



An introduction to regularisation and hyper-parameters

Henry W J Reeve

henry.reeve@bristol.ac.uk

Statistical Computing & Empirical Methods (EMATM0061)

MSc in Data Science, Teaching block 1, 2021.

What will we cover today?

- We will introduce the important concept of regularization.
- Regularization will allow us to do better on unseen data, whilst performing worse on train data.
- This leads to hyper-parameters which control the regularization level.
- To tune our hyper-parameters we will need to investigate the train-validate-test split.
- We will see how regularization applies to both regression and classification problems.

Ordinary least squares

The OLS minimizes the empirical error $\hat{\mathcal{R}}_{\text{MSE}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2$

over the class of linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ by $\phi_{w,w^0}(x) = w x^\top + w^0$

The OLS is $\hat{w} = \Sigma_{Y,X} (\Sigma_{X,X})^{-1}$ and $\hat{w}^0 = \bar{Y} - \hat{w} \bar{X}^\top$.



1. Maximum likelihood estimators of w and w^0 .
2. Unbiased estimators with $\mathbb{E}[\hat{w}] = w$ and $\mathbb{E}[\hat{w}^0] = w^0$
3. Minimum variance over all unbiased estimators (Gauss-Markov).

Ordinary least squares

The OLS minimizes the empirical error $\hat{\mathcal{R}}_{\text{MSE}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2$

over the class of linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ by $\phi_{w,w^0}(x) = w x^\top + w^0$

The OLS is $\hat{w} = \Sigma_{Y,X} (\Sigma_{X,X})^{-1}$ and $\hat{w}^0 = \bar{Y} - \hat{w} \bar{X}^\top$.

The OLS suffers from major limitations especially when the dimension d is large!



- 4. The variance can still be extremely large.
- 5. There are often numerical instabilities when $\Sigma_{X,X}$ has small determinant.
- 6. Sometimes $\Sigma_{X,X}$ is non-invertible so \hat{w} is not defined.

High dimensional regression problems

In many applications our goal is to learn a regression model $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ with a high-dimensional feature space \mathbb{R}^d .

Computer vision



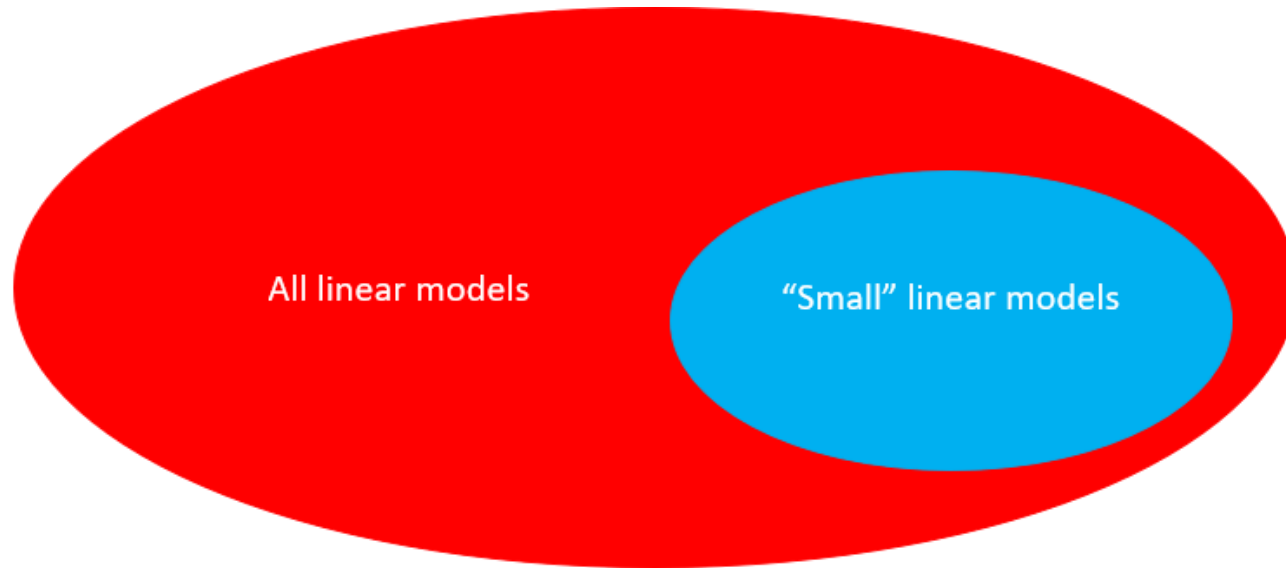
Genomic data



Ridge regression

The source of the instability is a search over a large space of all linear models.

Idea: To reduce the instability of our estimate we limit our search space to “smaller” models.



But how do we quantify the size of a model?

Ridge regression

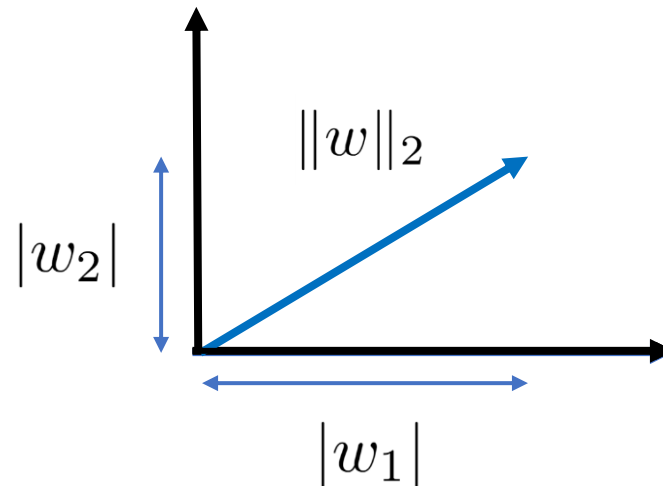
Idea: To reduce the instability of our estimate we limit our search space to “smaller” models.

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ and $\phi_{w,w^0}(x) = w x^\top + w^0$.

How can we measure the size of weight vectors $w = (w_1, \dots, w_d) \in \mathbb{R}^d$?

The Euclidean norm:

$$\|w\|_2 = \sqrt{(w_1)^2 + \dots + (w_d)^2}$$



Ridge regression

Idea: To reduce the instability of our estimate we limit our search space to “smaller” models.

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ and $\phi_{w,w^0}(x) = w x^\top + w^0$.

The OLS method minimizes the empirical error $\hat{\mathcal{R}}_{\text{MSE}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2$.

The **ridge regression** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_\lambda(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$$

Mean squared error on training data

Regularization term

Ridge regression

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ and $\phi_{w,w^0}(x) = w x^\top + w^0$.

The **ridge regression** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_\lambda(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$$

Mean squared error on training data

Regularization term

The quantity $\lambda \geq 0$ is referred to as a hyperparameter.

The higher λ is the higher the level of regularization.

Ridge regression

The **ridge regression** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^{\top} + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$$

The solution is $\hat{w}_{\lambda} = \Sigma_{Y,X} (\Sigma_{X,X} + \lambda \cdot \mathbf{I}_d)^{-1}$ and $\hat{w}_{\lambda}^0 = \bar{Y} - \hat{w}_{\lambda} \bar{X}^{\top}$

where

$$\bar{X} := \frac{1}{n} \sum_{i=1}^n X_i, \quad \Sigma_{X,X} := \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^{\top} (X_i - \bar{X})$$
$$\bar{Y} := \frac{1}{n} \sum_{i=1}^n Y_i, \quad \Sigma_{Y,X} := \frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y}) (X_i - \bar{X})$$

and \mathbf{I}_d is the $d \times d$ identity matrix.

Ridge regression

The **ridge regression** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^{\top} + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$$

The solution is $\hat{w}_{\lambda} = \Sigma_{Y,X} (\Sigma_{X,X} + \lambda \cdot \mathbf{I}_d)^{-1}$ and $\hat{w}_{\lambda}^0 = \bar{Y} - \hat{w}_{\lambda} \bar{X}^{\top}$

\mathbf{I}_d is the $d \times d$ identity matrix

$$\mathbf{I}_d = \left(\begin{array}{ccccc} & \overbrace{\hspace{1.5cm}}^d & & & \\ 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & \cdots & 0 & 1 \end{array} \right) \left. \vphantom{\begin{array}{ccccc} 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & \cdots & 0 & 1 \end{array}} \right\} d$$

Ridge regression

The **ridge regression** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^{\top} + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$$

The solution is $\hat{w}_{\lambda} = \Sigma_{Y,X} (\Sigma_{X,X} + \lambda \cdot \mathbf{I}_d)^{-1}$ and $\hat{w}_{\lambda}^0 = \bar{Y} - \hat{w}_{\lambda} \bar{X}^{\top}$

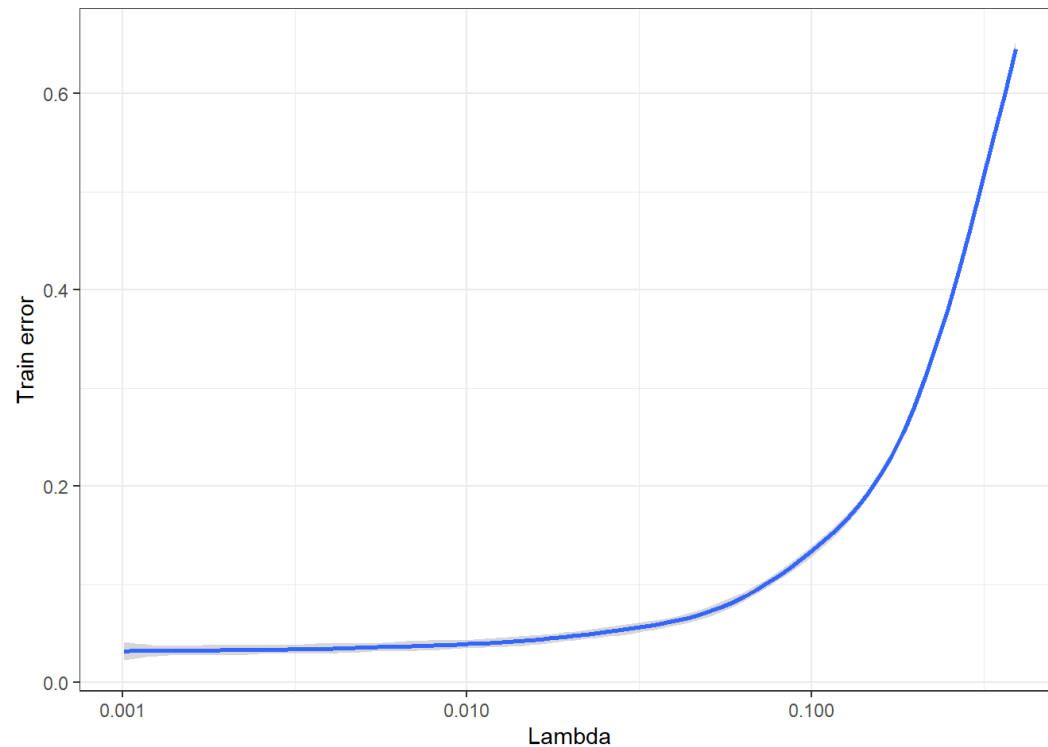
Whenever $\lambda > 0$ the matrix $\Sigma_{X,X} + \lambda \cdot \mathbf{I}_d$ is invertible.

This avoids the numerical issues encountered by the OLS when $\Sigma_{X,X}$ isn't invertible.

As we increase $\lambda > 0$ the Euclidean norm of the weights $\|\hat{w}_{\lambda}\|_2$ decreases.

The train error of ridge regression

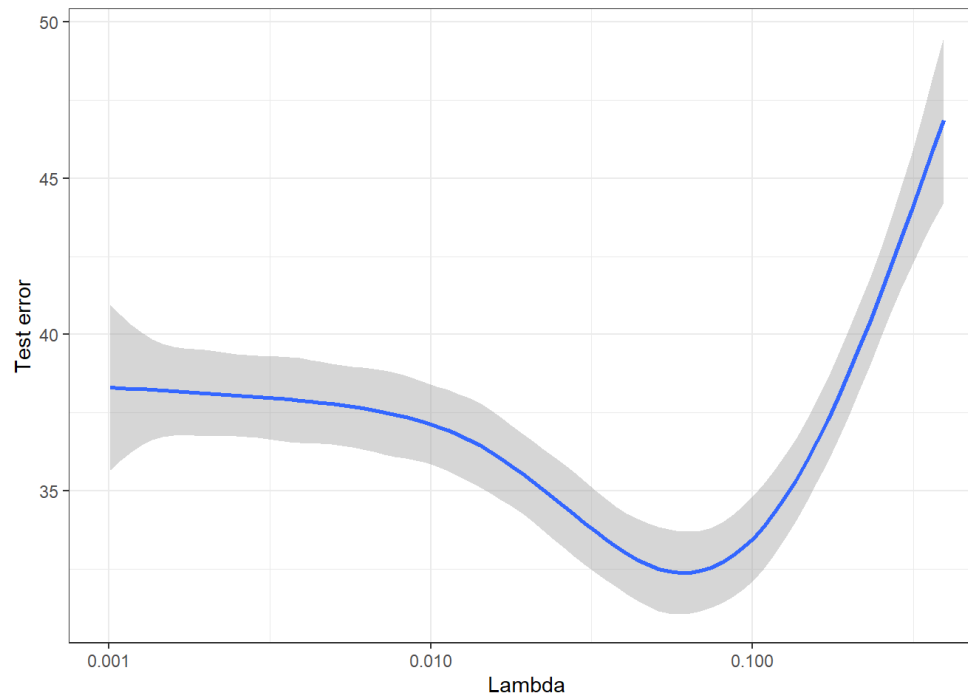
Ridge regression optimizes a regularized loss: $\hat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^{\top} + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2$.



As the hyper parameter λ rises the mean squared error on the training data rises.

The test error of ridge regression

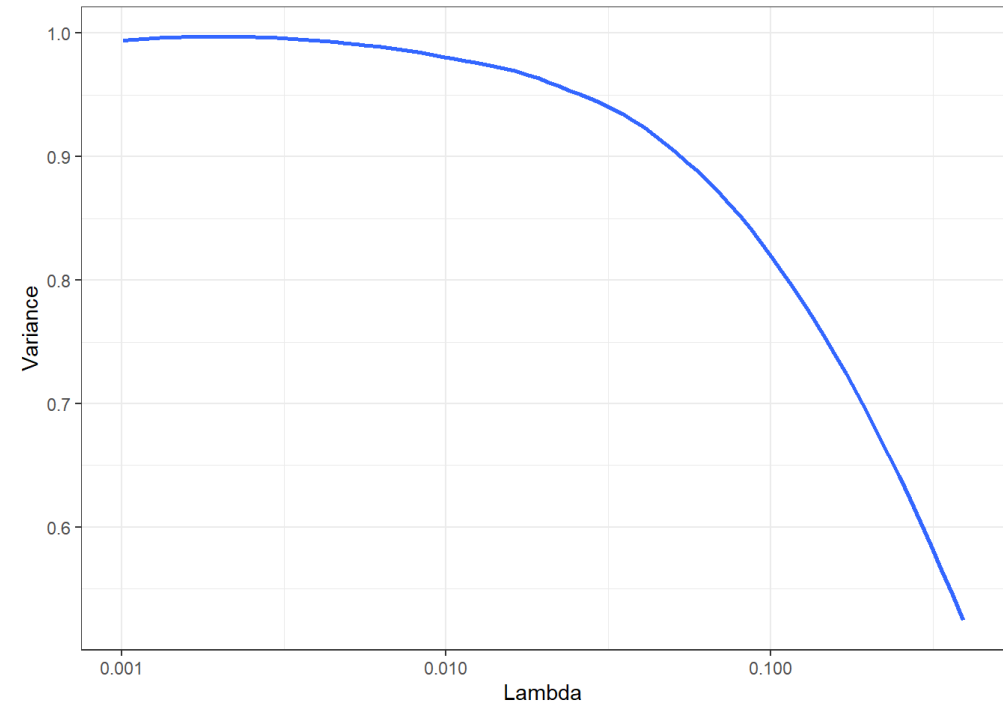
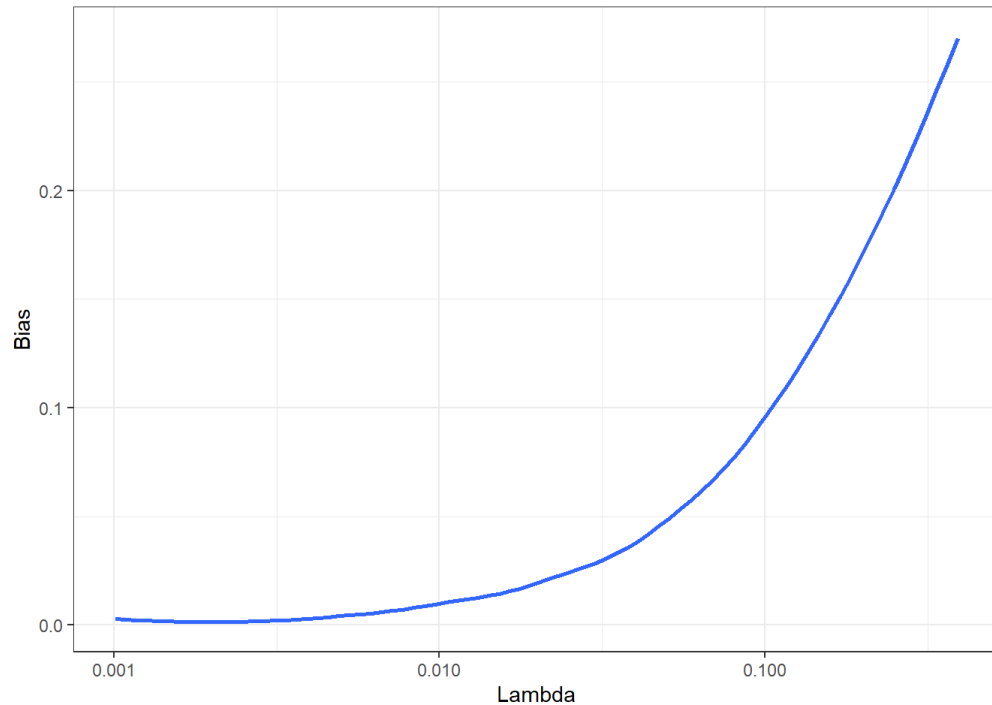
Ridge regression optimizes a regularized loss: $\hat{\mathcal{R}}_\lambda(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2$.



As the hyper parameter λ rises the mean squared error on the test data falls before rising again!

The bias-variance perspective on ridge regression

Ridge regression optimizes a regularized loss: $\hat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^{\top} + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$



The overall error in our parameter estimates is a combination of these two factors.

Ridge regression

Ridge regression optimizes a regularized loss: $\hat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^{\top} + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$

As we increase the hyper-parameter λ we observe:



- An increase in the mean squared error on the training data.
- An increase in statistical bias.

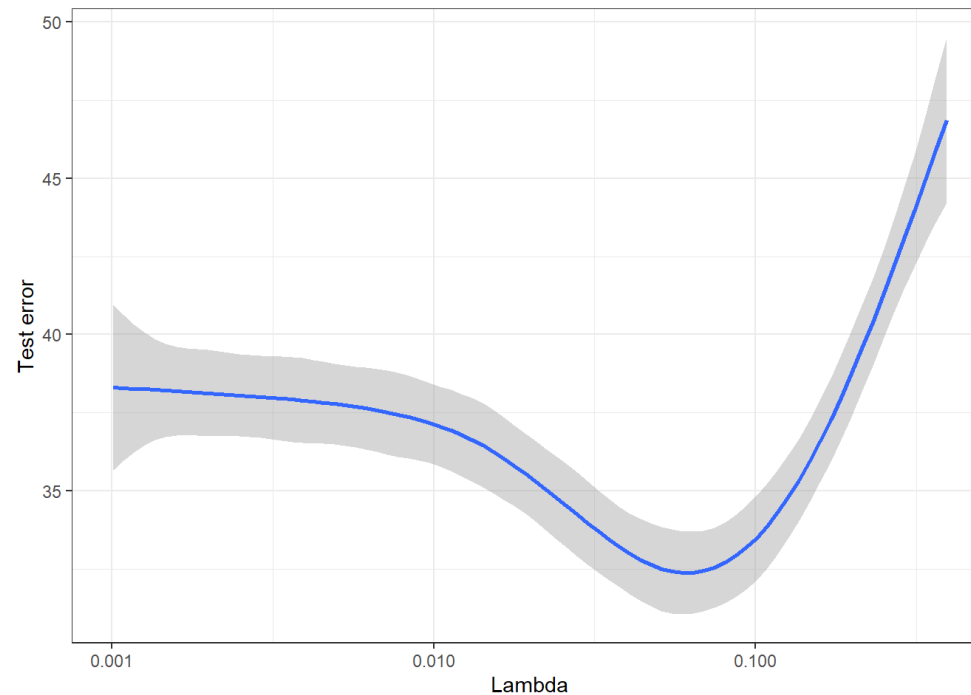


- A reduction in the norm of the weights $\|\hat{w}_{\lambda}\|_2$.
- A reduction in the statistical variance of the estimate.

Regularisation can improve performance on test data by damaging performance on training data.

Ridge regression

Ridge regression optimizes a regularized loss: $\hat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^{\top} + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$



How should we select the hyper-parameter λ ?

Now take a break!



Statistical Computing & Empirical Methods

Choosing hyperparameters

How should we select the hyper-parameter λ ?

1. Select λ to minimize error on the training data?

This is how we select the weights.

Problem: This will always lead to $\lambda = 0$.

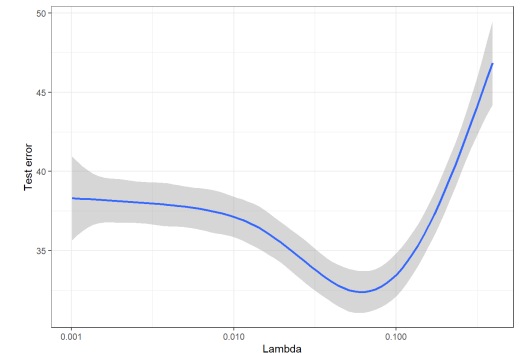
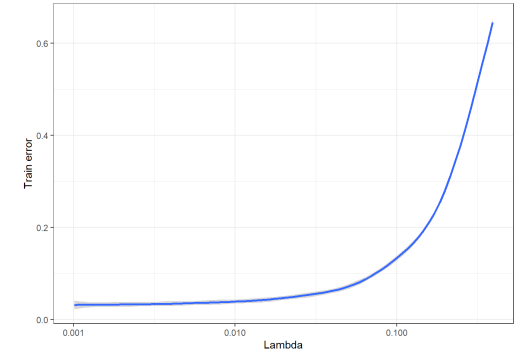
2. Select λ to minimize the error on the test data?

This will lead to reasonable choices of the λ .

Problem: We cannot use the test data to select any model parameters.

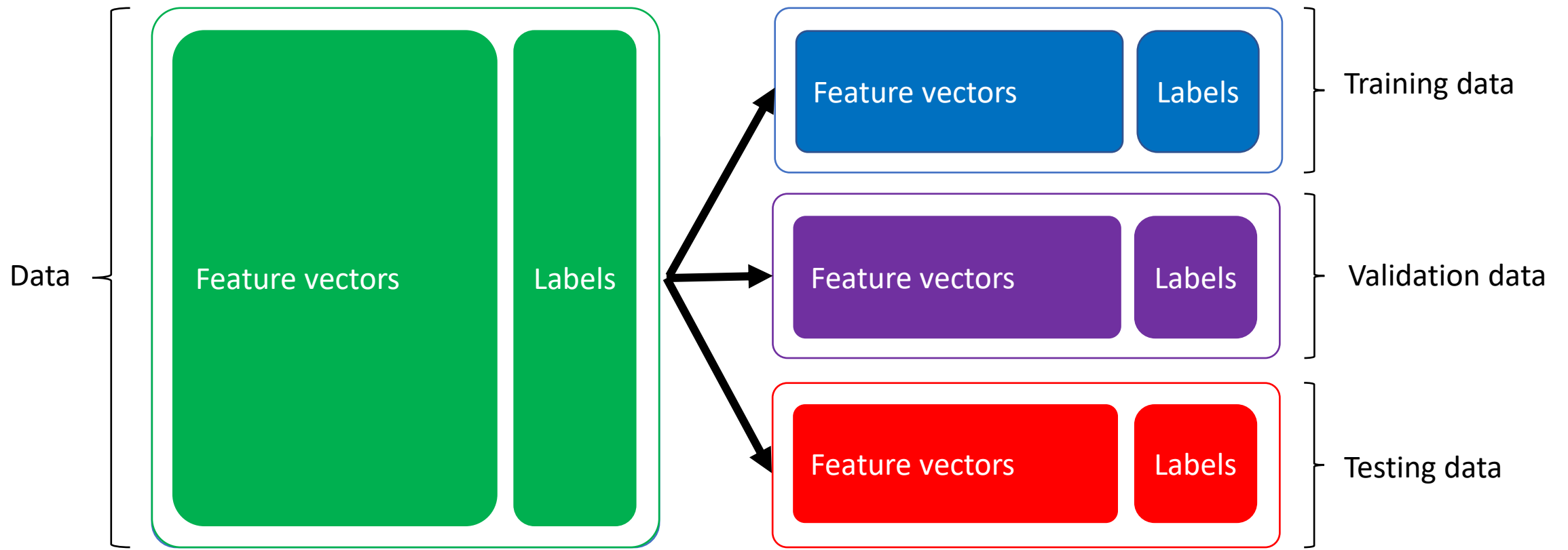
Doing so would prevent us from assessing performance on unseen data.

We require an additional subset of data: The validation data set.



The train-validation-test split

How should we select the hyper-parameter λ ?



The train-validation-test split

We begin by selecting a range of possible hyper-parameters $\lambda_1, \dots, \lambda_Q$.

For each $q = 1, \dots, Q$,

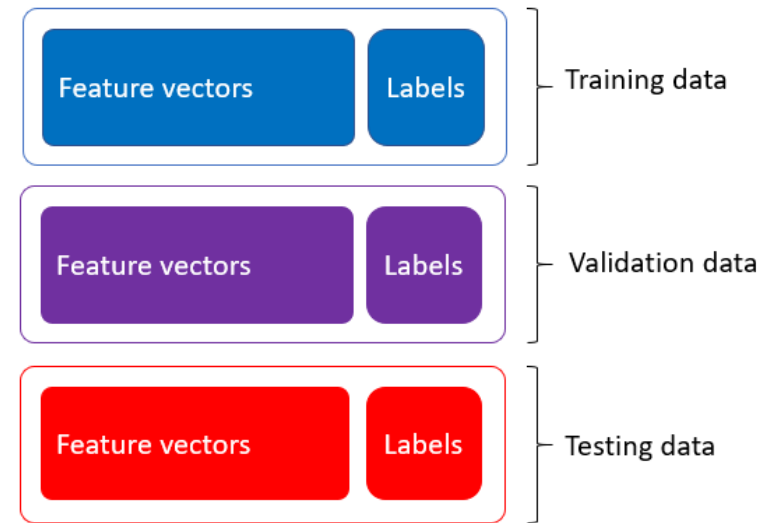
Train a model $\hat{\phi}_{\lambda_q}$ based on the **training data**
and the hyperparameter λ_q .

Compute the average error on the **validation data**.

Choose a hyper-parameter $\hat{\lambda} \in \{\lambda_1, \dots, \lambda_Q\}$
to minimize the average error on the validation data.

Use the **test data** to estimate the performance of the model $\hat{\phi}_{\hat{\lambda}}$ on unseen data.

Return the selected model $\hat{\phi}_{\hat{\lambda}}$.



Example: Phenotype regression with genomic data

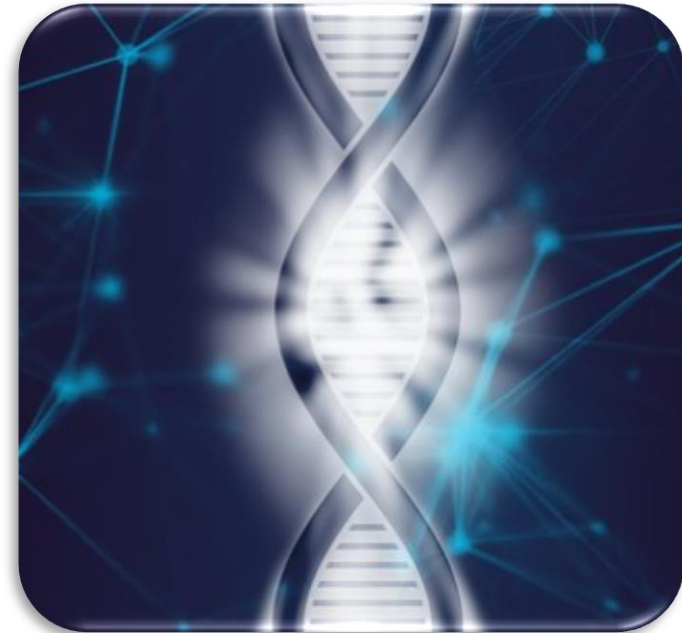
Let's suppose we want to learn a regression model $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$.

Feature vectors X which takes values in \mathbb{R}^d
and consists of d genotypes.

Continuous label Y which takes values in \mathbb{R}
and corresponds to a phenotype.

```
genomicSim%>%dim()
```

```
## [1] 1000 501
```



Train-validation-test split

We begin by carrying out a train-validation-test split of the data.

Set the size of the train, validate and test data sets.

```
num_total<-genomicSim%>%nrow()
num_train<-floor(0.5*num_total)
num_validate<-floor(0.25*num_total)
num_test<-num_total-num_train-num_validate
```

Randomly sample indices for the test, validation and train data.

```
set.seed(123) # set random seed for reproducibility
test_inds<-sample(seq(num_total),num_test) # test indicies
validate_inds<-sample(setdiff(seq(num_total),test_inds),num_validate) # validate inds
train_inds<-setdiff(seq(num_total),union(validate_inds,test_inds)) # train inds
```

Train-validation-test split

Extract the train, validation and test data sets based on their indices.

```
genom_train<-genomicSim%>%filter(row_number() %in% train_inds) # train data  
genom_validate<-genomicSim%>%filter(row_number() %in% validate_inds) # train data  
genom_test<-genomicSim%>%filter(row_number() %in% test_inds) # test data
```

Split the train, validation and test data sets into feature vectors and labels.

```
genom_train_x<-genom_train%>%select(-pheno)%>%as.matrix() # train features  
genom_train_y<-genom_train%>%pull(pheno) # train labels  
  
genom_validate_x<-genom_validate%>%select(-pheno)%>%as.matrix() # validate features  
genom_validate_y<-genom_validate%>%pull(pheno) # validate labels  
  
genom_test_x<-genom_test%>%select(-pheno)%>%as.matrix() # train features  
genom_test_y<-genom_test%>%pull(pheno) # train labels
```


Ridge regression performance on validation data

Construct a function to train a ridge regression model & check validation performance.

```
library(glmnet)

compute_train_validate_error_ridge<-function(train_x,train_y,validate_x,validate_y,lambda){

  glmRidge = glmnet(x=train_x, y=train_y, alpha = 0, lambda=lambda) # train model

  train_y_est<-predict(glmRidge,newx=train_x) # train predictions
  train_error = mean((train_y-train_y_est)^2) # train error

  validate_y_est<-predict(glmRidge,newx=validate_x) # validation predictions
  validate_error = mean((validate_y-validate_y_est)^2) # validation error

  return(list(train_error=train_error,validate_error=validate_error))

}
```

Ridge regression performance on validation data

Choose a set of hyper-parameters to consider.

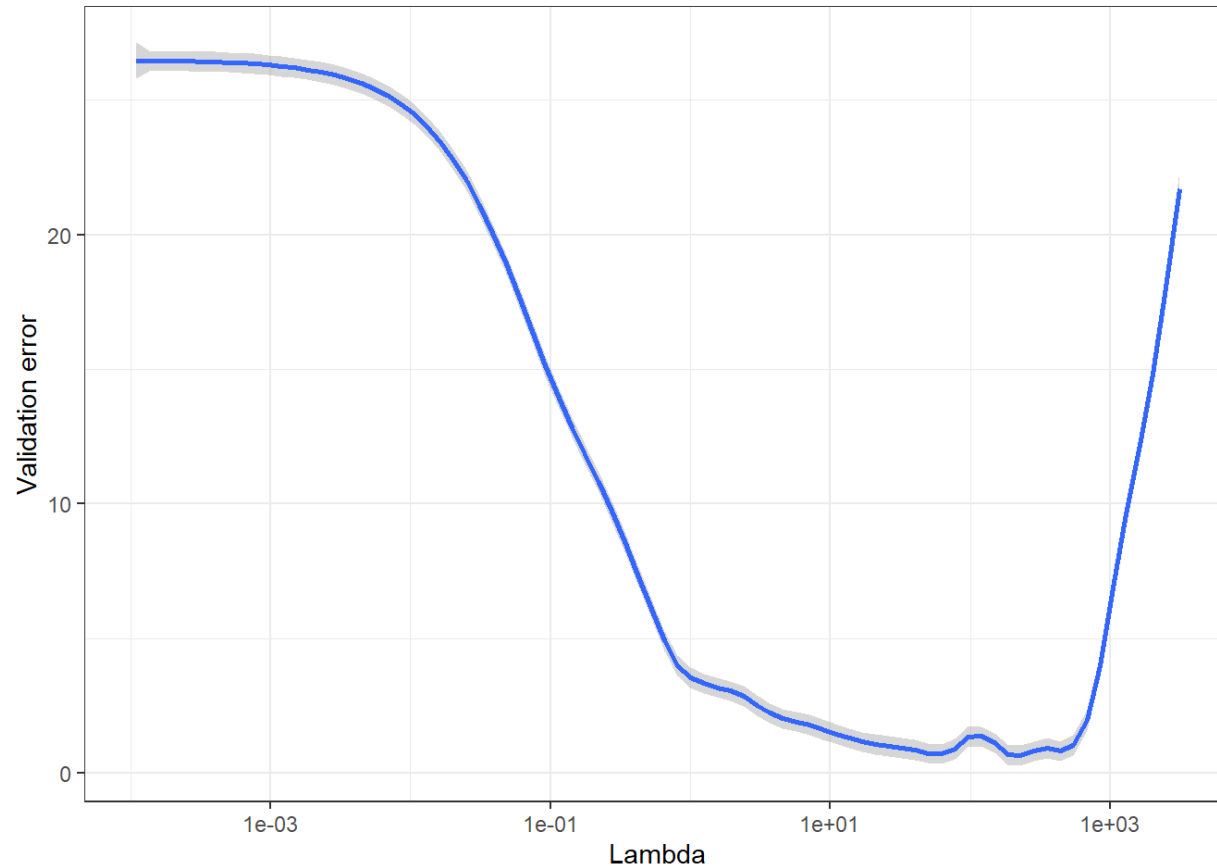
```
lambda_min=0.0001  
lambdas=0.0001*(1.1^seq(250))
```

Train a ridge regression model for each hyper-parameter and compute validation error.

```
ridge_results_df<-data.frame(lambda=lambdas)%>%  
  mutate(out=map(lambda,  
    ~compute_train_validate_error_ridge(train_x=genom_train_x,train_y=genom_train_y,  
    validate_x=genom_validate_x,validate_y=genom_validate_y,lambda=.x)))%>%  
  mutate(train_error=map_dbl(out,~((.x)$train_error)),  
    validate_error=map_dbl(out,~((.x)$validate_error)))%>%select(-out)
```

Ridge regression performance on validation data

Train a ridge regression model for each hyper-parameter and compute validation error.



Find hyper-parameter with minimal validation error

Find minimum validation error.

```
min_validation_error<-ridge_results_df%>%  
  pull(validate_error)%>%min()
```

Find the corresponding hyper-parameter.

```
optimal_lambda<-ridge_results_df%>%  
  filter(validate_error==min_validation_error)%>%  
  pull(lambda)
```

```
optimal_lambda
```

```
## [1] 177.949
```

Ridge regression performance on test data

Extract the ridge regression model with the optimal hyper-parameter.

```
final_ridge_model<-glmnet(x=genom_train_x, y=genom_train_y, alpha = 0, lambda=optimal_lambda)
```

Use the test data to estimate the out-of-sample performance of the trained model.

```
final_ridge_test_y_est<-predict(final_ridge_model,newx=genom_test_x) # test predictions  
final_ridge_test_error = mean((genom_test_y-final_ridge_test_y_est)^2) # test error
```

```
final_ridge_test_error
```

```
## [1] 0.5939739
```

Comparing ridge regression with the OLS

The OLS mean squared error on the test data

```
ols_model<-glmnet(x=genom_train_x, y=genom_train_y, alpha = 0, lambda=0)
```

```
ols_test_y_est<-predict(ols_model,newx=genom_test_x) # test predictions  
ols_test_error = mean((genom_test_y-ols_test_y_est)^2) # test error
```

```
ols_test_error
```

```
## [1] 29.27559
```

The ridge regression mean squared error on the test data.

```
final_ridge_test_error
```

```
## [1] 0.5939739
```

Now take a break!



Statistical Computing & Empirical Methods

Ridge regression

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ of the form $\phi_{w,w^0}(x) = w x^\top + w^0$

The **ridge regression** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_\lambda(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2 + \lambda \cdot \|w\|_2^2.$$

Here $\|w\|_2$ denotes the Euclidean norm (also known as the ℓ_2 norm) defined by

$$\|w\|_2 = \sqrt{(w_1)^2 + \cdots + (w_d)^2} \quad \text{for} \quad w = (w_1, \cdots, w_d) \in \mathbb{R}^d$$

Lasso regression

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ of the form $\phi_{w,w^0}(x) = w x^\top + w^0$

The **Lasso** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_{1,\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2 + \lambda \cdot \|w\|_1.$$

Here $\|w\|_1$ denotes the ℓ_1 norm defined by

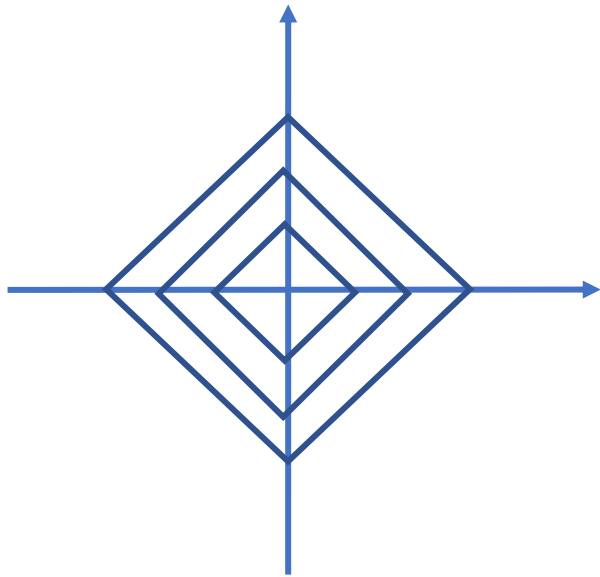
$$\|w\|_1 = |w_1| + \cdots + |w_d| \quad \text{for } w = (w_1, \cdots, w_d) \in \mathbb{R}^d$$

Lasso regression

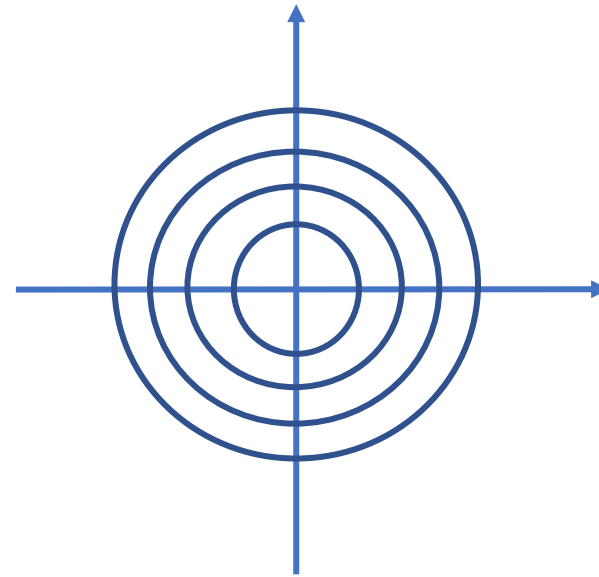
The ℓ_1 and ℓ_2 norms for weight vectors $w = (w_1, \dots, w_d) \in \mathbb{R}^d$

$$\|w\|_1 = |w_1| + \dots + |w_d|$$

$$\|w\|_2 = \sqrt{(w_1)^2 + \dots + (w_d)^2}$$



The LASSO algorithm



The ridge regression algorithm.

Lasso regression

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ of the form $\phi_{w,w^0}(x) = w x^\top + w^0$

The **Lasso** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_{1,\lambda}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2 + \lambda \cdot \|w\|_1.$$

Here $\|w\|_1$ denotes the ℓ_1 norm defined by

$$\|w\|_1 = |w_1| + \cdots + |w_d| \quad \text{for } w = (w_1, \cdots, w_d) \in \mathbb{R}^d$$

Variable selection

Suppose we want to learn a linear regression model $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ in a high-dimensional setting.

There is a very large number of variables d but only a small number are relevant $s \ll d$.

Let $\mathcal{S} \subseteq \{1, \dots, d\}$ be the set of all relevant features with $|\mathcal{S}| = s$.

The ideal solution ϕ_{w,w^0} would have weights $w = (w_1, \dots, w_d) \in \mathbb{R}^d$ with zero weights $w_j = 0$ whenever $j \notin \mathcal{S}$.

An advantage of the LASSO is that it often returns a sparse solution with lots of zero weights $\hat{w}_j = 0$.

We can use LASSO to estimate the set \mathcal{S} by $\hat{\mathcal{S}} = \{j \in \{1, \dots, d\} : \hat{w}_j = 0\}$.

Ridge regression and Lasso in R

We can perform both ridge and Lasso regression in R with the glmnet library.

```
library(glmnet)
```

For ridge regression we set the alpha parameter to zero.

```
ridge_model<-glmnet(x=train_x, y=train_y, alpha = 0, lambda=lambda,family="gaussian")
```

For the lasso algorithm we set the alpha parameter to one.

```
lasso_model<-glmnet(x=train_x, y=train_y, alpha = 1, lambda=lambda,family="gaussian")
```

Comparing lasso regression with ridge regression

Lasso has both advantages and disadvantages in comparison to the Lasso algorithm.



Lasso returns a sparse solution with automated variable selection.

Performs well when there are a small number of relevant features.



Unlike ridge there is no closed form solution for the Lasso.

The results can be unstable when several features are “similar” in some sense.

We can mitigate the instability by combining the Lasso with the ridge regression.

The elastic net algorithm

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \mathbb{R}$ of the form $\phi_{w,w^0}(x) = w x^\top + w^0$

The **elastic net** method minimizes the regularized objective:

$$\hat{\mathcal{R}}_{\alpha,\lambda}(\phi_{w,w^0}) = \frac{1}{2n} \sum_{i=1}^n (w X_i^\top + w^0 - Y_i)^2 + \lambda \left\{ \frac{(1-\alpha)}{2} \cdot \|w\|_2 + \alpha \cdot \|w\|_1 \right\}.$$

Here $\|w\|_2 = \sqrt{(w_1)^2 + \cdots + (w_d)^2}$ and $\|w\|_1 = |w_1| + \cdots + |w_d|$

The α parameter governs the trade off between the two forms of regularization.

The elastic net algorithm

We can perform both ridge and Lasso regression in R with the glmnet library.

```
library(glmnet)
```

```
elastic_net_model<-glmnet(x=train_x, y=train_y, alpha = alpha, lambda=lambda,family="gaussian")
```

A value of α close to one tends to give sparse solutions with less instability than the Lasso.

How to we choose value for the two hyper-parameters α and λ ?

Typically this is done by performance on a validation set.

Now take a break!



Statistical Computing & Empirical Methods

Logistic regression

Recall that logistic regression algorithm learns a binary classifier $\phi_{w,w^0} : \mathbb{R}^d \rightarrow \{0, 1\}$

which is of the form $\phi_{w,w^0}(x) = \mathbb{1} \{w x^\top + w^0 \geq 0\}$

Given a labelled data set $\mathcal{D} = ((X_1, Y_1), \dots, (X_n, Y_n))$ the logistic regression algorithm **maximises** the log-likelihood

$$\log \ell(w, w^0) = \sum_{i=1}^n \log S((2Y_i - 1) \cdot (w X_i^\top + w^0))$$

Equivalently, we **minimize** the rescaled negative log-likelihood

$$-\frac{1}{n} \log \ell(w, w^0) = \frac{1}{n} \sum_{i=1}^n \{-\log S((2Y_i - 1) \cdot (w X_i^\top + w^0))\}$$


Regularised logistic regression

Logistic regression **minimizes** the rescaled negative log-likelihood

$$-\frac{1}{n} \log \ell(w, w^0) = \frac{1}{n} \sum_{i=1}^n \{ -\log S((2Y_i - 1) \cdot (w X_i^\top + w^0)) \}$$

We can regularize logistic regression by also penalizing $\|w\|_2 = \sqrt{(w_1)^2 + \dots + (w_d)^2}$.

L2 regularized logistic regression minimizes the regularized loss:

$$\mathcal{R}_{0,\lambda}^{\text{bn}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n \{ -\log S((2Y_i - 1) \cdot (w X_i^\top + w^0)) \} + \frac{\lambda}{2} \|w\|_2^2$$


Fit to the data

Regularization term

By damaging our performance on the training data we often actually perform better on test data.


Regularised logistic regression

Logistic regression **minimizes** the rescaled negative log-likelihood

$$-\frac{1}{n} \log \ell(w, w^0) = \frac{1}{n} \sum_{i=1}^n \{ -\log S((2Y_i - 1) \cdot (w X_i^\top + w^0)) \}$$

We can also regularize logistic regression by also penalizing $\|w\|_1 = |w_1| + \dots + |w_d|$.

L1 regularized logistic regression minimizes the regularized loss:

$$\mathcal{R}_{1,\lambda}^{\text{bn}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n \{ -\log S((2Y_i - 1) \cdot (w X_i^\top + w^0)) \} + \lambda \|w\|_1$$


Fit to the data

Regularization term

Penalizing the L1 norm gives rise to sparse solutions but can be unstable.

Regularised logistic regression

L1 regularized logistic regression minimizes the l1 regularized loss:

$$\mathcal{R}_{1,\lambda}^{\text{bn}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n \{-\log S((2Y_i - 1) \cdot (w X_i^\top + w^0))\} + \lambda \|w\|_1$$

L2 regularized logistic regression minimizes the l2 regularized loss:

$$\mathcal{R}_{0,\lambda}^{\text{bn}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n \{-\log S((2Y_i - 1) \cdot (w X_i^\top + w^0))\} + \frac{\lambda}{2} \|w\|_2^2$$

We can also combine these two forms of regularization.

$$\mathcal{R}_{\alpha,\lambda}^{\text{bn}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n \{-\log S((2Y_i - 1) \cdot (w X_i^\top + w^0))\} + \lambda \left\{ \frac{(1 - \alpha)}{2} \cdot \|w\|_2^2 + \alpha \cdot \|w\|_1 \right\}.$$

The hyper parameters α and λ can be chosen by performance on a validation set.

Regularized logistic regression

We can perform both regularized logistic regression by using the glmnet library.

```
library(glmnet)
```

```
reg_log_model<-glmnet(x=train_x, y=train_y, alpha = alpha, lambda=lambda,family="binomial")
```

This optimizes the regularized logistic loss function given by

$$\mathcal{R}_{\alpha,\lambda}^{\text{bn}}(\phi_{w,w^0}) = \frac{1}{n} \sum_{i=1}^n \left\{ -\log S \left((2Y_i - 1) \cdot (w X_i^\top + w^0) \right) \right\} + \lambda \left\{ \frac{(1 - \alpha)}{2} \cdot \|w\|_2^2 + \alpha \cdot \|w\|_1 \right\}.$$

When $\alpha = 0$ this carries out l2-regularized logistic regression.

When $\alpha = 1$ this carries out l1-regularized logistic regression.

What have we covered today?

- In this lecture we introduced the important topic of regularization and hyper-parameters
- We saw how regularization can improve performance on test data at the expense of performance on train data.
- The level of regularization is controlled through hyper-parameters.
- To tune our hyper-parameters we must use validation data, obtained via a train-validate-test split.
- We saw that multiple forms of regularization are available including l1 and l2 regularization.
- L1 regularization performs automatic variable selection but can be unstable.
- We saw how the concepts of regularization also apply within the context of logistic regression.



University of
BRISTOL

Thanks for listening!

henry.reeve@bristol.ac.uk

Include EMATM0061 in the subject of your email.

Statistical Computing & Empirical Methods