# Multivariate probability

Adam Boult (www.bou.lt)

April 30, 2025

# Contents

# Preface

This is a live document, and is full of gaps, mistakes, typos etc.

# Part I

# Exploratory data analysis

# Chapter 1

# Summary statistics for multiple variables

## 1.1 Statistics for two variables

### 1.1.1 Sample covariance

We previously defined the population covariance as $\sigma_{XY} = E[(X - \mu_X)^T(Y - \mu_Y)]$.

We define the sample covariance as $\sigma_{XY} = \frac{1}{n}\sum_i (x_i - \bar{x})(y_i - \bar{y})$.

We can calculate this using matrices:

$M = X - \bar{x}$

$N = Y - \bar{y}$

$\sigma_{XY} = \frac{1}{n} M^T N.$

### 1.1.2 Sample correlation

$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$

### 1.1.3 Covariance matrix

If we have $n$ variables we can have a $n \times n$ matrix $\Sigma$ where:

$\Sigma_{ij} = \sigma_{ij} = \frac{1}{n}(X_i - \bar{x}_i)^T(X_j - \bar{x}_j)$

### 1.1.4 Centred covariance

If $\bar{x} = \bar{y} = 0$ then:

$$\sigma_{XY} = \frac{1}{n} X^T Y$$

### 1.1.5 Correlation matrix

Here each entry is the correlation rather than the covariance.

## 1.2 Correlation coefficients

### 1.2.1 Pearson correlation coefficient

The Pearson correlation coefficient is definited as the covariance normalised by the individual variances.

It is between $-1$ (total negative linear correlation), 0 (no linear correlation) and 1 (total negative linear correlation).

$$p_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

### 1.2.2 Spearman rank correlation

For each of 2 variables we create a ranking of them.

From $X$ and $Y$ we then have $R_X$ and $R_Y$.

We then calculate the Pearson correlation coefficient between the rankings.

$$r_S = \frac{cov(R_X, R_Y)}{\sigma_{R_X} \sigma_{R_Y}}$$

### 1.2.3 Kendall rank correlation

Concordant and discordant pairs

$$\tau = \frac{n_{concordant} - n_{discordant}}{\binom{n}{2}}$$

### 1.2.4 General correlation coefficient

## 1.3 Updating statistics

### 1.3.1 Updating the covariance

If it is centred:

$\sigma^n_{XY} = \frac{1}{n} X^T_n Y_n$

So:

$\sigma^{n+1}_{XY} = \frac{n\sigma^n_{XY} + x^t_{n+1} y_{n+1}}{n+1}$

### 1.3.2 Sparklines

# Chapter 2

# Visualisation for multiple variables

## 2.1 Visualising multiple continuous variables

### 2.1.1 Time series

### 2.1.2 Scatter plots (with size as variable)

### 2.1.3 Q-Q plots

Plot quartiles of variables against each other.

## 2.2 Visualising a single class variable

### 2.2.1 Bar and column charts

### 2.2.2 Pie charts

## 2.3 Visualising multiple class variables

### 2.3.1 Stacked bar and column charts

## 2.4 Visualising class and continous variables

### 2.4.1 Multiple box and whiskers

### 2.4.2 Scatter plots with colour

## 2.5 Visualising geographic data

## 2.6 Visualising time series

### 2.6.1 Heat maps

# Chapter 3

# Distance metrics and outliers

## 3.1 Measuring distance between vectors

### 3.1.1 $L_p$ norms

$L_p$ norms can be used to measure the distance between two metrics.

If we have data points $v$ and $w$ the distance is:

$$||v - w|| = (\sum_i |v_i - w_i|^p)^{\frac{1}{p}}$$

If $p = 2$ we have the Euclidian norm. If $p = 1$ we have the Manhatten norm.

### 3.1.2 Dot product

Given two vectors we can calculate:

$$\frac{a.b}{||a||||b||}$$

If the two vectors are identical, this is 1. If they are orthogonal this is 0. If they are opposite, this is $-1$.

### 3.1.3 Kernels

This is a generalisation of the dot product function, where we want to find similarity between two vectors.

If we have data points $v$ and $w$ the distance is:

$K(v, w)$

### 3.1.4 Mahalanobis distance

We have a point. How far away is this from the mean.

For a single dimension: number of standard deviations.

What about multidimensional data?

Could do sd for all distances, but correlations between variables. If two variables are highly correlated, it's not really twice as far.

We use this:

$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T S^{-1} (\mathbf{x} - \mu)}$

## 3.2 Measuring distance between matrices

### 3.2.1 Frobenius norm

If we have matrices $A$ and $A$ the distance is:

$||A - B|| = \sqrt{\sum_i \sum_j |a_{ij} - b_{ij}|^2}$

This is a Euclidian norm.

## 3.3 Measuring distance between time series

### 3.3.1 Dynamic time warping

We may want to examine the similarity between two sequences.

We want to match a sample from one sequence to a sample from the other sequence.

Simply matching at the same time point is naive, as samples may move at different speeds, or have offsets.

## 3.4 Finding neighbours

### 3.4.1 Nearest Neighbour Search (NSS)

### 3.4.2 Finding neighbours

Say we have a distance function and a sample. How can we identify the $k$-nearest neighbours?

We can find the distance for all points, sort this and take the top $k$ observations.

### 3.4.3 k-Nearest Neighbour Search

# Part II

# Unsupervised machine learning

# Chapter 4

# Dimensionality reduction with Principal Component Analysis (PCA)

## 4.1 Dimensionality reduction

### 4.1.1 Classical principal component analysis

#### Introduction

Principal component analysis takes a dataset $X$ with $m$ variables and returns a principal component matrix $A$ with size $m \times k$.

Each new dimension is a linear function of the existing data. $Z = XA$.

Each dimension in uncorrelated, and ordered, in order of descending explanation of variability.

The problem of principal component analysis is to find these weightings $A$.

#### Classical PCA

We take the first $k$ eigenvectors of the covariance matrix, ordered by eigenvalue.

#### Getting the eigenvectors using SVD

We can decompose $X = U\Sigma A^T$.

We can take the eigenvectors from $A$.

**Choosing the number of dimension**

We can choose $k$ such that a certain percentage of the variance is retained.

### 4.1.2 Robust principal component analysis

**Robust PCA**

Robust PCA can be used to deal with corrupted data, such as corrupted image data.

Rather than data $X$ we have $M = L_0 + S_0$ where $L_0$ is what we want to recover (and is low rank), and $S_0$ is noise (and sparce).

In video footage, $L_0$ can correspond to the background, while $S_0$ corresponds to movement.

# Chapter 5

# K-means and k-mediods clustering

## 5.1 Clustering

### 5.1.1 Evaluating clusterings

#### Davies-Bouldin index

The Davies-Bouldin index is a method for evaluating clustering algorithms, such as k-means.

It examines the distance between centroids, and the tightness of centroids.

### 5.1.2 k-means clustering

#### Introduction

K-means clustering is the most widely used unsupervised model.

In k-means clustering we identify $k$ centroids in the feature space. We then calculate the distance from each data point to each of the centroids, and allocate the data point to the nearest centroid.

This requires a method for calculating the location of the centroids.

#### Identifying the centroids

We apply an iterative approach to identifying the centroids.

We first initialise by assigning centroids randomly to existing data points.

We then iteratively perform the following:

- Calculate the distances between each data point and each centroid.
- Assign each data point to the closed centroid.
- Update each centroid location to the mean of the data points allocated to it.

#### Calculating distances

This method requires us to calculate the distance between two points in the feature space.

For k-means we use the Euclidian distance.

#### Potential issues

It is possibile for a centroid to have no data assigned to it. If this happens we can eliminate the cluster, or reassign some data points.

The algorithm may only arrive at a local minima. In order to maximise the chance of an effective clustering, we can do k-means under different initialisations of the centroids in order to minimise risk of bad local optima.

#### Choosing $k$

If the points in each cluster follow a normal distribution, that's a good sign. This can be tested with Anderson-Darling.

If it's not normal, we can split the cluster into 2.

#### Using clusting as part of data analysis

We can choose $k$ if output is being used in later data analysis (eg type assignment, complaint level or something)

### 5.1.3 k-medoids

#### Introduction

k-mediods is similar to k-means clustering, with two key differences:

- Centroids are now always located on data points, rather than floating freely.
- We mimimise $l_1$ distance, rathern than $l_2$.

#### Partitioning Around Medoids (PAM) algorithm

This is the most common approach for k-medoids.

We initialise randomly, as we do for k-means.

We then iterate the following:

- Calculate the loss for the current allocation
- For each medoid, see if swapping allocation with another (non-medoid) data point decreases the cost.
- If it does, make the swap.

# Part III

# Estimating generative probability distributions

# Chapter 6

# M-estimators

## 6.1 M-estimators

### 6.1.1 Introduction

page setting out linear stuff to come

OLS, generalised linea rmodels etc are m-estimators, as are gmm

h3 on parametric

With maximum likelihood estimation we maximise a function.

We could choose other functions to maximise or minimise.

$\sum_i f(x_i, \theta)$

If $f(x_i, \theta)$ is differentiable wrt to $\theta$ this can be solved by finding the stationay point.

This is a $\phi$ type.

Otherwise it is a $\rho$ type.

page on influence funcitons there

Generalisation of MLE.

$m_\theta = m_\theta(x, \theta)$

Z-estimator is where this is met, through diff

$\frac{\delta}{\delta\theta} m_\theta = z_\theta(\theta, x) = 0$

M-estimator for mean

$m_\theta(\theta) = -(x - \theta)^2$

$z_\theta(\theta) = x - \theta$

# Chapter 7

# The Generalised Method of Moments (GMM)

## 7.1 Generalised Method of Moments (GMM)

### 7.1.1 Difference from Method Of Moments (MOM)

More conditions than data.

### 7.1.2 Generalised Method of Moments (GMM)

We have a function on the output and a parameter:

$g(y, \theta)$

A moment condition is that the expectation of such a function is 0.

$m(\theta) = E[g(y, \theta)] = 0$

To do GMM, we estimate this using:

$\hat{m}(\theta) = \frac{1}{n} \sum_i g(y_i, \theta)$

We define:

$\Omega = E[g(y, \theta) g(y, \theta)^T]$

$G = E[\Delta_\theta g(y, \theta)]$

And then minimise the norm:

$||\hat{m}(\theta)||^2_W = \hat{m}(\theta)^T W \hat{m}(\theta)$

Where $W$ is a positive definite matrix for the norm.

$\Omega^{-1}$ is most efficient. But we don't know this. It depends on $\theta$.

We can estimate it if IID:

$\hat{W}(\hat{\theta}) = (\frac{1}{n}\sum_i g(y,\hat{\theta})g(y,\hat{\theta})^T)^{-1}$

### 7.1.3 Two-step feasible GMM

Estimate using $\mathbf{W} = \mathbf{I}$

Consistent, but not efficient.

### 7.1.4 Moment conditions

OLS:

$E[x(y - x\theta)] = 0$

WLS

$E[x(y - x\theta)/\sigma^(x)] = 0$

IV

$E[z(y - x\theta)] = 0$

MLE

$E[\Delta_\theta \ln f(x,\theta)] = 0$

### 7.1.5 New GMM

$m(\theta_0) = E[g(\mathbf{x}_i, \theta_0]$

We replace this with sample moment

$\hat{m}(\theta) = \frac{1}{n}\sum_i g(\mathbf{x}_i, \theta)$

We have the "score"

$\nabla_\theta g(\mathbf{x}_i, \theta_0)$

Information

$G = E[\nabla_\theta g(\mathbf{x}_i, \theta_0)]$

Variance-covariance loss matrix

$\Omega = E[g(\mathbf{x}_i, \theta_0)g(\mathbf{x}_i, \theta_0)^T]$

We want to minimise moment loss

$||\hat{m}(\theta)||^2_W = \hat{m}(\theta)^T W \hat{m}(\theta)$

$\hat{\theta} = argmin_\theta(\frac{1}{n}\sum_i g(\mathbf{x}_i, \theta))^T \hat{W}(\frac{1}{n}\sum_i g(\mathbf{x}_i, \theta))$

### 7.1.6 Asymptotic

CLT means normal.

They are consistent IF moment condition is true.

There is an explicit formula for variance.

$\sqrt{n}(\hat{\theta} - \theta_0) \rightarrow^d N[0, (G^T W G)^{-1} G^T W \Omega W^T G (G^T W^T G)^{-1}]$

If we choose $W \propto \Omega^{-1}$ then:

$\sqrt{n}(\hat{\theta} - \theta_0) \rightarrow^d N[0, (G^T \Omega^{-1} G)^{-1}]$

Problem: we need to estimate $\Omega$ and $G$.

$\Omega$: estimate from sample. allows us to choose estimator, but still leaves variance unidentified.

Do the above from OLS? This is where robust etc stuff comes from

If it is specified. Moment conditions are equal to the number of moments, then $W$ doesn't matter. This is normal Method of Moments.

Estimating the weighting matrix

### 7.1.7 Iterated GMM

### 7.1.8 Moment-covariance matrix

### 7.1.9 Bias and variance of the GMM estimator

page on Bias and variance of the GMM estimator (cluster assumption should be part of moment condition?) part of later calculation of weighting?

Can do robust, hac, clustering as part of GMM too.

# Part IV

# Other

# Chapter 8

# Association rules

## 8.1 Association rules

### 8.1.1 Association rules

**The data**

We have a transaction dataset, $D$.

This includes transactions of items in $I$.

Any subset of $I$ is an itemset.

A subset of size $k$ is a $k$-itemset.

Transactions are a $k$-itemset with a unique id, tid.

The set of all transactions is $T$.

A tidset is a subset of $T$.

**Forming a lattice**

We have a total order on the items.

An itemset $ab$ is greater than $a$, for examplle.

The two points of the lattice are the nullset, and $I$.

**Mappings**

We have a mapping from $I$ to $T$ called $t$.

We have another mapping from $T$ to $I$ called $i$.

**Frequency**

We define the frequency of an itemset as the number of transactions it appears in.

We can write the frequency of $A$ as $\sum A$.

### 8.1.2 Strong rules

**Strong rules**

We use frequent patterns to generate strong rules, $R$.

An example of a strong rule is $a \rightarrow b$.

We can look at this by comparing the support of $a$ to $a \wedge b$.

$supp(A \rightarrow B) = P(A \wedge B)$

$conf(A \rightarrow B) = P(B|A)$

$$conf(A \rightarrow B) = \frac{P(A \wedge B)}{P(A)}$$

### 8.1.3 Support

**Support**

We define the support of an itemset as the proportion of transactions which contain the itemset.

$$supp(A) = \frac{\sum A}{n}.$$

We can also consider this as:

$supp(A) = P(A)$

### 8.1.4 Frequent patterns

**Frequent patterns**

A frequent itemset is one where the support is above a minimum.

We know that if an itemset is frequent, then all its subsets are also frequent.

We look for frequent patterns, $F$, between the items $I$.

An example of a frequent pattern is $\{a, b\}$.

### 8.1.5 Confidence

**Confidence**

The confidence of a frequent pattern is defined as:

$$conf(A \rightarrow B) = \frac{supp(A \wedge B)}{supp(A)}$$

### 8.1.6 Finding strong rules using the Apriori algorithm

**Finding frequent patterns**

We can use search algorithms to find frequent patterns. Starting at the empty set.

**Apriori algorithm**

Breadth first search to generate candidate set of itemsets with support above some value.

Start with a 1-itemset, and increase $k$ once done.

Once we have found a frequent pattern, we can immediately identify other frequent patterns associated with it.

We can do this by looking at confidence, not support.

### 8.1.7 Interest

**Interest**

An alternative measure for finding rules is to use interest.

$$Interest(A \rightarrow B) = \frac{P(A \wedge B)}{P(A)P(B)}$$

$$Interest(A \rightarrow B) = \frac{supp(A \wedge B)}{P(supp(A))P(supp(B))}$$

If this is 1, then they are independent.

If this is greater than 1, they are positively dependent.

If this is less than 1, they are negatively dependent.

### 8.1.8 Quantitative association rules

**Quantitative association rules**

The search space is infinite in size. For example continuous age.

We choose intervals instead.