What we will cover today

We will introduce the important concept of regularisation.

Regularisation will allow us to do better on unseen data, whilst performing worse on training data.

This leads to hyper-parameters which control the regularisation level.

To tune our hyper-parameters we will need to investigate the train-validate-test split.

We will see how regularisation applies to both regression and classification problems.

Ridge regression

Ordinary least squares and problems

The Ordinary Least Squares method (OLS) minimises $\widehat{\mathcal{R}}_{\text{MSE}}\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$ over all possible linear models $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ by $\phi_{w,w^0}(x) = wx^T + w^0$.

The OLS solution is $\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$ and $\hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$. Here $\hat{w}$ and $\hat{w}^0$ are estimates:

1. Maximum likelihood estimators of w and w^0
2. Unbiased estimators with $\mathbb{E}(\hat{w})$ and $\mathbb{E}\left(\hat{w}^0\right)$
3. Minimum variance over all unbiased estimators (known as the Gauss Markov theorem).

Ordinary least squares and problems

The Ordinary Least Squares method (OLS) minimises $\widehat{\mathcal{R}}_{\text{MSE}}(\phi_{w,w^0}) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$ over all possible linear models $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ by $\phi_{w,w^0}(x) = wx^T + w^0$.

The OLS solution is $\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$ and $\hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$.

However, 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 a small determinant.

6. Sometimes $\Sigma_{X,X}$ is non-invertible so $\hat{w}$ is not defined.

Ordinary least squares can still perform very poorly in high dimensional settings.

We will see that regularisation is a technique for overcoming these limitations.

High dimensional regression problems

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

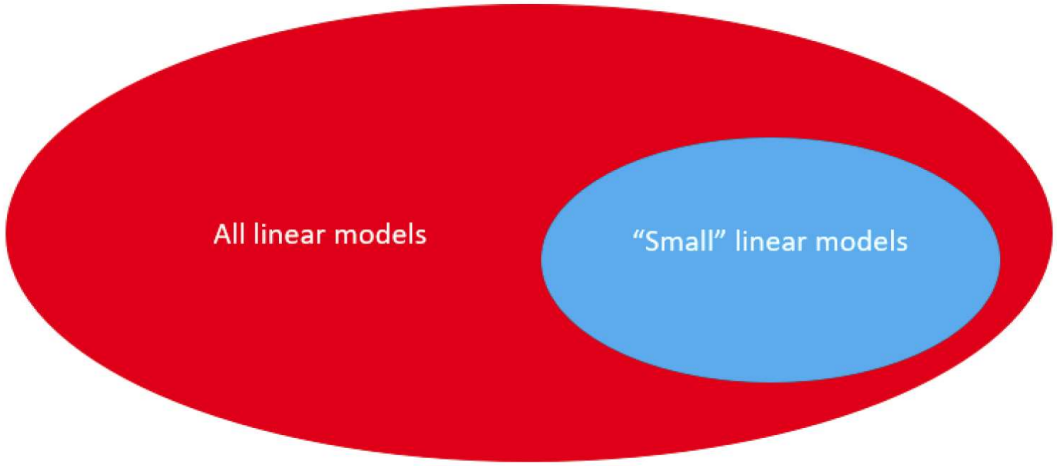
Computer vision

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 a subset of the large space that contains "smaller" models.



The assumption is that the "smaller models" subset contains reasonable models that we want to find.

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 a subset of the large space that contains "smaller" models.

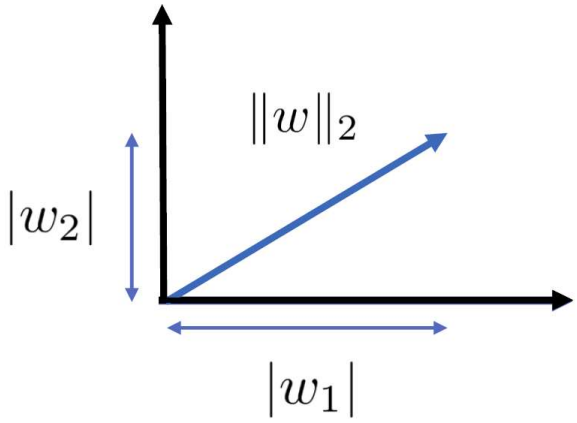
But how do we quantify the size of a model?

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ and $\phi_{w,w^0} = wx^T + w^0$.

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

The Euclidean norm

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



Ridge regression

Idea: to reduce the instability of our estimate we limit our search space to a subset of the large space that contains "smaller" models.

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ and $\phi_{w,w^0} = wx^T + w^0$.

The Euclidean norm $\|w\|_2 = \sqrt{(w_1)^2 + \cdots + (w_d)^2}$

The OLS minimises empirical error $\widehat{\mathcal{R}}_{\text{MSE}}(\phi_{w,w^0}) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$

But additionally, we are interested in only the "smaller" models, i.e., the models with small $\|w\|_2$.

Therefore, the ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_{\lambda}(\phi_{w,w^0}) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

Mean squared error on training data Regularization term

So ridge regression is a modified version of OLS (by adding a regularisation term)

Ridge regression

Idea: to reduce the instability of our estimate we limit our search space to a subset of the large space that contains "smaller" models.

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ and $\phi_{w,w^0} = wx^T + w^0$.

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

Mean squared error on training data

Regularisation term

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

We need to choose the value of λ such that it balances both terms.

The higher λ is, the higher the level of regularisation.

Ridge regression – closed-form solution

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda(\phi_{w,w^0}) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

Mean squared error on training data — Regularisation term

The solution for the ridge regression method is:

$$\hat{w}_\lambda = \Sigma_{Y,X}(\Sigma_{X,X} + \lambda I_d)^{-1} \quad \hat{w}^0 = \overline{Y} - \hat{w}_\lambda \overline{X}^T$$

where

$$\overline{X} := \frac{1}{n}\sum_i^n X_i \qquad \overline{Y} := \frac{1}{n}\sum_i^n Y_i$$

$$\Sigma_{X,X} := \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^T(X_i - \overline{X})$$

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

$$\mathrm{I}_d = \begin{pmatrix} 1 & 0 & \cdots & 0 & \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & \cdots & 0 & 1 \end{pmatrix}$$

(d columns, d rows)

$d \times d$ identity matrix

In comparison, the OLS solutions are $\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$ $\quad \hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$

Ridge regression – closed-form solution

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

Mean squared error on training data — Regularisation term

The solution for the ridge regression method is:

$$\hat{w}_\lambda = \Sigma_{Y,X}(\Sigma_{X,X} + \lambda I_d)^{-1} \quad \hat{w}^0 = \overline{Y} - \hat{w}_\lambda \overline{X}^T$$

Observation: Whenever $\lambda > 0$, the matrix $\Sigma_{X,X} + \lambda I_d$ is invertible.

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

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

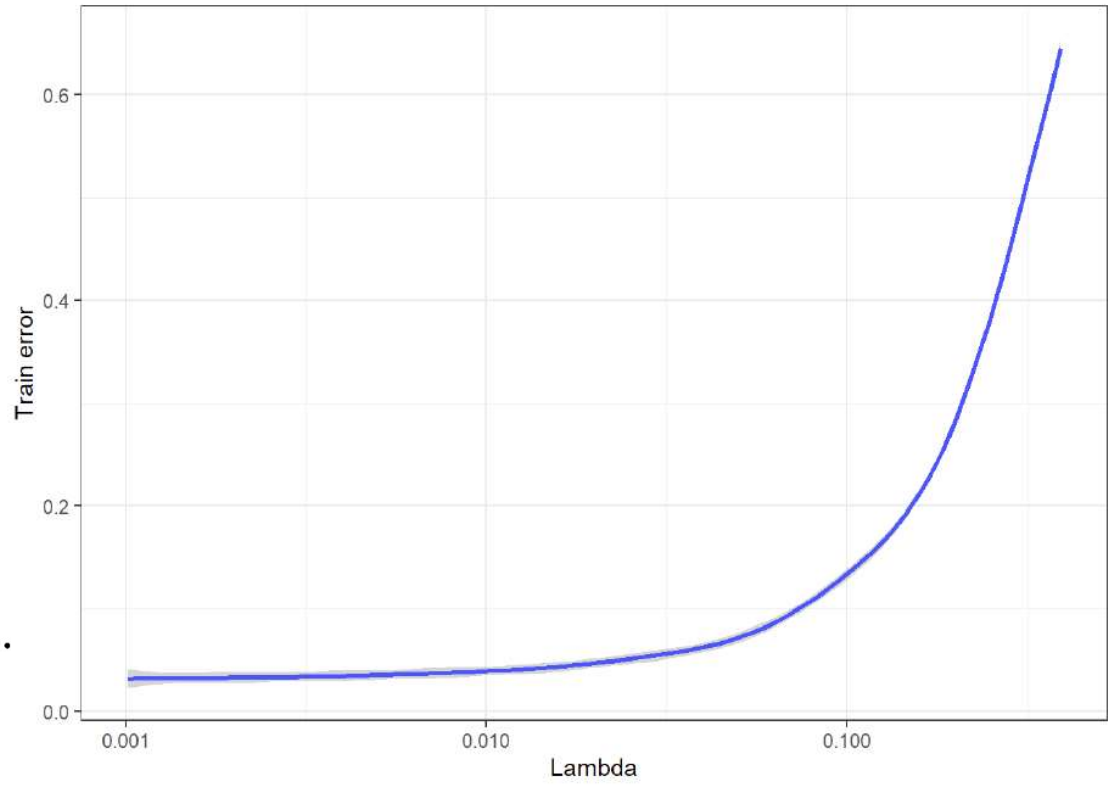
The training error vs hyper-parameter

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

train error
$\frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$

Regularisation can worsen performance on training data.



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

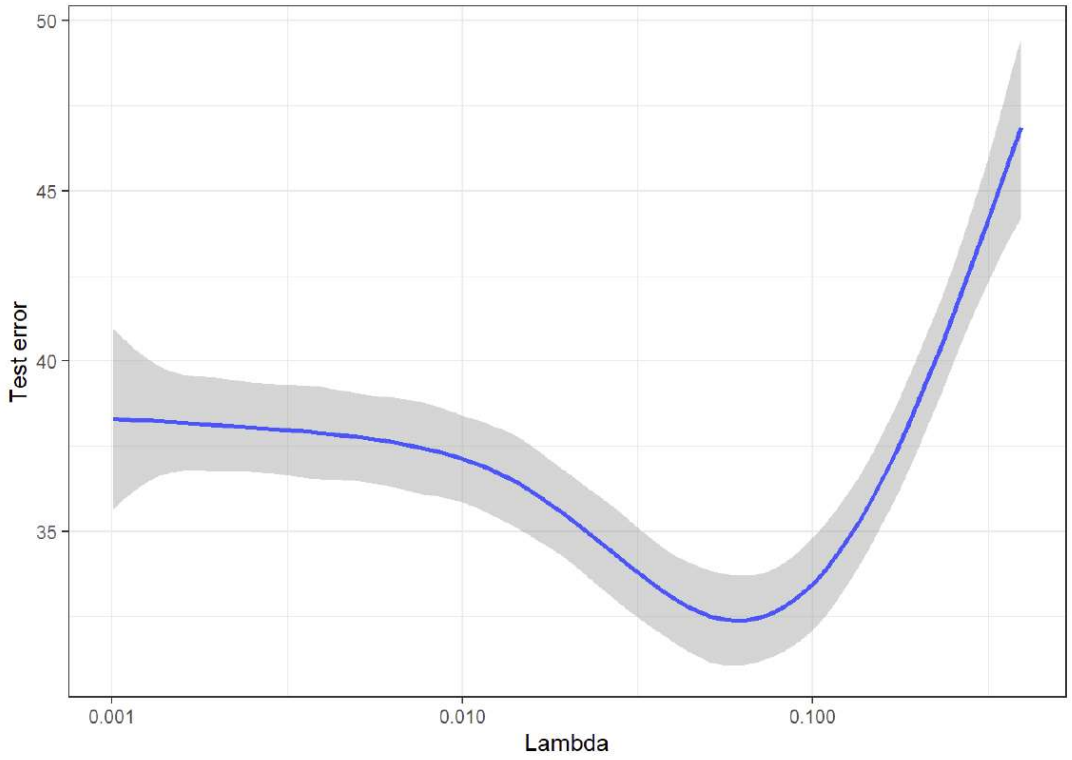
The test error vs hyper-parameter

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

test error:
the MES on test data

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



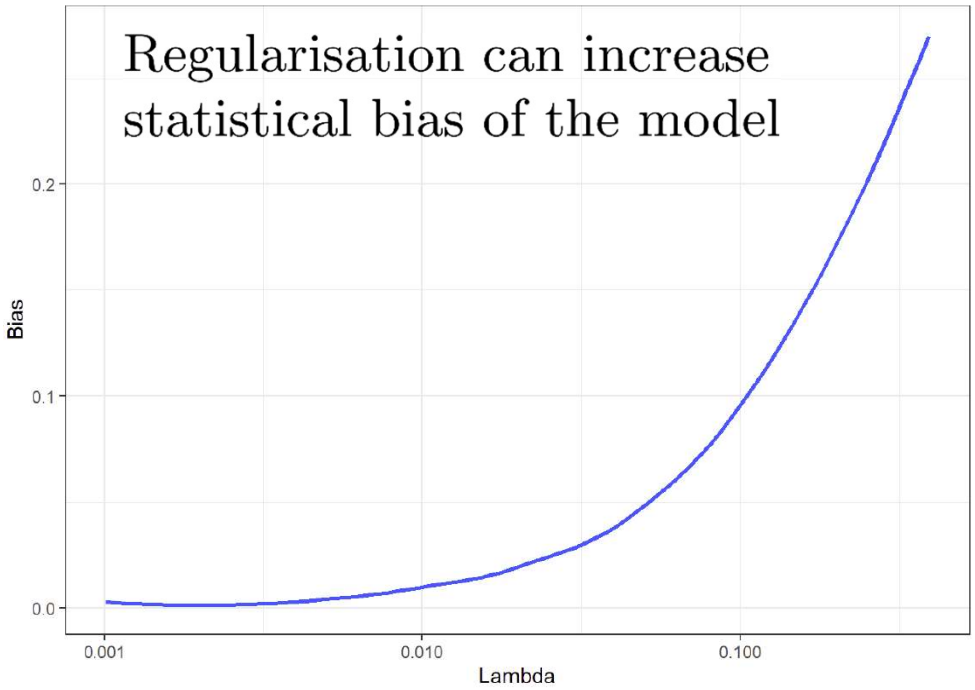
As the hyper-parameter λ rises the mean squared error on the test data falls before rising again!

The bias-variance perspective on ridge regression

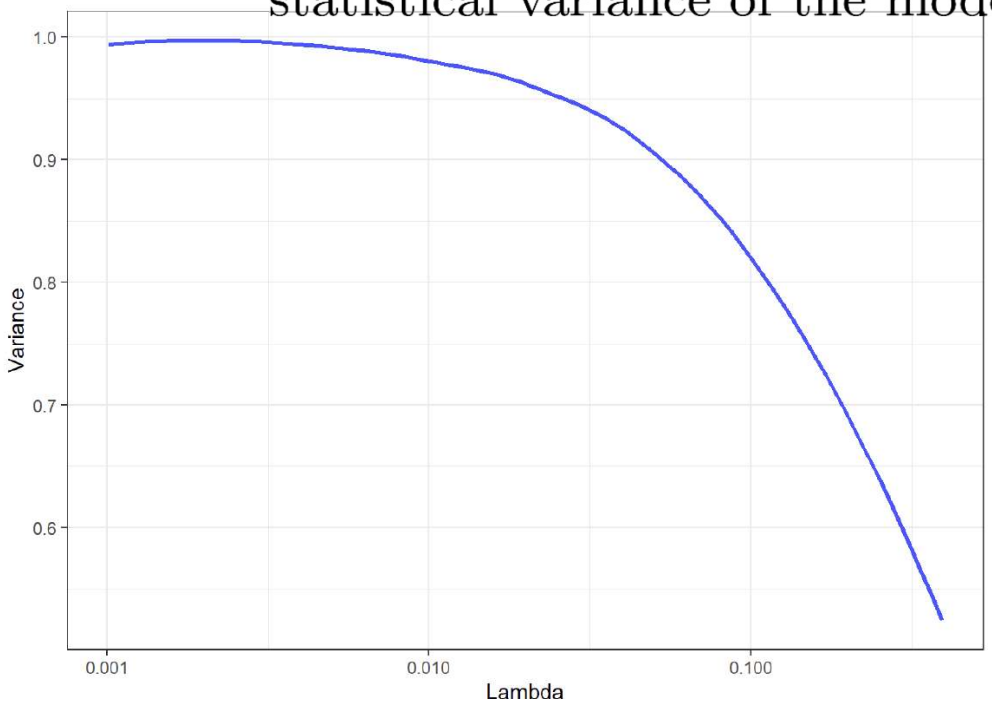
The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

Regularisation can increase statistical bias of the model



Regularisation can decrease statistical variance of the model



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

The bias-variance perspective on ridge regression

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

As we increase the hyper-parameter λ we observe:

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

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

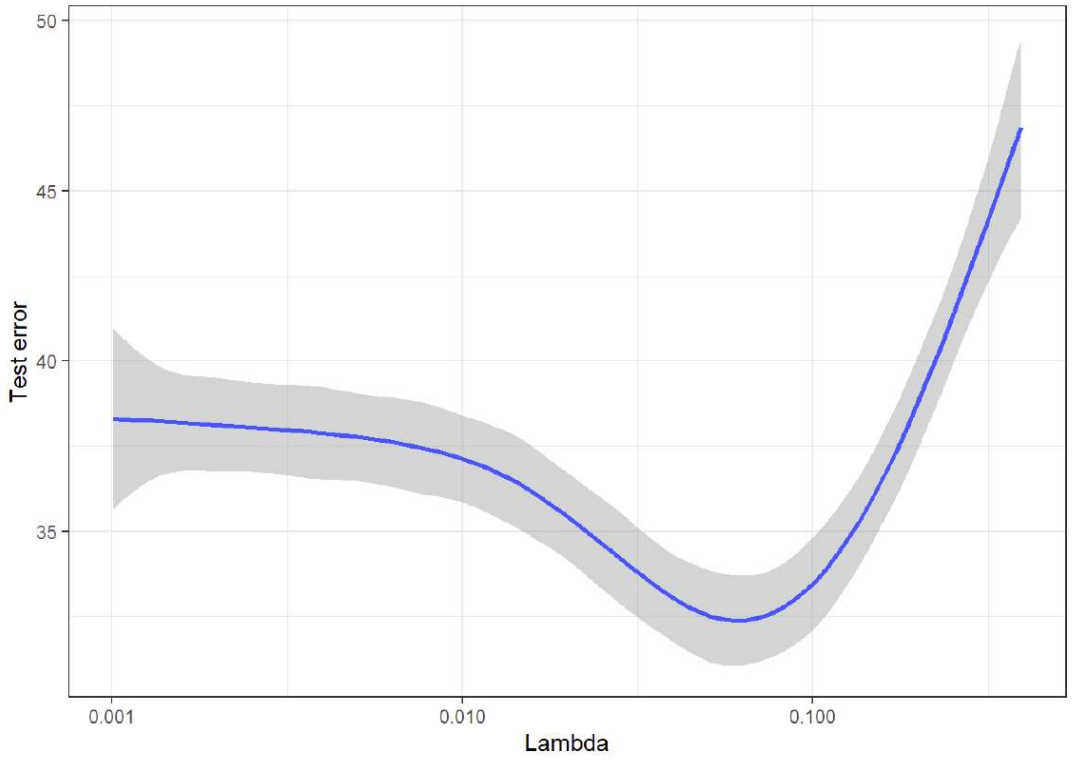
Regularisation can improve performance on test data by damaging performance on training data.

Hyper-parameter

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

test error:
the MES on test data



How should we select the hyper-parameter λ?

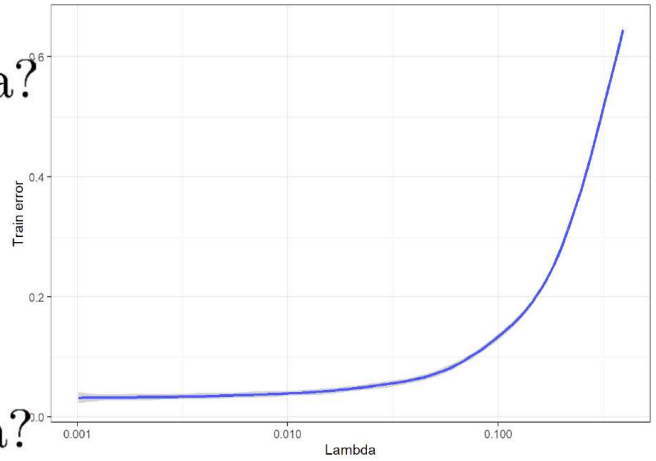
Choosing hyperparameters

Choosing hyper-parameters

How should we select the hyper-parameters?

Our goal is to select the hyper-parameters such that the obtained model performs well on unseen data.

Option 1. Select λ to minimise error on the training data?



This is how we select the weights.

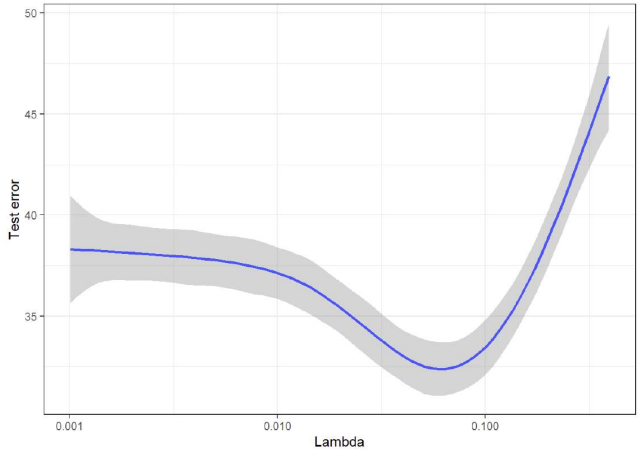
Problem: This will always lead to $\lambda = 0$.

Option 2. Select λ to minimise 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

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

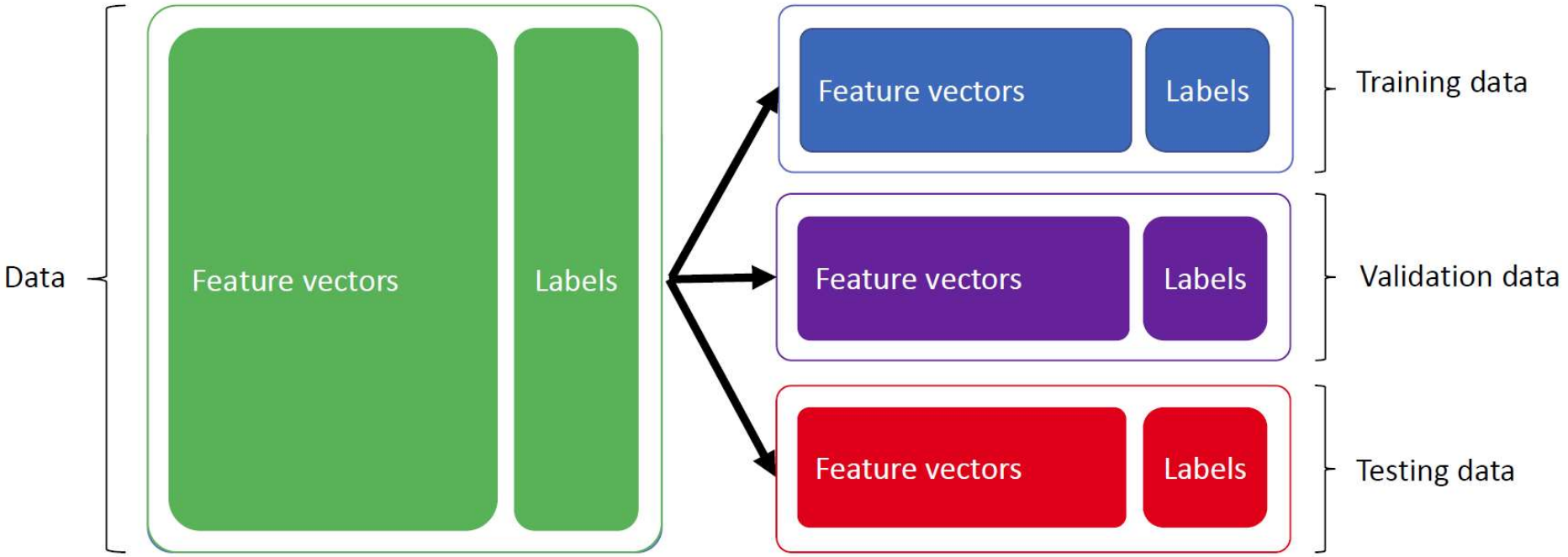


We require an additional subset of data: The validation data set.

The train-validation-test split

How should we select the hyper-parameters?

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



The train-validation-test split

To choose the hyper-parameter, we begin by selecting a range of possible hyper-parameters $\lambda_1, \cdots, \lambda_Q$.

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

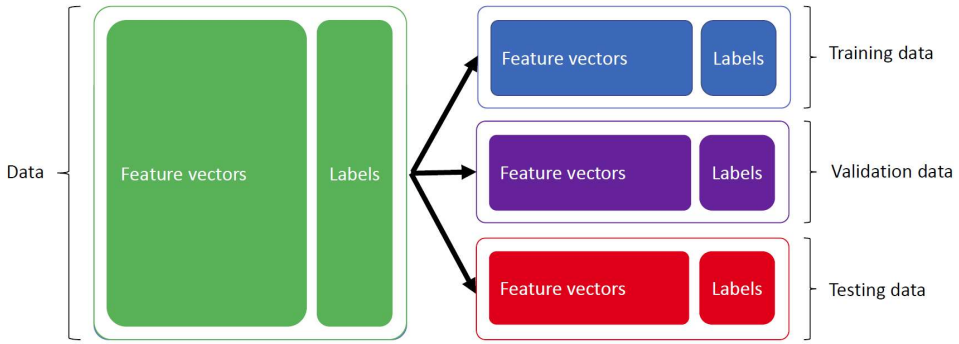
(**train**) Train a model based on the training data and the hyperparameter λ_q.

(**val**) Compute the average error on the validation data

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

(**test**) Use the test data to estimate the performance of the model on unseen data.

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



Ridge regression example

Example: Phenotype regression with genomic data

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

Feature vector 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.

suppose that we have the data:

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

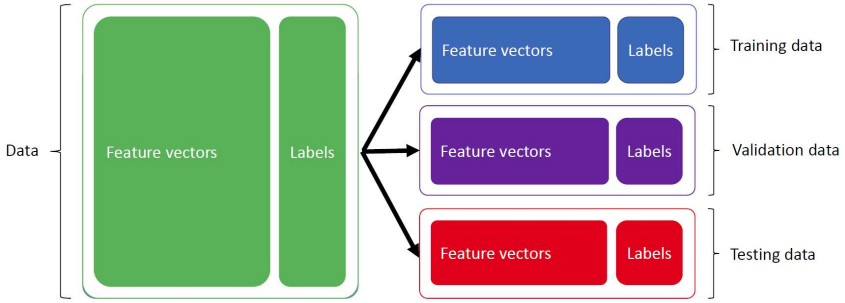
```
## [1] 1000  501
```

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

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

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.

(**train**) Train a model based on the training data and the hyperparameter λ_q.

(**val**) Compute the average error on the validation data

```
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))
```

To choose the hyper-parameter, we begin by selecting a range of possible hyper-parameters $\lambda_1, \cdots, \lambda_Q$.

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)
```

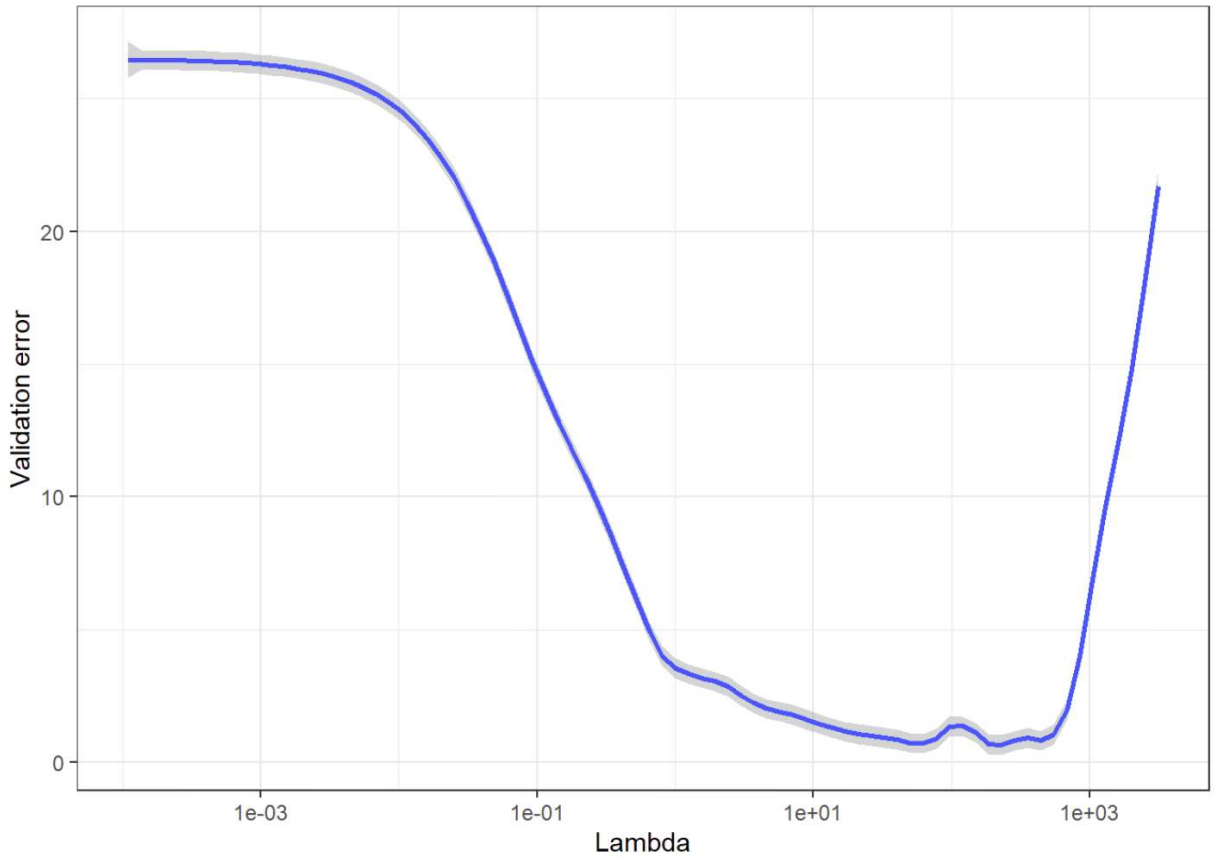
For each $q = 1, \cdots, Q$,

(**train**) Train a model based on the training data and the hyperparameter λ_q.

(**val**) Compute the average error on the validation data

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()
```

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

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.

(**test**) Use the test data to estimate the performance of the model on unseen data.

```
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
```

The improvement is due to the regularisation in ridge regression!

Lasso regression

Recall: Ridge regression

Consider linear models $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ and $\phi_{w,w^0} = wx^T + w^0$.

The ridge regression method minimises the regularised objective:

$$\widehat{\mathcal{R}}_\lambda\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_2^2$$

Mean squared error on training data

Regularisation term

Here $\|w\|_2$ denotes the Euclidean norm (also known as l_2 the 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 $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ and $\phi_{w,w^0} = wx^T + w^0$.

The Lasso method minimises the regularised objective:

$$\widehat{\mathcal{R}}_{1,\lambda}(\phi_{w,w^0}) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_1$$

Mean squared error on training data

Regularisation term

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

$$\|w\|_1 = |w_1| + \cdots + |w_d| \quad \text{for} \quad w = (w_1, \cdots, w_d) \in \mathbb{R}^1$$

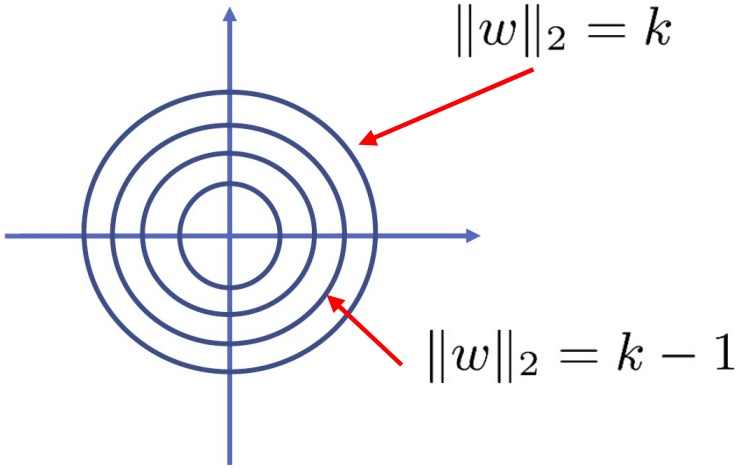
note: l_1 norm, instead of l_2 norm, is used here.

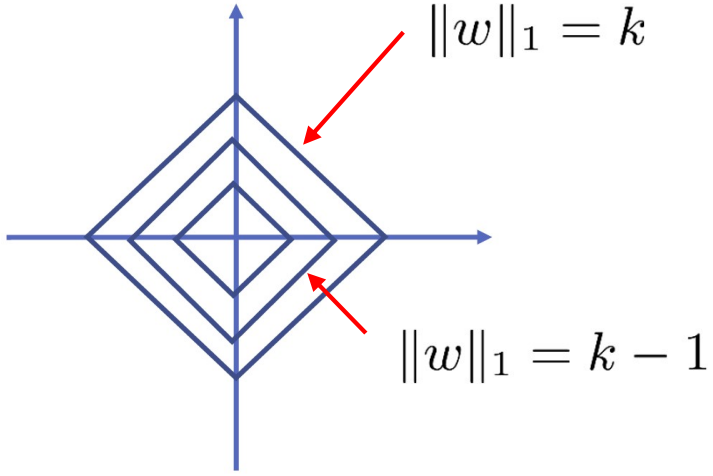
Lasso regression

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

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

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





The ridge regression algorithm.

The LASSO algorithm

The Lasso method minimises the regularised objective:

$$\widehat{\mathcal{R}}_{1,\lambda}\left(\phi_{w,w^0}\right) := \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\|w\|_1$$

Variable selection

Suppose we want to learn a linear regression model ϕ_{w,w^0} 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 $\mathcal{S} \subseteq \{1, 2, \cdots, d\}$ be the set of all relevant features with $|\mathcal{S}| = s$. We are interested in $\mathcal{S}$ but it is unknown.

The ideal solution ϕ_{w,w^0} would have weights $w = (w_1, \cdots, w_d) \in \mathbb{R}^d$ 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 $\hat{\mathcal{S}} = \{j \in \{1, \cdots, d\} : \hat{w}_j \neq 0\}$

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 ridge regression 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 $\phi_{w,w^0} : \mathbb{R}^d \to \mathbb{R}$ and $\phi_{w,w^0} = wx^T + w^0$.

The elastic net method minimises the regularised objective:

$$\widehat{\mathcal{R}}_{1,\lambda}(\phi_{w,w^0}) := \tfrac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2 + \lambda\left(\tfrac{1-\alpha}{2}\|w\|_2^2 + \alpha\|w\|_1\right)$$

Mean squared error on training data

Regularisation term

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

The parameter α governs the trade-off between the two forms of regularisation

$\alpha = 0$ for ridge and $\alpha = 1$ for Lasso.

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")
```

$\alpha = 0$ for ridge and $\alpha = 1$ for Lasso.

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

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

Typically this is done by performance on a validation set.

Regularised logistic regression

Recall: Logistic regression

Recall that logistic regression algorithm learns a binary classifier $\phi_{w,w^0} : \mathbb{R}^d \to \{0,1\}$ which is of the form $\phi_{w,w^0} = \mathbb{1}\{wx^T + w^0 \geq 0\}$.

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

$$\log l(w, w^0) = \sum_{i=1}^{n} \log S\Big((2Y_i - 1) \cdot \Big(wX_i^T + w^0\Big)\Big)$$

Equivalently, we minimise the rescaled negative log-likelihood

$$-\frac{1}{n} \log l(w, w^0) = -\frac{1}{n} \sum_{i=1}^{n} \log S\Big((2Y_i - 1) \cdot \Big(wX_i^T + w^0\Big)\Big)$$

Regularised logistic regression

Logistic regression minimise the rescaled negative log-likelihood

$$-\frac{1}{n}\log l(w, w^0) = -\frac{1}{n}\sum_{i=1}^{n}\log S\Big((2Y_i - 1)\cdot\Big(wX_i^T + w^0\Big)\Big)$$

We can regularise logistic regression by also penalising $\|w\|_2 = \sqrt{(w_1)^2 + \cdots + (w_d)^2}$.

l_2 regularised logistic regression minimises the regularised loss:

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

fit to data

Regularisation term

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

Regularised logistic regression

Similar to Lasso, we can also use l_1 regularisation

l_1 regularised logistic regression minimises the regularised loss:

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

fit to data

Regularisation term

Penalising the l_1 norm gives rise to sparse solutions but can be unstable, so we can consider the combined version:

combined regularised logistic regression minimises the regularised loss:

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

fit to data

Regularisation term

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

Regularised logistic regression in R

We can perform both regularised 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")
```

$\alpha = 0$ for l_2 regularisation and $\alpha = 1$ for l_1 regularisation.

This optimises the regularised logistic loss function given by

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

What have we covered?

In this lecture we introduced the important topic of regularisation and hyper-parameters

We saw how regularisation can improve performance on test data at the expense of performance on train data.

The level of regularisation is controlled through hyperparameters.

To tune our hyper-parameters we must use validation data, obtained via a train-validate-test split.

We saw that multiple forms of regularisation are available including L1 and L2 regularisation.

L1 regularisation performs automatic variable selection but can be unstable.

We saw how the concepts of regularisation also apply within the context of logistic regression.