Using linear regression for fitting terrain data

Eina B. Jørgensen, Anna Lina P. Sjur and Jan-Adrian H. Kallmyr

September 26, 2019

Abstract

1 Introduction

The use of machine learning for problem solving has risen in popularity as large data sets have become available for analysis. There now exists many different methods in varying complexity for both supervised and unsupervised learning. All of these methods have advantages and drawbacks, as well as many similarities. This means that we can get familiar with some of the central themes in machine learning by studying simple algorithms, such as different linear regression schemes. A notable example is the bias-variance trade-off, where it is observed that the total mean-square-error (MSE) of a model starts off high, decreases until a minimum, and increases again (see Theory and Methods section.) In this report, we will implement three different supervised learning algorithms with increasing complexity, as well as the k-fold resampling technique.

2 Theory and methods

2.1 Linear Regression

We consider a dataset with n cases consisting of the response variable $\boldsymbol{y} = [y_0, y_1, \ldots, y_{n-1}]^T$ and some explanatory variables. The assumption is then made that $\boldsymbol{y}$ can be explained as a functional relationship on the form

$$\boldsymbol{y} = f(\cdot) + \boldsymbol{\epsilon} \tag{1}$$

Here, $\boldsymbol{\epsilon}$ is assumed to be normally distributed with mean 0 and variance σ^2.

A linear Regression model is built on the assumption that $f(\cdot)$ is a linear mapping from the explanatory variables x_{ij} to the response variable y_i, given by computing a weighted sum of the explanatory variables:

$$\tilde{y}_i = \beta_0 x_{i0} + \beta_1 x_{i1} + \ldots + \beta_{p-1} x_{ip-1}$$

Here, $i = 0, 1, \ldots, n-1$, $\tilde{y}_i$ is the prediction, $\{\beta_j\}_{j=0}^{p-1}$ are the regression parameters, while p is the number of explanatory variables.

In vectorized form, this can be written as

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$$

where $\tilde{\boldsymbol{y}} = [\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_{n-1}]^T$ are the predicted values, $\boldsymbol{\beta} = [\beta_0, \beta_1, \ldots . \beta_{n-1}]^T$ are the regression parameters, and $\boldsymbol{X}$ is the so called design matrix given by

$$\boldsymbol{X} = \begin{bmatrix} x_{00} & x_{01} & \ldots & x_{0p-1} \\ x_{10} & x_{11} & \ldots & x_{1p-1} \\ \vdots & \ddots & \ddots & \vdots \\ x_{n0} & \ldots & \ldots & x_{np-1} \end{bmatrix},$$

In this project, we are dealing with a two dimensional problem, where each row of the

design matrix represents the variables of a mth order polynomial, i.e. is on the form $[\{x^i y^j : i + y \leq m\}]$

In order to compute the regression parameters, a cost function $C(\boldsymbol{\beta})$ is introduced. The $\boldsymbol{\beta}$ is then defined as the minimization of the cost function. Different cost functions gives rise to different regression methods. Here we will look at Ordinary Least Squares, Ridge and Lasso regression.

2.2 Ordinary least squares

One form of the cost function is given as

$$\begin{aligned} C(\boldsymbol{\beta}) &= \frac{1}{n}\sum_{n=0}^{n-1}(y_i - \tilde{y}_i)^2 \\ &= \frac{1}{n}\sum_{n=0}^{n-1}(y_i - \boldsymbol{x}_{i*}\boldsymbol{\beta})^2 \\ &= \frac{1}{n}\Big((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\Big) \end{aligned}$$

where $\boldsymbol{x}_i$ is the ith row of the design matrix. By setting $\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$, one can show that the model parameters that minimizes the cost function are given by

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{2}$$

These are the model parameters used in the Ordinary Least Squares (OLS) model.

2.3 Ridge regression

A problem that can occur, especially when dealing with a large design matrix $\boldsymbol{X}$, is that the columns of $\boldsymbol{X}$ are linearly dependent, causing $\boldsymbol{X}^T\boldsymbol{X}$ to be singular. Hoerl and Kennard (1970) proposed a solution to the singularity problem by introducing a tuning parameter λ. This tuning parameter is added to the diagonal elements of $\boldsymbol{X}^T\boldsymbol{X}$, and thus causing the matrix to be non-singular. In Ridge regression, the cost function then takes the form

$$C(\boldsymbol{\beta}) = \frac{1}{n}\sum_{n=0}^{n-1}(y_i - \boldsymbol{x}_{i*}\boldsymbol{\beta})^2 + \lambda\sum_{i=0}^{p-1}\beta_i^2 \tag{3}$$

where $\lambda \in [0, \infty)$. The last term of (3) is often called the penalty term. Minimizing this results in model parameters on the form

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\lambda}\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4}$$

Here, $\boldsymbol{I}$ is the identity matrix of the same size as $\boldsymbol{X}^T\boldsymbol{X}$. The parameter λ then has to be tuned, for example by the use of cross-validation. One can see that by setting $\lambda = 0$, (4) simplifies to the OLS solution in (2).

2.4 Lasso regression

The choice of the penalty term in (3) is somewhat arbitrary, and other penalty functions could be considered. Tibshirani (1996) introduced a penalty function such that the cost function takes the form

$$C(\boldsymbol{\beta}) = \frac{1}{n}\sum_{n=0}^{n-1}(y_i - \boldsymbol{x}_{i*}\boldsymbol{\beta})^2 + \lambda\sum_{i=0}^{p-1}|\beta|$$

2.5 Evaluation scores

In this project, we will use two well known expressions to calculate the error in the predicted values. That is the Mean Square error (MSE)

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{5}$$

and the R^2 score function

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \overline{y})^2}$$

Here, the mean value $\overline{y}$ is given as

$$\overline{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$$

2.6 k-fold cross validation

There are several methods for estimating the skill of a machine learning model. One such method is the k-fold cross-validation procedure, which can be used when working with a limited data sample. The idea is to divide the data sample into k groups or folds, and then retain one of the folds to use as a test set after fitting a model to the remaining data. This is done for all folds. Algorithm 1 outlines the different steps in the procedure.

Shuffle the dataset randomly;
Divide the dataset into k folds;
foreach k **do**
  Take the kth fold out to use as test data set;
  Set the remaining folds as training data set;
  Fit a model to the training set;
  Evaluate the model on the test set;
  Retain the evaluation score and discard the model;
Calculate the mean of the evaluation scores;

Algorithm 1: The k-fold cross-validation algorithm.

This yields a statistical estimate for how well the model will preform on new data. The choice of k will however effect the bias and variance in the estimation of the evaluation scores. It has been shown empirically that $k = 5$ or $k = 10$ gives neither a high bias nor variance (James et al., 2013). In this project, a value of 5 was chosen for k.

2.7 Bootstrap method for resampling

Algorithm 2 gives an overview of the bootstrap method.

Split the data into training and test sets;
Set a number of bootstrap samples n;
Set a sample size N;
foreach n **do**
  Draw a N-sized sample with replacement from the training set;
  Fit a model to the data sample;
  Apply the model on the test set;
Evaluate all the model predictions;
Calculate the mean of the evaluation scores;

Algorithm 2: The bootstrap algorithm.

2.8 The bias-variance trade-off

Taking a look at the MSE again, (5) can be expressed as the expectation value of $(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2$, which can be decomposed as follows. For the full derivation, see 5.

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] =& \frac{1}{n}\sum_{i=0}^{n-1}(f_i - \mathbb{E}\left[\tilde{y}_i\right])^2 \\ &+ \frac{1}{n}\sum_{i=0}^{n-1}(\tilde{y}_i - \mathbb{E}\left[\tilde{y}_i\right])^2 + \sigma^2 \\ =& Bias^2 + Variance + \sigma^2\end{aligned} \tag{6}$$

Here, f_i comes from (1) written out element wise as $y_i = f_i + \epsilon_i$. The first term in (6) is the squared bias, while the second term is the variance of the model. The last term, σ^2, comes from the assumption of a normal distributed noise in (1), and is the so called irreducible error. This error is beyond our control, even if the true value of f_i is known.

When the complexity of the model, i. e. the order of the polynomial, increases, the squared bias tends to decrease. For the variance, the opposite tends to happen (Hastie et al., 2009).

2.9 Franke's function and digital terrain data

In this project, the different regression methods were applied on both constructed and real data. The first was in the form of a sum of weighted exponentials, known as Franke's function:

$$
\begin{aligned}
f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\
&+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right) \\
&+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\
&- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right)
\end{aligned}
$$

In addition to the above terms, a normal distributed noise term with $\mu = 0$ and $\sigma^2 = 1$ was added.

After testing the code on the simpler Franke's function, the same regression methods were used and evaluated on real digital terrain data downloaded from `https://earthexplorer.usgs.gov/`. The data used in this project is of the Oslo fjord region.

2.10 Implementation

Since both OLS and Ridge gives an explicit expression for $\boldsymbol{\beta}$, the model parameters can be calculated directly. In this project, expression (2) and (4) was calculated using the linear algebra functionality in the Python package `numpy`.

Unlike OLS and Ridge regression, there is no general, explicit expression for the model parameters in Lasso regression. Therefore, functionalities from the `scikit-learn` package was used to calculate $\boldsymbol{\beta}$ in the Lasso regression case.

All plots were made using the Python package `matplotlib`.

3 Results

Figure 1

4 Discussion

5 Conclusion

References

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: prediction, inference and data mining. *Springer-Verlag, New York*, 2009.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2013. ISBN 9781461471387.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Appendix

$$
\begin{aligned}
\mathbb{E}[(y-\tilde{y})^2] =& \mathbb{E}[(f+\epsilon-\tilde{y})^2] \\
=& \mathbb{E}[(f+\epsilon-\tilde{y}+\mathbb{E}[\tilde{y}]-\mathbb{E}[\tilde{y}])^2] \\
=& \mathbb{E}[(f-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[\epsilon^2]+\mathbb{E}[(\mathbb{E}[\tilde{y}]-\tilde{y})^2]+2\mathbb{E}[(f-\mathbb{E}[\tilde{y}])\epsilon] \\
& +2\mathbb{E}[\epsilon(\mathbb{E}[\tilde{y}]-\tilde{y})]+2\mathbb{E}[(\mathbb{E}[\tilde{y}]-\tilde{y})(f-\mathbb{E}[\tilde{y}])] \\
=& (f-\mathbb{E}[\tilde{y}])^2+\mathbb{E}[\epsilon^2]+\mathbb{E}[(\mathbb{E}[\tilde{y}]-f)^2]+2(f-\mathbb{E}[\tilde{y}])\mathbb{E}[\epsilon] \\
& +2\mathbb{E}[\epsilon]\mathbb{E}[\mathbb{E}[\tilde{y}]-\tilde{y}]+2\mathbb{E}[\mathbb{E}[\tilde{y}]-\tilde{y}](f-\mathbb{E}[\tilde{y}]) \\
=& (f-\mathbb{E}[\tilde{y}])^2+\mathbb{E}[\epsilon^2]+\mathbb{E}[(\mathbb{E}[\tilde{y}]-\tilde{y})^2] \\
=& Bias[\tilde{y}]^2+Var[y]+Var[\tilde{y}] \\
=& Bias[\tilde{y}]^2+\sigma^2+Var[\tilde{y}]
\end{aligned}
\tag{A.1}
$$