Statistical Methods: Portfolio 1

By Henry Bourne

Abstract

In this document we will summarize content from the first 4 lectures from the Statistical methods course (at Compass, University of Bristol). These lectures cover content on using statistical methods for decision-making.

1 Introduction to Decision-Making

Computers are often used to answer complex questions, however, these methods are often a "black-box". We would like to have methods which let us conduct **rational decision-making**:

1. Predictions should be precise (no gibberish)
2. They should be data driven
3. They should take cost (of making the wrong decision) into consideration
4. They should take the random nature of data into consideration

In a **regression problem** we want to predict an outcome given some known inputs. We can use the following as an objective function:

Definition 1.1 ***Least Squares (LS):***

$$\min_f \sum_{i \in D_0} (y_i - f(x_i))^2 \tag{1}$$

where f is the function that gives our prediction for x, where x_i is the i-th input, y_i is the i-th (observed) output and $D_0 \subseteq D$ is the training dataset.

By minimizing this objective function wrt. f, we obtain a function f that minimizes the squared difference between its predictions and our observed values of the target variable.

Let $\boldsymbol{w}$ be a vector parameterising f [1], then the LS solution is $\boldsymbol{w}_{LS} := \text{argmin}_{\boldsymbol{w}} \sum_{i \in D_0} (y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2$. We can prove that:

$$\boldsymbol{w}_{LS} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{2}$$

The proof can be found in the appendix (CITE).

Alternatively we can find $\boldsymbol{w}$ in another data-driven way but also whilst taking the randomness of the data into account by using a probabilistic approach. We define a new objective function:

Definition 1.2 ***Maximum likelihood estimation:***

$$\max_{\boldsymbol{w}} \log \mathbb{P}(D|\boldsymbol{w}) \tag{3}$$

*we denote the parameters that maximize this as $\boldsymbol{w}_{ML}$, this is called the **Maximum Likelihood Estimator (MLE)**, we can write,*

$$\boldsymbol{w}_{ML} := \underset{\boldsymbol{w}}{\text{argmax}} \log \mathbb{P}(D|\boldsymbol{w}) \tag{4}$$

*where D is the dataset and $\mathbb{P}(D|\boldsymbol{w})$ is called the **likelihood**.*

Note that we can show $\boldsymbol{w}_{ML} = \boldsymbol{w}_{LS}$ [2]. We can also show that $\sigma^2_{ML} = \frac{1}{n} ||\boldsymbol{y} - f(\boldsymbol{x}; \boldsymbol{w}_{ML})||^2$.

[1] In the case of linear LS we have: $f(\boldsymbol{x}; \boldsymbol{w}) := \langle \boldsymbol{w_1}, \boldsymbol{x} \rangle + w_0$.

[2] This is true in cases where the underlying data-generating process is normal, ie. error terms Gaussian and iid.

1.1 LS with Feature Transform

It is possible to fit non-linear curves to our data using linear LS. All we do is augment our predictor variable, $\boldsymbol{x}$, using a function $\phi(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^{d \times b}$, where:

$$\phi(\boldsymbol{x}) := (\boldsymbol{x}, \boldsymbol{x}^2, ..., \boldsymbol{x}^b)^T \tag{5}$$

With a feature transform we have $\boldsymbol{w}_{LS} = (\phi(\boldsymbol{X})^T \phi(\boldsymbol{X}))^{-1} \phi(\boldsymbol{X})^T \boldsymbol{y}$. A feature transform can let us fit more complex models for our data, however, it can lead to problems...

2 Overfitting and the curse of dimensionality

In previous section we introduced the feature transform. By increasing the value of b we can fit increasingly complex models (models with terms with higher powers of x). We note that when we increase b our model can "bend" to better fit our data-points, however there reaches a point where if it can bend too much then our resultant model is going to be too specific to our training dataset and not generalize well when tested on more data.

2.1 Overfitting

Let's split our data into disjoint sets D_0 and D_1 [3]. We'll use LS as our error function and denote it, E_{LS}, so $E_{LS}(D, \boldsymbol{w}) := \sum_{i \in D} (y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2$. The error function quantifies how well our model fits a dataset. We will use our partitioned datasets such that the **training error** is $E(D_0, \boldsymbol{w}_{LS})$ and the **testing error** is $E(D_1, \boldsymbol{w}_{LS})$.

The testing error tells us how well the model fits data it hasn't seen; how well it **generalizes**. Again if we consider fitting models with different values of b what we will see is that as we initially increase b the testing and training error decrease. However, at a certain point the testing error will begin to increase as the training error continues decreasing. What is happening here is called:

Definition 2.1 ***Overfitting:***
Phenomenon where $f(\boldsymbol{x}; \boldsymbol{w}_{LS})$ fits too well on the training set, D_0, while under-performing on unseen (testing) datasets, D_1.

2.2 Cross-Validation

Consider the problem of how best to test the performance of our model given a dataset, D. We would like to both train our dataset on as much data as possible, but also evaluate its performance on unseen (testing) data such that the score quantifies well the model's ability to generalize. With our scenario earlier where we partitioned the dataset into D_0 (training) and D_1 (testing) datasets we can identify some shortcomings:

1. We have wasted D_1 for validation (testing), what if D_1 contains useful information for fitting a good model?
2. The selection of D_0 and D_1 is random, perhaps D_1 is better for training, or perhaps a different selection altogether would be better.

We now introduce a method which aims to solve these shortcomings:

Definition 2.2 ***K-fold Cross-Validation (CV)***
Split D into disjoint $D_0, ..., D_k$,
for $i = 0, ..., k$:
fit $f^{(i)}$ on all subsets but D_i
for all b compute: $E(D_i, f^{(i)})$
select b which minimizes: $\frac{\sum_i E(D_i, f^{(i)})}{k+1}$
Note: the k picked must be leqn -1 and if $k = n - 1$ then we call this ***leave-one-out-validation****.*

Although CV solves the problems stated earlier it does have its problems of its own:

1. Computational cost is high, as $f^{(i)}$ must be fitted and validated for all splits
2. The effectiveness of CV depends on the assumption that the data is iid. and often this assumption doesn't hold (eg. time series data)

[3] We assume D contains iid. data-points

2.3 Curse of dimensionality

Let's consider carrying out polynomial transform on higher dimensional datasets. When our input $\boldsymbol{x} \in \mathbb{R}^d$ then $\phi(\boldsymbol{x}) \in \mathbb{R}^{d \times b}$ and $\boldsymbol{w} \in \mathbb{R}^{b+1}$. And this is without considering pairwise cross-dimensional polynomials (eg. $x^{(1)}x^{(2)}$ or $x^{(1)}x^{(2)}x^{(3)}$). We can implement this by redesigning ϕ. Let $\phi(\boldsymbol{x}) := (h(x^{(1)}), ..., h(x^{(d)}), \forall_{u<v} x^{(u)}x^{(v)})$, where $h(t) := (t^1, ..., t^b)$, then $\phi(\boldsymbol{x}) \in \mathbb{R}^{d \times (b + \binom{d}{2})}$. We can do the same for cross terms all the way up to d-plets and we know $\binom{d}{1} + \binom{d}{2} + ... + \binom{d}{d} = 2^d$. The output dimension of $\phi(\boldsymbol{x})$ can grow exponentially with dimensionality[4] and the number of observations much at least match the dimensionality, otherwise we cannot obtain $\boldsymbol{w}_{LS}$.

Definition 2.3 ***The Curse of Dimensionality:***
A phenomenon where the number of observations needed to solve a problem grows exponentially with d, it 'forbids' us from solving high-dimensional problems.

Appendices

A Proofs

A.1 Proof of Equation (2)

Let $\boldsymbol{X}$ be,

$$\begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \tag{6}$$

we can write our linear model in matrix form,

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon} \tag{7}$$

note that,

$$\sum_{i \in D_0} (y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2 = ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||^2 \tag{8}$$

we can find the minimum by differentiating wrt. $\boldsymbol{w}$ and finding the solution when the gradient equals zero. However, first we expand our expression,

$$||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) \tag{9}$$

$$= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} \tag{10}$$

we now can find the derivative wrt. $\boldsymbol{w}$,

$$\frac{\partial}{\partial \boldsymbol{w}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||^2 = -2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} \tag{11}$$

now setting this to zero and solving,

$$\Rightarrow \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \tag{12}$$

$$\Rightarrow \boldsymbol{w} = \boldsymbol{X}^{-1}\boldsymbol{X}^{-T}\boldsymbol{X}^T\boldsymbol{y} \tag{13}$$

$$\Rightarrow \boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{14}$$

hence proven.

B Homeworks

B.1 For Section 1

Question B.1 *Prove* $\boldsymbol{w}_{LS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.
Answer is section A.1

[4] We could also include even more complex terms such as $(x^{(u)})^2x^{(v)}$ for example

Question B.2 *Why is the solution of w_{LS} useless if $n<d$?*
If $n < d$ then the columns of the model matrix don't have full rank, we know that for a matrix $A \in \mathbb{R}^{n\times d}$, rank$(\boldsymbol{A}) \leq \min(n, d)$. Therefore, for model matrix X with n, d we have rank$(X) \leq n < d$ and so X is not full rank. Since X is not full rank we have that $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible.

Question B.3 *In what scenarios is the use of the Normal dist. to model $\mathbb{P}(y|\boldsymbol{x}, \boldsymbol{w}, \sigma)$ a bad idea?*
When the errors aren't normally distributed, which could arise when the predictor or target variables are non-normal or when outliers disrupt the model prediction. When this is the case it can cause lots of problems as we often use this assumption of normality when computing confidence intervals and to carry out hypothesis testing for example. So when our assumption of normality is incorrect it can lead to incorrect analyses.

Question B.4 *Prove $\boldsymbol{w}_{LS} = (\phi(\boldsymbol{X}))^{-1}\boldsymbol{y}$.*
We have,

$$\boldsymbol{w}_{LS} = (\phi(\boldsymbol{X})^T\phi(\boldsymbol{X}))^{-1}\phi(\boldsymbol{X})^T\boldsymbol{y} \tag{15}$$
$$= \phi(\boldsymbol{X})^{-1}\phi(\boldsymbol{X})^{-T}\phi(\boldsymbol{X})^T\boldsymbol{y} \tag{16}$$
$$= \phi(\boldsymbol{X})^{-1}\boldsymbol{y} \tag{17}$$

Question B.5 *If we increase b of $\phi(\boldsymbol{x})$ by 2-fold, by how many folds will the computation time of $\boldsymbol{w}_{LS}$ increase?*
Let's consider computing the solution of $\boldsymbol{w}_{LS}$ using the shortened form we found in question B.4. Increasing b by 2-fold will increase the number of rows in the model matrix by 2-fold, finding the inverse of a matrix is $O(n^3)$ (if using Gaussian elimination), so this would lead to a 2^3-fold increase in computation of the inverse. We also will have a 2-fold increase in the number of computations during the matrix multiplication.

B.2 For Section 2

Question B.6 *Why do machine learning algorithms still work on high-dimensional datasets (such as images), despite the curse of dimensionality telling us that the number of observations needed for solving high dimensional problems should grow exponentially with dimensionality?*
This is down to machine learning algorithms being able to learn something about the underlying structure of the data. In learning about the underlying structure it reduces the dimensionality of the problem, as it can use features within the data. For example in convolutional neural networks it's been shown that the network uses sets of features built up hierarchically in order to help it classify images. Instead of having to directly learn what every combination of pixels contained within an image should be classified as it simply learns that certain low level-patterns of pixels are important for classification, it then learns that these low-level patterns put together in different ways create other larger features which are important and so on...

References