Introduction to Statistical Learning *with applications in Python*

Based on "Introduction to Statistical Learning, with applications in R" by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibishirani

High Dimensons

High Dimensional Data, The Curse of High Dimensions Dimension Reduction, Principal Components, Partial Least Squares

Kurt Rinnert

Physics Without Frontiers

The Abdus Salam
International Centre
for Theoretical Physics

UNIVERSITY OF
LIVERPOOL

Abstract

We will only cover a subset of the sections in chapter 6 of the ISL book, namely the problem of high dimensional data.
High dimensional data has become more and more common in statistical learning applications. It it therefore important to understand the implications and modern methods to mitigate the problem.

Overview

- High Dimensional Data
- The Curse of High Dimensions
- Feature Selection
- Principal Components
- Partial Least Squares

This is part of what the ML community calls *feature enginieering*.

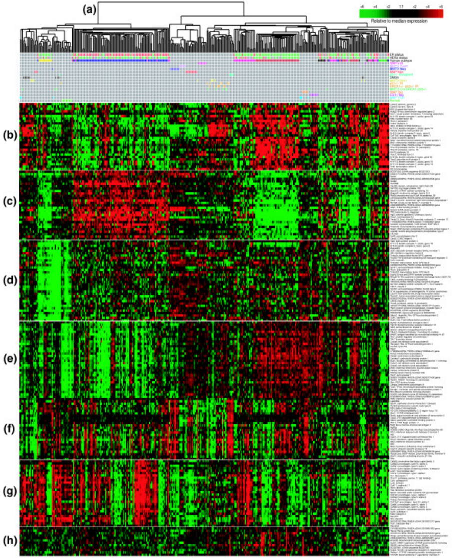
High Dimensional Data

- Traditionally, most statistical learning techniques only involved a small number of features, $p \sim \mathcal{O}(10)$, $n \gg p$.
- That is, the data was mostly *low dimensional*.
- For example, one might try to predict blood pressure from a few variables like BMI, age and gender.
- There are two reasons for this:
    1. High dimensional data could not be collected and stored.
    2. The involved computations were impossible to perform.

This situation has changed dramatically over the last decades.

High Dimensional Data

- Genetic data is a typical example.
- Instead of just BMI, age gender we now have access to measurements of many single nucleotide polymorphisms (SNP).
- At the same time, the number of individual observation is not much larger.
- A situation like $n \approx 200$ and $p \approx 500000$ is not uncommon.



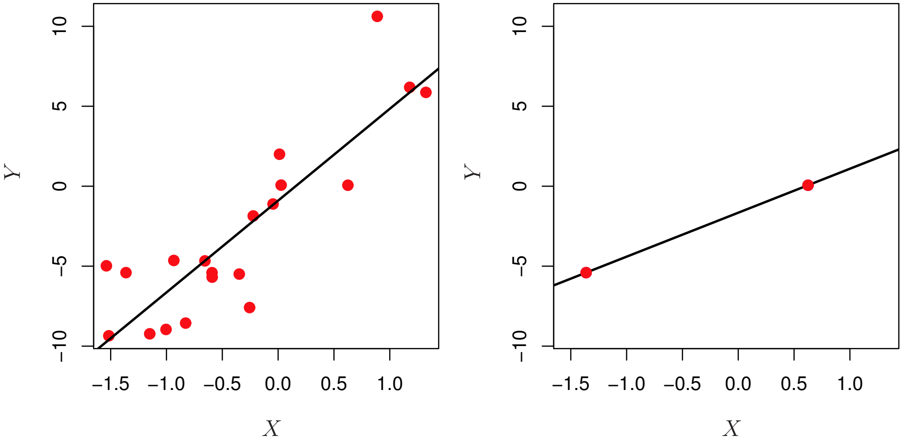
What we learned so far stops working.

High Dimensional Data

- We refer to data sets with $p \gg n$ as *high dimensional*.
- Regression, in particular least squares, is impossible in these cases.
- But even if $n > p$ problems can arise.
- If n is not much larger than p we are in danger of over-fitting.
- At the very least we have to think carefully about the bias-variance trade-off.
- Supervised learning on high dimensional data is tricky.

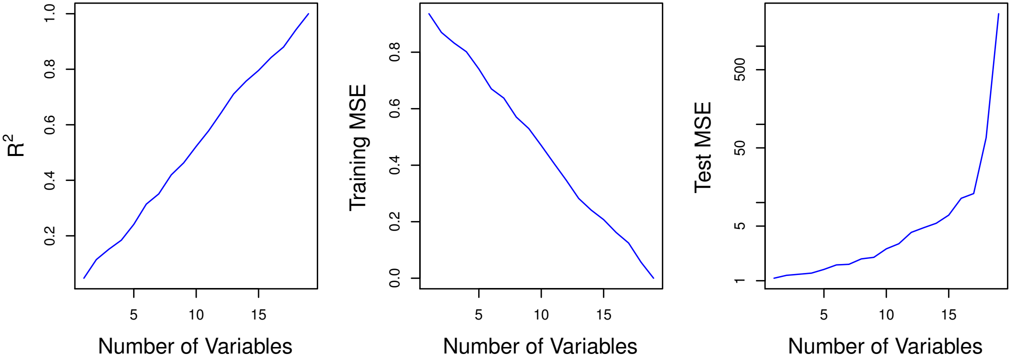
We can resort to unsupervised learning, but there also other ways.

What Goes Wrong in High Dimensions?



In high dimensions even simple models can be too flexible.

What Goes Wrong in High Dimensions?



Statistics, like R^2, can be misleading when predictors are not related to the response.

Interpretations in High Dimensions

- We have to very careful when drawing conclusions from high dimensional data sets.
- For example, suppose we want to find SNPs in an individuals DNA that are correlated with certain phenotypic traits.
- The traits could be affinities to certain diseases.
- Now let's say we find 17 SNPs out of 500,000 that seem to be related to high blood pressure.
- We should be very careful with the interpretation of this finding.
- It is *very likely* that we just found *one of many* models that suggests a correlation.
- That is, there might be a large number of subsets of similar size that wrok equally well.

This is by no means an artificial example!

Dimension Reduction: Feature Selection

- We can try to reduce the dimensionality by excluding features.

  1. *Forward Selection*: start with only the intercept and add predictors with the lowest RSS. Proceed until some cutoff is reached.
  2. *Backward Selection*: Start with all variables and keep removing the ones with the largest p-values. Stop when a cutoff is reached.
  3. *Mixed Selection*: Like forward selection, but remove variables that acquire large p-values after another variable is added.

- This is computationally expensive and of limited use for very high dimensions ($p \gg n$).

We have seen this before and won't elaborate further on this.

Dimension Reduction: Principal Components

- The idea of *principal component analysis* (PCA) is to find *linear combinations* of predictors that captures the relevant information present in the features.
- PCA allows us to directly quantify which linear combinations are most useful for prediction.
- Let $Z_1, Z_2, \ldots, Z_M$ represent $M < p$ linear combinations of the p predictors:

$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j$$

  where $m = 1, 2, \ldots, M$.
- Unsurprisingly, this is nothing but the matrix multiplication $\boldsymbol{Z} = \boldsymbol{\phi} \boldsymbol{X}$.

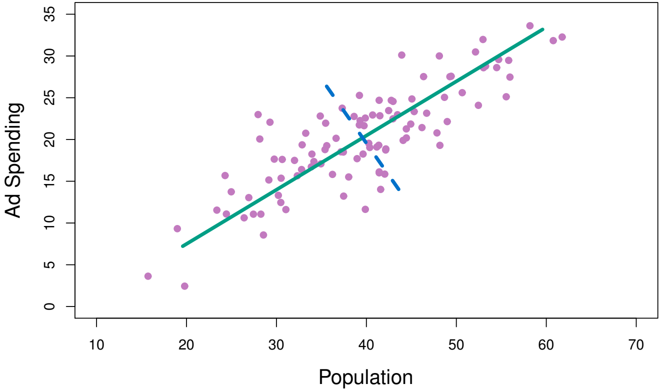
Note that the responses are *not* used. Therefore PCA is an unsupervised approach.

Dimension Reduction: Principal Components

- PCA finds the directions in the feature space along which the *observations vary the most*.
- We can then pick the M combinations (directions) with the highest variance and train (fit) our model using $Z_1, \dots, Z_M$ instead of the original p predictors.
- That way, we have *reduced the dimension* from p to M.
- Note that we do *not* make any assumption what kind of model we are going to train.

The question is how do we identify these directions.

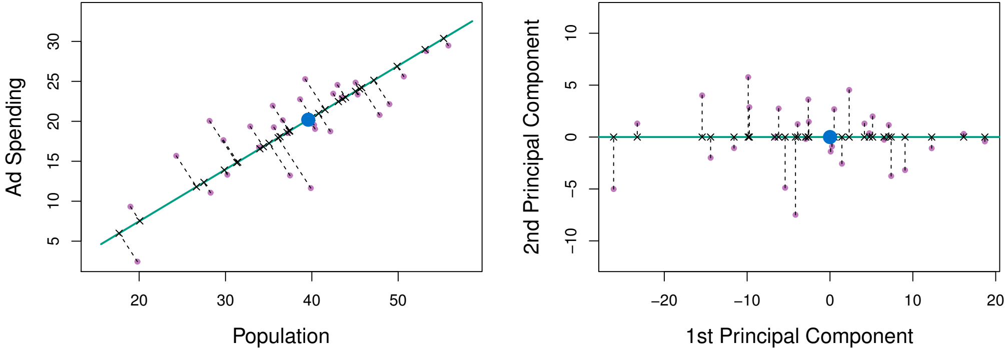
Dimension Reduction: Principal Components



- The green solid line indicates the *first principal component*.
- The blue dashed line indicates the *second principal component*.

The book is a bit shy on the math. We will tell you what really happens.

Dimension Reduction: Principal Components



- Left: before the linear transformation.
- Right: after the linear transformation.

This is nothing but a rotation in feature space!

Diagonalising Matrices

- If a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ can be diagonalised we have:

$$\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix} \implies \boldsymbol{A}\boldsymbol{P} = \boldsymbol{P}\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

  where

$$\boldsymbol{P} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \dots, \boldsymbol{\alpha}_n)$$

  and

$$\boldsymbol{A}\boldsymbol{\alpha}_i = \lambda_i\boldsymbol{\alpha}_i$$

- The $\boldsymbol{\alpha}_i$ are the *normalised eigenvectors* and the λ_i the *eigenvalues* of $\boldsymbol{A}$.

In PCA we *diagonalise the covariance matrix* and pick components with large $|\boldsymbol{\lambda}|$.

Principal Component Regression

- In *principal component regression* (PCR) we perform a PCA and pick $M < p$ components.
- The we perform a linear least square fit on the $Z_1, Z_2, \ldots, Z_M$ instead of the $X_1, X_2, \ldots, X_P$.
- That is, we fit the model

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \quad i = 1, 2, \ldots, n$$

with

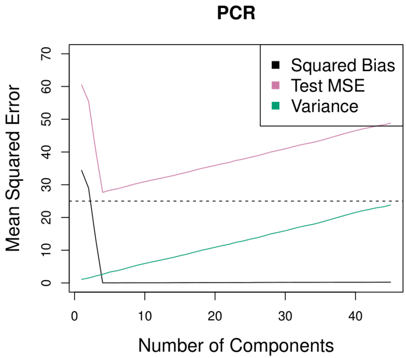
$$\beta_j = \sum_{m=1}^{M} \phi_{jm} \theta_m$$

- The Z_m are linear combinations of *all* X_j.

Note that PCR does *not* do any feature selection.

Principal Component Regression

- We want the model with minimal test MSE.
- In practice, we choose M using cross-validation.
- The more components we include, the more flexible the model becomes.
- That is, the bias goes down and the variance increases.



This is an artificial data set designed for illustration.

Partial Least Squares

- PCA is *unsupervised*, it does not take the responses into account.
- There is no guarantee that the components that best explain the predictors are ideal for predicting the response.
- *Partial least squares* (PLS) tries to address this by taking the response into account.
- The basic idea is still to find $M < p$ linear combinations

$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j$$

in order to reduce the dimensions.

PLS is a supervised reduction method.

Partial Least Squares

- The computation of the PLS direction proceeds as follows.

  1. *First direction, Z_1*: we fit p OLS models

     $$y_{ji} = \beta_{j0} + \beta_{j1} x_{ji} + \epsilon_i, \quad j = 1, 2, \ldots, p$$

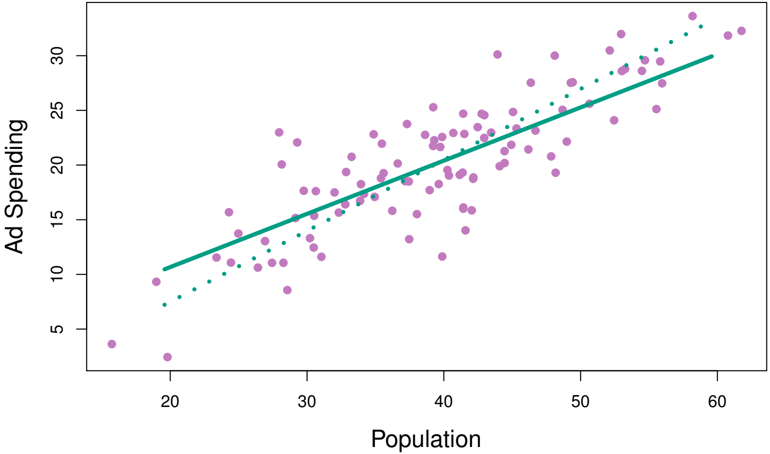
     and then set $\phi_{j1} = \beta_{j1}$:

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

  2. *Rotation*: we rotate all the data in the direction Z_1.
  3. *Iteration*: repeat the procedure to obtain $Z_2, \ldots, Z_M$.

Just like with PCR, we decide on *M* using cross-validation.

PCR versus PLS



In practice, PLS rarely provides significant benefits over PCR.