# Machine Learning and Terrain data

Mathias M. Vege

October 8, 2018

**Abstract**

None

## Contents



## 1 Introduction

A now-fully emergent field of data analysis, is that of regression and resampling. Included as a subset of the field of machine learning, regression is widely used as tool of prediction. Together with resampling we are offered ways of estimating the error of our models.

One common way to investigate the efficacy of a model is to use the Franke function[3].

## 2 Methods and theory

Let us start of by summing up notational conventions. We will denote a *case* or *sample* by $x_i$, where $i = 0, \cdots, n$. This will have a corresponding *response* or *outcome* $y_i = f(x_i)$. We typically want to estimate $f$ by some approximate function $\tilde{f}$. As we shall see, one method is by performing linear regression.

### 2.1 Linear regression

As mentioned, linear regression is a method of performing a fit of parameters $x_i$ to a given data set $y_i$. From this we wish to minimize the error between the *response* or *real* data points $y_i$, and some approximation $\tilde{y}_i = \tilde{f}(x_i)$. This defines our loss function, which we will seek to minimize. In linear regression this will be by building an *design matrix* built from $p$ *explanatory variables* or *predictors* $\boldsymbol{x_i} = x_i^0 + x_i^1 + \cdots + x_i^p$. The explanatory variables is simply the degrees of the polynomial we seek to approximate $f$ by. So far, we have not mentioned the variables which will be tuned to the model, the *regression parameters* $\beta = (\beta_1, \ldots, \beta_p)^T$. The keen-eyed reader will now notice that the predictors and regression parameters all are of equal number, and may have guessed that they form our linear approximation - this is true:

$$\tilde{y}_i = \beta_0 x_{i0} + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$$

From this, we get a neat matrix equation,

$$\boldsymbol{y} = \beta \boldsymbol{X} + \boldsymbol{\epsilon} \tag{1}$$

We see that our goal has immediately become to minimize $\beta$. The definition of the error vector $\boldsymbol{\epsilon}$ will be given later.

The simplest case of such a linear regression minimization problem is just a simple line[5, ch. 3.1, p. 61]. However, this can be generalized greatly and we end up with the standard go-to method called *Ordinary Least Squares*.

#### 2.1.1 Ordinary Least Squares regression(OLS)

To begin our journey towards a solution of OLS, we start by defining the *loss function* for OLS.

$$\min_{\beta \in \mathbb{R}^p} ||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 = \min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \left(\boldsymbol{x}_i^T \beta - y_i\right)^2 \tag{2}$$

We see that the goal is to minimize the $L_2$ norm between the response $y_i$ and the fitted predictors $\boldsymbol{x}_i^T \beta$ [6, ch. 4, p. 21].

To find a solution to the loss function, we can start by observing the dimensions of equation (1),

$$\underbrace{\boldsymbol{y}}_{\mathbb{R}^n} = \underbrace{\boldsymbol{X}}_{\mathbb{R}^{n \times p}} \underbrace{\beta}_{\mathbb{R}^p} \tag{3}$$

We then see that in order to free $\beta$, we must multiply by the transposed of $\boldsymbol{X}$ such that we can take the inverse and solve for $\beta$.

But, before we can do that we need to define what the optimal $\beta$ vector is. We start by defining a function $Q(\beta)$ which gives us the squared error of the spread,

$$\begin{aligned} Q(\beta) &= \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2 \\ &= (\boldsymbol{y} - \tilde{\boldsymbol{y}})^T (\boldsymbol{y} - \tilde{\boldsymbol{y}}) \end{aligned}$$

which is the matrix equation

$$Q(\beta) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta) \tag{4}$$

This equation is called the cost function[6, p. 12]. Before moving on, we should not that we can generalize it to

$$Q(\boldsymbol{y}, g(\boldsymbol{x})) = \sum_i (y_i - g(\boldsymbol{x})) \tag{5}$$

where $g(\boldsymbol{x})$ is some function for the predictor. In OLS, this is $\beta_i \boldsymbol{x}_i$.

The minimum of the cost function (4) can be found by differentiating by $\beta$ and setting the result equal to zero.

$$\begin{aligned} \frac{\partial Q(\beta)}{\partial \beta_j} &= \frac{\partial}{\partial \beta_j} \sum_{i=0}^{n-1} (y_i - \beta_0 x_{i0} - \beta_1 x_{i1} - \dots \beta_{p-1} x_{i,p-1})^2 \\ &= -2 \sum_{i=0}^{n-1} x_{ij} (y_i - \beta_0 x_{i0} - \beta_1 x_{i1} - \dots \beta_{p-1} x_{i,p-1}) \end{aligned}$$

where the element $x_{ij}$ is picked out by the differentiation w.r.t. $\beta_j$. Performing all of the derivatives leaves us with the matrix equation,

$$\frac{\partial Q(\beta)}{\partial \beta} = X^T (\boldsymbol{y} - \boldsymbol{X}\beta) \tag{6}$$

which we require to be zero in order to find an optimal $\beta$. From this the *residues* or *error vector* $\boldsymbol{\epsilon}$ is defined,

$$\boldsymbol{\epsilon} = \boldsymbol{y} - \tilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}\beta \tag{7}$$

The residues is the error between the *approximation* and the *real* data, and is not to be confused with the inherent error in the data. The solution should now be obvious. We need to solve equation (6) for $Q(\beta) = 0$,

$$\boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\beta) = 0.$$

As alluded to in (3), we now need to multiply by the inverse of $\boldsymbol{X}^T \boldsymbol{X}$, which yields after splitting the parenthesis

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

And thus, the $\beta$-values is simply

$$\beta = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \left(\boldsymbol{X}^T \boldsymbol{X}\right) \boldsymbol{y} \tag{8}$$

This solution does not account for errors in the measurements or responses. Say there is an $\sigma_i$ associated with each measured $y_i$, we will have to use the $\chi^2$ function as a our function to minimize[4, see notes on regression]. We will have to rescale our solution by $\boldsymbol{X} \to \boldsymbol{A} = \boldsymbol{X}/\boldsymbol{\Sigma}$ where $\Sigma$ is a diagonal matrix of every uncertainty $\sigma_i$.

### 2.1.2 Ridge regression

The idea of Ridge regression is to add a small term $\lambda$ to constraint,

$$\beta(\lambda) = \min_{\beta \in \mathbb{R}^p} (||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda ||\beta||_2^2) \tag{9}$$

This amounts to finding a solution in parameters space $\beta$ which tangents a circle defined by the $\mathrm{L}^2$ norm of the constraint (11).

As shown in [5, 6], the solution to the Ridge regression, is to replace the inverse in the OLS solution for $\beta$ (8), by

$$\boldsymbol{X}^T \boldsymbol{X} \to \boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I} \tag{10}$$

with $\boldsymbol{I}$ as the identity matrix.

The idea of adding an $\lambda \boldsymbol{I}$ can aid in resolving issues surrounding singularities in the inverse which can be troublesome when dealing with a large parameter space $\beta$[4, see notes on regression, p. 17].

### 2.1.3 Lasso regression

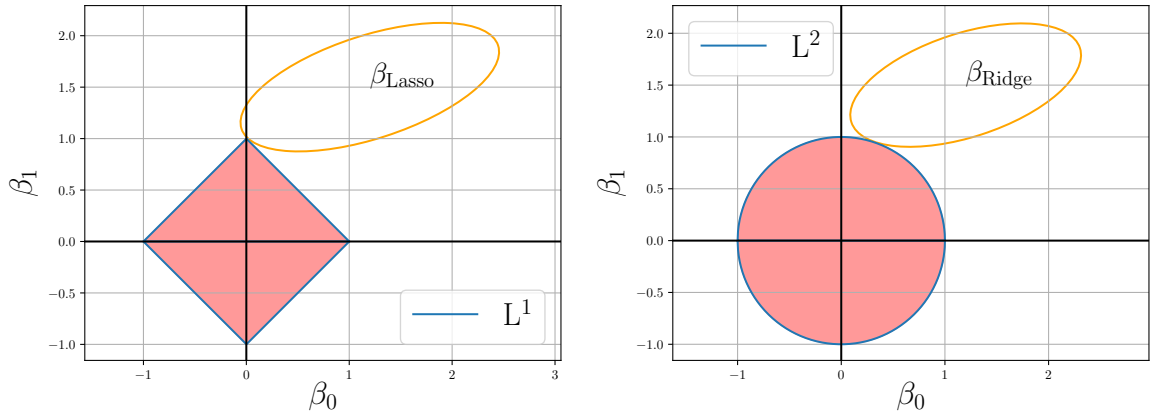
The idea behind the Lasso regression is similar to that of Ridge regression. However, it comes with a few of its own benefits, such that it tends to zero out $\beta$ coefficients. To see this, we can take a look at the constraint function,

$$\beta(\lambda) = \min_{\beta \in \mathbb{R}^p} (||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda ||\beta||_1^2) \tag{11}$$

where we are taking the $\mathrm{L}^1$ norm(popularly called Taxicab or Manhattan norm) in the $\lambda$ term, $\lambda ||\beta||_1^2$. Since the $\mathrm{L}^1$ norm is defined as

$$||\boldsymbol{a}||_1 = |a_0| + \cdots + |a_{n-1}| = \sum_{i=0}^{n-1} |a_i|$$

To visualize and better understand the difference between Ridge and Lasso regression, we can take a look at figure 1. Here we see the difference to be that the Ridge will tend to get more $\beta_i$ values as zero, due to the $\mathrm{L}^1$ norm having a greater chance of *"hitting"* the corners of the diamond in figure 1a.




(a) $L^1$ norm for Lasso.

(b) $L^2$ norm for Ridge.

Figure 1: In the figure 1b we see how the search space for the Ridge regression, where we define a success when the $\beta$ parameter hits the circle. The search space for Lasso is viewed on the figure to the right 1a.

### 2.2 The Bias-Variance decomposition

It is possible to define variance of the predicted values $\tilde{y}$. If we take the estimator of the general cost function for linear regression (5),

$$E[Q(g(\boldsymbol{x}))] = E\left[\sum_i (y_i - g(\boldsymbol{x_i}))^2\right]$$

it can be shown that we can decompose this into the the variance of $\boldsymbol{y}$, bias and noise[6, 4].

$$\text{MSE} = \text{Bias}^2 + \text{Var} + \text{Noise} \tag{12}$$

The bias is defined as

$$\text{Bias}^2 = \sum_i (\tilde{y}_i - E[\boldsymbol{y}])^2, \tag{13}$$

which is measure of how much the prediction deviates from the real data.

A note on the Noise is needed here before we move on. The noise is defined as the the variance in the real data, Noise = $\text{Var}(\epsilon_{\text{noise}})$. Since we cannot know the noise beforehand, this will be gobbled up into the bias, as that is a measure of the deviation in real data and prediction. Since the noise is irreducible in the sense we cannot remove the initial noise in the data, the MSE will never lie below $\text{Var}(\epsilon)$[5, ch. 2, p. 34].

The variance in (12) is defined as the variance in in the prediction, $\tilde{\boldsymbol{y}}$.

The MSE in (12) is the Mean Square Error is defined as,

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2, \tag{14}$$

and is simply the averaged squared errors.

### 2.3 $R^2$ score

A assessment of the fit quality is given by the $R^2$ score,

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{\boldsymbol{y}})^2} \tag{15}$$

with $\bar{\boldsymbol{y}}$ defined as

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

### 2.4 Bootstrapping

Bootstrapping is a resampling technique particularly useful for small datasets or observables with difficult-to-asses statistics[2]. The bootstrap technique is inspired by the Jackknife method[1], but differs since we randomly selects a $n$ new samples with replacement of the original dataset $\{y_i\}$ a total of $N_{\text{bs}}$ times. We use the $N_{\text{bs}}$ samples to find the MSE, variance and bias.

A basic for bootstrap algorithm can be set up as following,

**Algorithm 1** Bootstrap

1: Split the dataset $\boldsymbol{X}_{\mathcal{L}} = \{y_i, \boldsymbol{x}_i\}$ into a training set and a test set(or holdout set), $\boldsymbol{X}_{\mathcal{L},\text{train}}$ and $\boldsymbol{X}_{\mathcal{L},\text{test}}$.
2: **for** $N_{\text{bs}}$ times **do**
3: Build a new data set $\boldsymbol{X}_{\mathcal{L},\text{bs}} = \{y_{i,\text{bs}}, \boldsymbol{x}_{i,\text{bs}}\}$ with $n$ randomly drawn with replacement values, from the training set $\boldsymbol{X}_{\mathcal{L},\text{train}}$.
4: Fit the bootstrapped dataset to a given model.
5: Evaluate the model on on the test set $\boldsymbol{X}_{\mathcal{L},\text{test}}$.
6: Store relevant results.
7: **end for**
8: Perform final statistics on either the collection of bootstrapped datasets and/or take the averages on the stored results. Depending on the type of variable you are sampling, you might need to use latter one.

### 2.5 $k$-fold Cross Validation

The Cross Validation(CV) is a resampling technique which is similar in thought to the Jackknife method, but differs at a few critical places. Details on the $k$-fold CV method can be found in the book An Introduction to Statistical learning by **(author?)** [5, ch. 5.1], and can be summarized as following,

**Algorithm 2** $k$-fold Cross Validation

1: Shuffle the dataset $\boldsymbol{X}_{\mathcal{L}} = \{y_i, \boldsymbol{x}_i\}$.
2: Split the data set into training and test data, $\boldsymbol{X}_{\mathcal{L},\text{train}}$ and $\boldsymbol{X}_{\mathcal{L},\text{test}}$.
3: Split the training data into $k$ *folds*. $\boldsymbol{X}_{\mathcal{L},\text{train}} \to \{\boldsymbol{X}_{\mathcal{L},i_k}\}$
4: **for** each $i_k$ in the $k$ folds **do**
5: Select the set $\tilde{\boldsymbol{X}}_{\mathcal{L}} = \boldsymbol{X}_{\mathcal{L}}/\{\boldsymbol{X}_{\mathcal{L},i_k}\}$(excluding the $k$'th set).
6: Fit the $\tilde{\boldsymbol{X}}_{\mathcal{L}}$ to the model which is being used.
7: Evaluate the $\tilde{\boldsymbol{X}}_{\mathcal{L}}$ on the test set $\boldsymbol{X}_{\mathcal{L},\text{test}}$, and perform necessary statistics.
8: **end for**
9: Summarize the evaluations and statistics gained from the $k$ folds.

# 3 Implementation

The full implementation can be found at [8].
[7]

# 4 Results

# 5 Discussion and conclusion

# 6 Appendix

## 6.1 The Franke Function

As given in [3],

$$F(x,y) = 0.75\exp\left(-\left(0.25\left(9x-2\right)^2\right) - 0.25\left(\left(9y-2\right)^2\right)\right) \tag{16}$$

$$= 0.75\exp\left(-\frac{1}{49}\left(\left(9x+1\right)^2\right) - 0.1\left(9y+1\right)\right) \tag{17}$$

$$= 0.5\exp\left(-\left(9x-7\right)^2/4.0 - 0.25\left(\left(9y-3\right)^2\right)\right) \tag{18}$$

$$= -0.2\exp\left(-\left(9x-4\right)^2 - \left(9y-7\right)^2\right) \tag{19}$$

# References


[1] B. Efron and C. Stein. The jackknife estimate of variance. *Ann. Statist.*, 9(3):586–596, 05 1981.

[2] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[3] Richard Franke. A critical comparison of some methods for interpolation of scattered data. Technical report, Monterey, California: Naval Postgraduate School., 1979.

[4] Morten Hjort-Jensen. Data analysis and machine learning: Linear regression and more advanced regression analysis lecture notes. https://compphysics.github.io/MachineLearning/doc/web/course.html, October 2018.

[5] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[6] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *ArXiv e-prints*, March 2018.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Mathias Vege. Fysstk4155: Project 1 github repository. https://github.com/hmvege/FYSSTK4155-Project1, October 2018.