

$$\text{Corr}(X) = D^{-1} \Sigma D^{-1} \quad \text{where} \quad D = \text{diag}(\Sigma)^{1/2} \tag{14}$$

   - The covariance matrix is positive definite $\iff$ the features are linearly independent.

3.4 Idempotent Matrices

1. A matrix $A \in \mathbb{R}^{n \times n}$ is idempotent if $A^2 = A$. The following are some properties of idempotent matrices:

   - The eigenvalues of an idempotent matrix are either 0 or 1, this can be seen by:

$$Av = \lambda v \implies A^2 v = \lambda^2 v \implies \lambda^2 v = \lambda v \implies \lambda = 0, 1 \tag{15}$$

- The rank of an idempotent matrix is equal to its trace this is because: since the trace is the sum of the eigenvalues, the rank is the number of non-zero eigenvalues.
- The matrix $I - A$ is also idempotent, we can see this by:

$$(I - A)^2 = I^2 - 2A + A^2 = I - A \tag{16}$$

2. Any **projection matrix** is idempotent, this is because the projection matrix projects a vector onto a subspace and then projects it again onto the same subspace.

3.5 QR Decomposition

1. The **QR decomposition** of a matrix $A \in \mathbb{R}^{n \times p}$ is a decomposition of the form:

$$A = QR \quad \text{where} \quad Q^T Q = I \quad \text{and} \quad R = \left[r_{ij} \right] \tag{17}$$

where Q is an orthogonal matrix and R is an upper triangular matrix.

$$Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_p \end{bmatrix} \quad \text{and} \quad R = \left[r_{ij} \right] \tag{18}$$

each q_i in Q is an orthogonal vector with $||q_i|| = 1$ and $r_{ij} = 0$ for $i > j$.

2. The **Gram-Schmidt** process is a method for computing the QR decomposition of a matrix. The process is as follows:

   (a) Let $a_1, a_2, \cdots, a_p$ be the columns of the matrix A.

   (b) Set $q_1 = a_1 / ||a_1||$.

   (c) For $i = 2, 3, \cdots, p$, set:

$$q_i = a_i - \sum_{j=1}^{i-1} \text{proj}_{q_j}(a_i) = a_i - \sum_{j=1}^{i-1} \frac{a_i^T q_j}{q_j^T q_j} q_j \rightarrow q_i = \frac{q_i}{||q_i||} \tag{19}$$

   the intuition for this is that we are projecting the vector a_i onto the subspace spanned by $q_1, q_2, \cdots, q_{i-1}$ and then subtracting that projection from a_i.

   Here the R matrix is given by:

$$R_{ij} = q_i^T a_j \tag{20}$$

   as we can get back the original matrix A by:

$$A = [a_1, a_2, \cdots, a_p] = [q_1, q_2, \cdots, q_p] \begin{bmatrix} q_1^T a_1 & q_1^T a_2 & \cdots & q_1^T a_p \\ 0 & q_2^T a_2 & \cdots & q_2^T a_p \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_p^T a_p \end{bmatrix} \tag{21}$$

   as $a_i = \sum_{j=1}^{p} q_j R_{ij}$. Intuitively this is true because each a_i is a linear combination of the q_i vectors, specifically the linear component is how much of a_i is in the direction of q_i (the projection).

3. **Linear Least Squares**: QR decomposition can help aid in solving the linear least squares problem especially in cases of high collinearity, as:

$$\text{cond}(X^TX) = \text{cond}(X)^2 \implies (X^TX)^{-1} \quad \text{is ill-conditioned} \tag{22}$$

Reframing linear least squares with the QR decomposition, we have:

$$X^TX\hat{\beta} = X^Ty \implies R^TQ^TQR\hat{\beta} = R^TQ^Ty \implies R\hat{\beta} = Q^Ty \tag{23}$$

Here since R is upper triangular, we can solve for $\hat{\beta}$ by back substitution, creating a more numerically stable solution with easier computation.

3.6 Spectral Theorem

1. The **spectral theorem** states that any symmetric matrix $A \in \mathbb{R}^{n\times n}$ can be decomposed into the form:

$$A = Q\Lambda Q^T \tag{24}$$

Where Q is an orthogonal matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues.

2. We first show that if a matrix is symmetric all of its eigenvalues are real. We can show this by considering the eigenvalue equation:

$$Ax = \lambda x \implies x^TAx = x^T\lambda x \implies \lambda = \frac{x^TAx}{x^Tx} \tag{25}$$

and since A is symmetric we have that $A = A^T$ and hence:

$$\lambda = \frac{x^TAx}{x^Tx} = \frac{x^TA^Tx}{x^Tx} = \frac{x^TAx}{x^Tx} = \lambda \tag{26}$$

and hence λ is real.

3. We must also show that the eigenvectors are orthogonal. We show this by considering any two eigenvectors x and y of A:

$$Ax = \lambda x \quad \text{and} \quad Ay = \mu y \tag{27}$$

then we have:

$$x^TAy = \lambda x^Ty \quad \text{and} \quad y^TAx = \mu y^Tx \tag{28}$$

and since A is symmetric we have that,

$$x^TAy = y^TAx \implies \lambda x^Ty = \mu y^Tx \implies (\lambda - \mu)x^Ty = 0 \tag{29}$$

and since $x^Ty \neq 0$ we have that $\lambda = \mu$.

4. We can show that $A = Q\Lambda Q^T$ by considering the spectral decomposition of A:

$$Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix} \quad \text{and} \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n) \tag{30}$$

We then have that:

$$Aq_i = \lambda_i q_i \implies AQ = Q\Lambda \tag{31}$$

therefore we have that,

$$Q\Lambda Q^T = (A)QQ^T = A \tag{32}$$

3.7 Eigen Decomposition

1. The **eigen decomposition** of a matrix $A \in \mathbb{R}^{n \times n}$ is a decomposition of the form:

$$A = Q \Lambda Q^{-1} \tag{33}$$

   where Q is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

2. Any symmetric matrix is diagonalizable, this is because the eigenvectors of a symmetric matrix are orthogonal.

3.8 Pseudo Inverse

※ Optimization

※ Probability

5.1 Multivariate Normal Distribution

1. A random vector $X \in \mathbb{R}^n$ is said to follow a **multivariate normal distribution** if it has a joint probability density function of the form:

$$f(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{34}$$

   where $\mu \in \mathbb{R}^n$ is the mean vector and $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix. The covariance matrix Σ must be symmetric and positive semi-definite.

※ Statistics

6.1 Hypthothesis Testing

1. A **hypothesis test** is a statistical test that is used to determine whether there is enough evidence in a sample of data to infer that a certain condition is true for the entire population. It usually consists of

   - A null hypothesis H_0 that represents a default position that there is no effect or no difference.
   - A test statistic that is used to determine whether the null hypothesis should be rejected.
   - A significance level α that determines the probability of rejecting the null hypothesis when it is true.

2. The **p-value** is the probability of observing a test statistic as extreme as the one computed from the sample data, assuming that the null hypothesis is true. If the p-value is less than the significance level α, then the null hypothesis is rejected.

※ T-Tests

1. A **t-test** is a statistical test used to determine if there is a significant difference between the means of two groups. There are different types of t-tests:

   - A **one-sample t-test** is used to compare the mean of a single sample to a known mean.

   $$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} \sim t_{n-1} \tag{35}$$

   where $\bar{x}$ is the sample mean, μ is the known mean, s is the sample standard deviation, and n is the sample size.

   - A **two-sample t-test** is used to compare the means of two independent samples.

   $$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}} \sim t_{n_1+n_2-2} \tag{36}$$

   where $\bar{x}_1$ and $\bar{x}_2$ are the sample means, s_1 and s_2 are the sample standard deviations, and n_1 and n_2 are the sample sizes.

   - A **paired t-test** is used to compare the means of two related samples.

   $$t = \frac{\bar{d}}{s_d/\sqrt{n}} \sim t_{n-1} \tag{37}$$

   where $\bar{d}$ is the mean of the differences between the paired samples, s_d is the standard deviation of the differences, and n is the number of pairs.

   after obtaining the t-statistic, the p-value can be computed using the t-distribution and then a decision can be made based on the p-value.

※ Estimators

1. An **estimator** is a rule or method for estimating the value of an unknown parameter based on observed data. An estimator is a random variable since it depends on the data.

   $$\hat{\theta} = g(X_1, X_2, \cdots, X_n) \quad \text{where} \quad X_1, X_2, \cdots, X_n \sim F_\theta \tag{38}$$

   where F is some distribution that depends on the parameter θ.

2. The **bias** of an estimator is the difference between the expected value of the estimator and the true value of the parameter being estimated.

   $$\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta \tag{39}$$

3. An estimator is **unbiased** if its expected value is equal to the true value of the parameter being estimated, for example if we have,

   $$X_i \sim \text{Normal}(\theta, 1) \quad \text{then} \quad \hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} X_i \quad \text{is unbiased since} \quad \mathbb{E}[\hat{\theta}] = \theta \tag{40}$$

4. An estimator is **consistent** if it converges in probability to the true value of the parameter being estimated as the sample size increases,

$$\hat{\theta} \xrightarrow{p} \theta \quad \text{as} \quad n \to \infty \tag{41}$$

5. The **mean squared error** (MSE) of an estimator is the expected value of the squared difference between the estimator and the true value of the parameter being estimated. This can be decomposed into the variance of the estimator and the square of the bias:

$$\text{MSE}(\hat{\theta}) = \mathbb{E}[(\hat{\theta} - \theta)^2] = \text{Var}(\hat{\theta}) + \text{Bias}(\hat{\theta})^2 \tag{42}$$

   - **Variance** is a measure of how much the estimates for the parameter vary as the sample data changes, for example the error caused by sampling variability.
   - **Bias** is a measure of how much the estimates for the parameter differ from the true value of the parameter, for example the error caused by using an incorrect or overly simplified model.

6. **Maximum Likelihood Estimation**: The maximum likelihood estimator (MLE) is an estimator that maximizes the likelihood function of the observed data, formally:

$$\hat{\theta}_{\text{MLE}} = \underset{\theta}{\text{argmax}} \prod_{i=1}^{n} f_\theta(X_i) \tag{43}$$

   where f_θ is the probability density function of the data. The MLE is consistent and asymptotically normal.

7. **Maximum A Posteriori Estimation**: The maximum a posteriori estimator (MAP) is an estimator that maximizes the posterior distribution of the parameter given the observed data and a prior distribution, formally:

$$\hat{\theta}_{\text{MAP}} = \underset{\theta}{\text{argmax}}\, f(\theta|X) = \underset{\theta}{\text{argmax}}\, f(X|\theta)\pi(\theta) \tag{44}$$

   where $f(\theta|X)$ is the posterior distribution, $f(X|\theta)$ is the likelihood function, and $\pi(\theta)$ is the prior distribution. The MAP estimator is consistent and asymptotically normal.

※ Bootstrap

9.1 Inspiration

1. The inspiration for the bootstrap comes from the idea of **resampling** the data that can be used to estimate the distribution of certain statistics.

2. Consider a sample $X_1, X_2, \cdots, X_n$ from some distribution F. We can use the bootstrap to estimate the distribution of the mean of the sample.

   - We repeat the following process B times:
     - Draw a bootstrap sample of size n from the original sample (with replacement).

   - Compute the statistic of interest on the bootstrap sample to generate an empirical distribution of the statistic.

3. The bootstrap is a non-parametric method, meaning that it does not make any assumptions about the distribution of the data.

※ Information Theory

10.1 Entropy

1. The **entropy** of a random variable X is a measure of the uncertainty in the random variable, formally defined as:

$$H(X) = -\sum_{x} p(x) \log p(x) \tag{45}$$

   where $p(x)$ is the probability mass function of the random variable. The entropy is maximized when all outcomes are equally likely (uniform distribution)

※ Linear Regression

11.1 Assumptions

1. There are 5 key assumptions that are made in linear regression:

   - **Linearity:** The relationship between the dependent and independent variables is linear.
   - **Independence:** The residuals are independent of each other,

$$\text{Cov}(\epsilon_i, \epsilon_j) \approx 0 \quad \forall i \neq j \tag{46}$$

   - **Homoscedasticity:** The residuals have constant variance (i.e the variance is not a function of the independent variables).
   - **Normality:** The residuals are normally distributed,

$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right) \quad \text{where} \quad \epsilon_i = y_i - \hat{y}_i \tag{47}$$

   - **No Multicollinearity:** The independent variables are not highly correlated with each other,

$$\text{Corr}(X_i, X_j) \approx 0 \quad \forall i \neq j \tag{48}$$

     This condition when violated can lead to unstable estimates of the coefficients due to the matrix $X^T X$ being ill-conditioned (high condition number).

11.2 Ordinary Least Squares

1. **Problem:** Given a set of data points $\{(x_i, y_i)\}_{i=1}^n$, represented by a matrix $X \in \mathbb{R}^{n \times p}$ and a vector $y \in \mathbb{R}^n$, we want to find a linear function $f(x) = x^T\beta$ that minimizes the sum of squared errors:

$$\underset{\beta}{\operatorname{argmin}} ||y - X\beta||^2 \tag{49}$$

   Note that X is usually augmented with a column of ones to account for the intercept term.

2. **Solution:** We can derive the solution to the ordinary least squares problem by setting the gradient of the loss function to zero:

$$||y - X\beta||^2 = (X\beta - y)^T(X\beta - y) = \beta^T X^T X\beta - 2\beta^T X^T y + y^T y \tag{50}$$

$$\nabla_\beta ||y - X\beta||^2 = 2X^T X\beta - 2X^T y = 0 \implies \hat{\beta} = (X^T X)^{-1} X^T y \tag{51}$$

   where $\hat{\beta}$ is the least squares estimate of the coefficients.

3. By the **Gauss-Markov theorem**, the least squares estimate is the best linear unbiased estimator (BLUE) of the coefficients.

4. The **residuals** are the differences between the observed values and the predicted values:

$$\hat{y} = X\hat{\beta} \quad \text{and} \quad \hat{e} = y - \hat{y} \tag{52}$$

   It can be shown that the residuals are orthogonal to the column space of X (the feature space):

$$\hat{e} = y - X\hat{\beta} = y - X(X^T X)^{-1} X^T y \implies X^T\hat{e} = X^T y - X^T X(X^T X)^{-1} X^T y \tag{53}$$

$$X^T X(X^T X)^{-1} = I \implies X^T\hat{e} = X^T y - X^T y = 0 \tag{54}$$

   This is also intuitive as the residuals are the component of y that is orthogonal to the column space of X (as $\hat{y}$ is the projection of y onto the column space of X).

   It can also be shown that the sum of the residuals is zero:

$$X^T\hat{e} = 0 \quad \text{and} \quad X^T = [\hat{1} \mid X] \implies \hat{1}^T\hat{e} = 0 \tag{55}$$

   where $\hat{1}$ is the vector of ones that is used to fit the intercept term.

5. The **residual sum of squares** (RSS) is the sum of the squared residuals:

$$\text{RSS} = ||y - X\hat{\beta}||^2 = \hat{e}^T\hat{e} \tag{56}$$

   And the **total sum of squares** (TSS) is the sum of the squared differences between the observed values and the mean of the observed values:

$$\text{TSS} = ||y - \bar{y}\hat{1}||^2 = (y - \bar{y}\hat{1})^T(y - \bar{y}\hat{1}) \tag{57}$$

   where $\bar{y}$ is the mean of the observed values.

6. The **coefficient of determination** R^2 is the proportion of the variance in the dependent variable that is predictable from the independent variables:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = 1 - \frac{\hat{e}^T \hat{e}}{(y - \bar{y}\hat{1})^T (y - \bar{y}\hat{1})} \tag{58}$$

the R^2 value can range from $-\infty$ to 1 and is a measure of how well the model fits the data, or how much of the variance in the dependent variable is explained by the independent variables.

7. The least squares estimator for β is a **maximum likelihood estimator** under the assumption that the residuals are normally distributed,

$$\epsilon_i = y_i - x_i^T \beta \sim \mathcal{N}(0, \sigma^2) \quad \text{where} \quad \text{PDF}(\epsilon_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\epsilon_i^2\right) \tag{59}$$

therefore we have that:

$$\text{likelihood}(\beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - x_i^T \beta)^2\right) \tag{60}$$

we maximize by considering the log-likelihood:

$$\ell(\beta) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 \propto \min_{\beta} ||y - X\beta||^2 \tag{61}$$

which is clearly proportional to minimizing the least squares loss.

11.3 Regularization

1. When $X \in \mathbb{R}^{n \times p}$ is not full rank, the matrix $X^T X$ is not invertible and the least squares estimate $\hat{\beta} = (X^T X)^{-1} X^T y$ does not exist. this can be seen by the fact that,

$$\text{rank}(X^T X) \leq \min(n, p) \quad \text{and} \quad \text{rank}(X) \leq \min(n, p) \quad \text{with} \quad p < n \tag{62}$$

hence the matrix $X^T X$ is not full rank.

Another way $X^T X$ isn't invertible is when $n < p$, in this case the system is underdetermined and there are infinitely many solutions to the least squares problem.

2. In both of these cases, we can use **regularization** to stabilize the estimates of the coefficients.

11.4 Ridge Regression

1. **Ridge regression** adds a penalty term proportional to the L_2 norm of the coefficients to the least squares loss function:

$$\underset{\beta}{\text{argmin}} ||y - X\beta||^2 + \lambda ||\beta||_2^2 \tag{63}$$

where λ is the regularization parameter that controls the strength of the penalty term.

2. **Solution:** We can derive the solution to ridge:

$$||y - X\beta||^2 + \lambda||\beta||_2^2 = (X\beta - y)^T(X\beta - y) + \lambda\beta^T\beta \tag{64}$$
$$\nabla_\beta||y - X\beta||^2 + \lambda||\beta||_2^2 = 2X^TX\beta - 2X^Ty + 2\lambda\beta = 0 \tag{65}$$
$$\implies \hat{\beta} = (X^TX + \lambda I)^{-1}X^Ty \tag{66}$$

where I is the identity matrix.

3. **Rank Analysis:** We can see that if X^TX is not full rank, then matrix $X^TX + \lambda I$ must be full rank. We can see this by considering the eigenvalues of X^TX and $X^TX + \lambda I$:

$$\alpha \in \text{eig}(X^TX) \implies X^TXv = \alpha v \tag{67}$$

Consider the eigenvalues of $X^TX + \lambda I$:

$$(X^TX + \lambda I)v = X^TXv + \lambda v = \alpha v + \lambda v = (\alpha + \lambda)v \tag{68}$$

hence if we have a set of eigenvalues α for X^TX, we will have a set of eigenvalues $\alpha + \lambda$ for $X^TX + \lambda I$, and since X^TX is a positive semi-definite matrix, the eigenvalues of $X^TX + \lambda I$ will be strictly positive. Therefore we have that,

$$\det(X^TX) = \prod_{i=1}^{p} \alpha_i \quad \text{and} \quad \det(X^TX + \lambda I) = \prod_{i=1}^{p}(\alpha_i + \lambda) \tag{69}$$

and since the determinant of a matrix is the product of its eigenvalues, we have that the matrix $X^TX + \lambda I$ is full rank.

Note that the matrix can still be ill-conditioned if the eigenvalues of X^TX are close to zero, as the eigenvalues of $X^TX + \lambda I$ will be close to λ, and if λ is small then the condition number of the matrix will remain large.

4. The ridge regression estimate is a **maximum a posteriori (MAP)** estimate under the assumption that the coefficients are normally distributed with mean zero and variance proportional to $1/\lambda$:

$$\beta_i \sim \mathcal{N}(0, \lambda) \implies \text{PDF}(\beta_i) = \frac{1}{\sqrt{2\pi\lambda}} \exp\left(-\frac{1}{2\lambda}\beta_i^2\right) \tag{70}$$

using Bayes' theorem, we have that:

$$\text{likelihood}(\beta) \propto \exp\left(-\frac{1}{2\sigma^2}||y - X\beta||^2\right) \quad \text{and} \quad \text{prior}(\beta) \propto \exp\left(-\frac{1}{2\lambda}||\beta||^2\right) \tag{71}$$

therefore we can take the product of the likelihood and the prior to get the posterior:

$$\text{posterior}(\beta) \propto \exp\left(-\frac{1}{2\sigma^2}||y - X\beta||^2 - \frac{1}{2\lambda}||\beta||^2\right) \tag{72}$$

and maximizing the posterior is equivalent to minimizing the ridge regression loss.

11.5 Lasso Regression

※ Trading (Game Theory)

12.1 Making Markets

1. When **making a market** the most important thing to consider is the fair value of whatever it is you are trying to price. For example that could be the expectation of a sum of random variables,

$$\mathbb{E}[X_1 + X_2 + \cdots + X_n] \quad \text{where} \quad X_i \sim \text{Coin Flip}(p) \tag{73}$$

   in this case we would be trading on the sum of n coin flips with probability of heads p.

2. The **bid-ask spread** is the difference between the price at which you are willing to buy and the price at which you are willing to sell.

   - In general for less liquid assets you would want to have a larger spread, as the risk of holding the asset is higher.
   - For higher variance assets you would also want to have a larger spread as the probability that the asset will move against you is higher. For example if we are trading,

$$X_1 + X_2 + \cdots + X_n \quad \text{where} \quad X_i \sim \text{Normal}(0, 1) \tag{74}$$

     the variance of the sum is n, while in the following case:

$$X_1 + X_2 + \cdots + X_n \quad \text{where} \quad X_i \sim \text{Normal}(0, 0.1) \tag{75}$$

     the variance of the sum is $0.1n$, hence the spread should be larger in the first case, even though the expected value of the sum is the same.

3. **Impact Function**: When someone trades (typically a large order), on your market it makes sense to adjust your theoretical price, for example if you had a market on X,

$$\text{Bid} \quad [100] \quad 95.0 \quad - \quad 105.0 \quad [100] \quad \text{Ask} \tag{76}$$

   and someone buys 100 shares at 105, you would want to think to yourself, "*why did they buy at* 105*?*".

   - The buyer could have information that moves the fair value up, and they wanted to profit off of that.
   - The buyer could just be willing to pay the spread to get the shares quickly to hold for longer
   - The buyer could be uninformed and just buying because they think the price will go up.

   In the first case (espcially if you think the buyer is informed), you would want to adjust your theoretical price up so that you don't get crushed by future informed traders.

   In the other cases you might not want to adjust your price as much, as the buyer might not have any information that you don't have and you would be losing money by adjusting your price.

※ Practical Statistical Learning

13.1 Cross Validation

1. **Cross Validation** is a technique used to estimate the performance of a model on an independent dataset.

2. The basic idea is to split the data into a training set and a validation set, and then use the training set to fit the model. Formally,

$$\text{Data} \rightarrow [\text{Train Set} \mid \text{Validation Set} \mid \text{Test Set}] \tag{77}$$

3. The most common type of cross validation is **k-fold cross validation**, where

Algorithm 1 K-Fold Cross-Validation

Input: Dataset D, number of folds k, model M, hyperparameters p, performance metric Score
Output: Average performance score S_{avg}
Shuffle D randomly and partition into k equal-sized folds $\{F_1, F_2, \ldots, F_k\}$ ▷ Adjust if $|D|$ not divisible by k
Initialize scores list $S \leftarrow []$
for $i = 1$ to k **do** ▷ Iterate over each fold
    Define test set $D_{\text{test}} \leftarrow F_i$
    Define training set $D_{\text{train}} \leftarrow D \setminus F_i$ ▷ Union of all other folds
    Train model M on D_{train} using hyperparameters p
    Evaluate M on D_{test} to compute $s_i = \text{Score}(M, D_{\text{test}})$
    Append s_i to S
Compute $S_{\text{avg}} \leftarrow \frac{1}{k} \sum_{i=1}^{k} s_i$ ▷ Average score across folds
Return S_{avg}

4. In practice this helps reduce overfitting and helps us get a better estimate of the performance of the model.

5. For time series data, we can use **rolling cross validation** where we use a window of the data to fit the model and then use the next observation as a validation set.

13.2 Grid Search

1. **Grid Search** is a technique used to find the optimal hyperparameters for a model.

2. The basic idea is to search over a grid of hyperparameters and select the one that minimizes the error of the model.

3. Formally,

Algorithm 2 Grid Search for Hyperparameter Tuning

Input: D_{train}, D_{val}, M, hyperparameter sets $P_1, P_2, \ldots, P_k$, Score Function
Output: Best hyperparameters P_{best}, best score S_{best}
Initialize $S_{\text{best}} \leftarrow -\infty$ ▷ Assuming higher score is better; adjust for minimization
Initialize $P_{\text{best}} \leftarrow \emptyset$
Generate all possible combinations C of hyperparameters from $P_1 \times P_2 \times \cdots \times P_k$ ▷ Cartesian product
for $p \in C$ **do** ▷ Iterate over each combination
    Train model M on D_{train} using hyperparameters p
    Evaluate M on D_{val} to compute $s = \text{Score}(M, D_{\text{val}})$
    if $s > S_{\text{best}}$ **then**
        $S_{\text{best}} \leftarrow s$
        $P_{\text{best}} \leftarrow p$
Return $P_{\text{best}}, S_{\text{best}}$

13.3

※ Pandas

14.1 DataFrames

1. A DataFrame is a 2-dimensional labeled data structure with columns of potentially different types, some common operations on DataFrames are:

   - **Selecting columns:** Columns can be selected using the column name as an attribute or as a key.
   - **Selecting rows:** Rows can be selected using the `loc` and `iloc` methods.
   - **Filtering rows:** Rows can be filtered using boolean indexing.
   - **Applying functions:** Functions can be applied to columns using the `apply` method.
   - **Grouping data:** Data can be grouped using the `groupby` method.
   - **Merging data:** Data can be merged using the `merge` method.

   Some examples of these operations are:

```python
import pandas as pd
# Create a sample DataFrame
data = {
    'name':   ['Alice', 'Bob', 'Charlie', 'Dave', 'Eve'],
    'age':    [24, 42, 18, 68, 32],
    'city':   ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix'],
    'salary': [50000, 60000, 45000, 70000, 65000]
}
df = pd.DataFrame(data)
# 1. Selecting columns
# Using the column name as a key
names = df['name']
```

```
# Using the column name as an attribute (if it doesn't conflict with method names)
ages = df.age
# 2. Selecting rows
# Using label-based indexing (loc)
first_row_loc = df.loc[0]      # Select row with label 0
subset_loc = df.loc[1:3]       # Select rows with labels 1 through 3
# Using integer-based indexing (iloc)
first_row_iloc = df.iloc[0]    # Select the first row
subset_iloc = df.iloc[1:3]     # Select the 2nd and 3rd rows
# 3. Filtering rows (boolean indexing)
older_than_30 = df[df['age'] > 30]
# 4. Applying functions
# Apply a lambda function to increase salary by 10%
df['salary_with_bonus'] = df['salary'].apply(lambda x: x * 1.1)
# 5. Grouping data
# Calculate the mean salary for each city
mean_salary_by_city = df.groupby('city')['salary'].mean()
# 6. Merging data
# Suppose we have another DataFrame df2 that shares a common key 'name'
df2 = pd.DataFrame({
    'name':   ['Alice', 'Bob', 'Eve'],
    'bonus':  [5000, 7000, 4000]
})
merged_df = pd.merge(df, df2, on='name', how='left')
```

14.2 Summary Statistics

1. Some common summary statistics for a DataFrame `df` are:

   - **Descriptive statistics:** The `describe` method provides summary statistics for numerical columns.
   - **Correlation matrix:** The `corr` method provides the correlation matrix for numerical columns.
   - **Unique values:** The `nunique` method provides the number of unique values for each column.
   - **Value counts:** The `value_counts` method provides the frequency of each unique value in a column.
   - **Missing values:** The `isnull` method provides a DataFrame of missing values.

   Some examples of these operations are:

```
# 1. Descriptive statistics
summary_stats = df.describe()
# 2. Correlation matrix
correlation_matrix = df.corr()
# 3. Unique values
unique_values = df.nunique()
# 4. Value counts
value_counts = df['city'].value_counts()
```

```
# 5. Missing values
missing_values = df.isnull()
```

14.3 Data Cleaning

1. A lot of the time data is not clean and therefore needs preprocessing before it can be used for analysis. Some of the basic operations for data cleaning are:

   - **Removing duplicates:** Duplicates can be removed using the `drop_duplicates` method.
   - **Filling missing values:** Missing values can be filled using the `fillna` method.
   - **Replacing values:** Values can be replaced using the `replace` method.
   - **Changing data types:** Data types can be changed using the `astype` method.

   Some examples of these operations are:

   ```
   # 1. Removing duplicates
   df_no_duplicates = df.drop_duplicates()
   # 2. Filling missing values as 0
   df_filled = df.fillna(0)
   # 3. Replacing values
   df_replaced = df.replace('New York', 'NY')
   # 4. Changing data types
   df['age'] = df['age'].astype(float)
   ```

2. **One-Hot Encoding**: for data given in a categorical form, it is often useful to convert it to a numerical form. This can be done using the `get_dummies` method.

   ```
   # Convert the 'city' column to dummy variables
   df_with_dummies = pd.get_dummies(df, columns=['city'])
   ```

   This will create a new column for each unique value in the 'city' column with a 1 if the value is present and a 0 otherwise.

3. **Clipping**: Sometimes it is useful to clip the values of a column to a certain range. This can be done using the `clip` method.

   ```
   # Clip the 'age' column to be between 18 and 65
   df_clipped = df['age'].clip(18, 65)
   ```

14.4 Time Series

1. Time series data is data that is indexed by time. Some common operations on time series data are:

   - **Resampling:** Time series data can be resampled using the `resample` method.
   - **Shifting:** Time series data can be shifted using the `shift` method.

- **Rolling windows:** Rolling windows can be applied to time series data using the `rolling` method.

Some examples of these operations are:

```
# 1. Resampling
# Resample the data to monthly frequency
df_resampled = df.resample('M').mean()
# 2. Shifting
# Shift the data by 1 period (move the data down by 1)
df_shifted = df.shift(1)
# 3. Rolling windows
# Calculate the 7-day rolling mean
df_rolling = df.rolling(window=7).mean()
```

※ NumPy

15.1 Arrays

1. NumPy arrays are the core data structure for numerical computations in Python. Some common operations on NumPy arrays are:

   - **Creating arrays:** Arrays can be created using the `array` function or using convenience functions like `zeros`, `ones`, and `arange`.
   - **Indexing:** Elements of an array can be accessed using square brackets.
   - **Slicing:** Subarrays can be accessed using slicing.
   - **Reshaping:** Arrays can be reshaped using the `reshape` method.
   - **Stacking:** Arrays can be stacked vertically or horizontally using the `vstack` and `hstack` functions.
   - **Broadcasting:** Operations can be performed on arrays of different shapes using broadcasting.

   Some examples of these operations are:

```
import numpy as np
# 1. Creating arrays
a = np.array([1, 2, 3])
b = np.zeros((2, 3))
c = np.ones((3, 2))
d = np.arange(10)
# 2. Indexing
first_element = a[0]
# 3. Slicing
first_two_elements = a[:2]
# 4. Reshaping
e = d.reshape((2, 5))
# 5. Stacking
f = np.vstack((b, c))
```

```
g = np.hstack((b, c))
# 6. Broadcasting
h = a + 1
```

※ Scikit-Learn

References

[1] Williams, David. *Probability with Martingales*. Cambridge University Press, 1991. Print.