MATH60026/MATH70026
Methods for Data Science
Lecture 1

Barbara Bravi, Imperial College London

Department of Mathematics, Academic year 2024-2025

IMPERIAL

Learning from Data

Data Science: an interdisciplinary field concerning methods and strategies to extract knowledge, insights, predictions from data

Learning from Data

Data Science: an interdisciplinary field concerning methods and strategies to extract knowledge, insights, predictions from data

Multiplicity of data types, the main ones being:

1. categorical (or discrete): take on one of a limited (fixed) number of values, such as classes: name of a team, country, type of chemical etc.

Learning from Data

Data Science: an interdisciplinary field concerning methods and strategies to extract knowledge, insights, predictions from data

Multiplicity of data types, the main ones being:

1. categorical (or discrete): take on one of a limited (fixed) number of values, such as classes: name of a team, country, type of chemical etc.

2. quantitative (or continuous): the variable is numerical and represents a measurable quantity: height, speed, price, concentration of a chemical etc.

Learning from Data

Mathematically, we start from a set of data specified by:

Number of data points: measurements, individuals ...

$$\left\{ x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)} \right\}_{i=1}^{N}$$

Number of features (descriptors): the characteristics of each data point (variables measured, attributes of an individual …)

Learning from Data

Mathematically, we start from a set of data specified by:

Number of data points: measurements, individuals ...

$$\left\{ x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)} \right\}_{i=1}^{N}$$

Number of features (descriptors): the characteristics of each data point (variables measured, attributes of an individual …)

We might want to carry out different analysis tasks:

Supervised learning: modelling an input – output mapping;
Unsupervised learning: discovering structure and properties.

Learning from Data

Supervised learning: we have pairs of input – output data

$$\{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, \quad y^{(i)}\}_{i=1}^{N}$$

Input data, whose features are *predictors*

Output data (*outcomes*) to predict

Learning from Data

Supervised learning: we have pairs of input – output data

$$\{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, \quad y^{(i)}\}_{i=1}^{N}$$

Input data, whose features are *predictors*

Output data (*outcomes*) to predict

If *continuous*: the task is to predict its quantitative value (**regression**)

If *categorical*: the task is to predict (or assign to) the category (**classification**)

Linear Regression

We model the input – output mapping through a *linear* function:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p =: f_{\mathrm{LR}}(\mathbf{x}, \boldsymbol{\beta})$$

Why linear? A priori knowledge, data exploration, convenience.

Linear Regression

We model the input – output mapping through a *linear* function:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p =: f_{\mathrm{LR}}(\mathbf{x}, \boldsymbol{\beta})$$

Why linear? A priori knowledge, data exploration, convenience.

The vector of coefficients:

$$\boldsymbol{\beta} = \underbrace{\begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix}}_{(p+1)\times 1} \in \mathbb{R}^{(p+1)}$$

contains the parameters of the models, that are *learnt* from data.

Linear Regression

We need the parameter values making the LR satisfied for all data:

$$\left\{\begin{array}{l} \hat{y}^{(1)} = \beta_0 + \beta_1 x_1^{(1)} + \cdots + \beta_p x_p^{(1)} \stackrel{?}{=} y^{(1)} \\ \hat{y}^{(2)} = \beta_0 + \beta_1 x_1^{(2)} + \cdots + \beta_p x_p^{(2)} \stackrel{?}{=} y^{(2)} \\ \quad \vdots \\ \hat{y}^{(N)} = \beta_0 + \beta_1 x_1^{(N)} + \cdots + \beta_p x_p^{(N)} \stackrel{?}{=} y^{(N)} \end{array}\right\}$$

Outcome *predicted by LR*

Outcome values *given, known*

Linear Regression

We need the parameter values making the LR satisfied for all data:

$$\left\{ \begin{array}{l} \hat{y}^{(1)} = \beta_0 + \beta_1 x_1^{(1)} + \cdots + \beta_p x_p^{(1)} \stackrel{?}{=} y^{(1)} \\ \hat{y}^{(2)} = \beta_0 + \beta_1 x_1^{(2)} + \cdots + \beta_p x_p^{(2)} \stackrel{?}{=} y^{(2)} \\ \quad \vdots \\ \hat{y}^{(N)} = \beta_0 + \beta_1 x_1^{(N)} + \cdots + \beta_p x_p^{(N)} \stackrel{?}{=} y^{(N)} \end{array} \right\}$$

Outcome *predicted by LR*

Outcome values *given, known*

Problem: this linear system is over-determined: $N >> (p+1)$
There is *not* a solution (i.e., parameter values satisfying all eqs).

Least Squares Method

The solution comes from the Least Squares method: it learns optimal parameters making true and predicted outcomes *close*.

Let's first re-write everything in a convenient vector form:

Predictors

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_p^{(1)} \\ 1 & x_1^{(2)} & \dots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \dots & x_p^{(N)} \end{bmatrix}_{N \times (p+1)} \xrightarrow{?} \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}_{N \times 1}$$

Outcome

We model this mapping assuming a linear relationship:

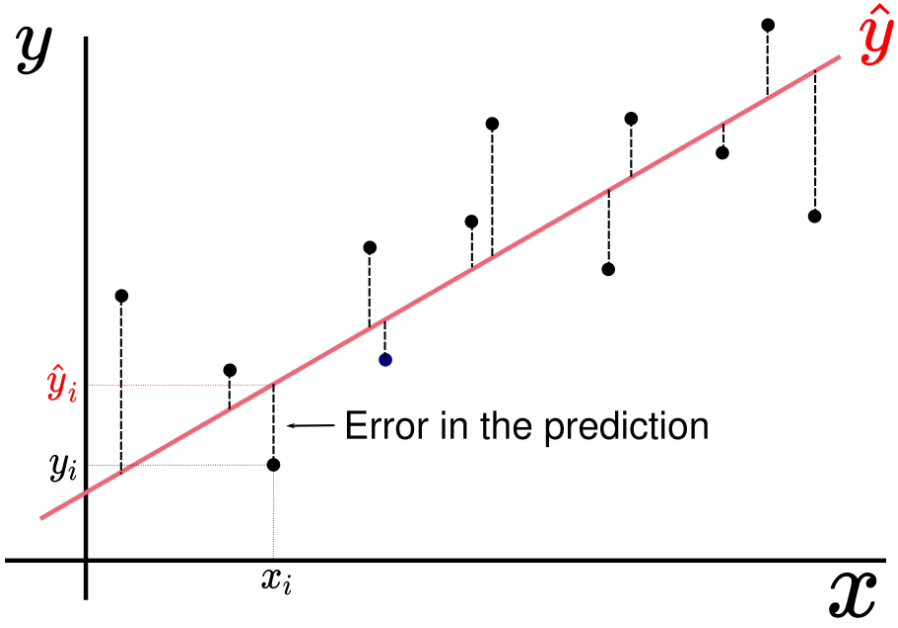
$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$$

With model parameters:

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix}_{(p+1)\times 1} \in \mathbb{R}^{(p+1)}$$

Least Squares Method

Main idea: look at the error between true and predicted outcomes.

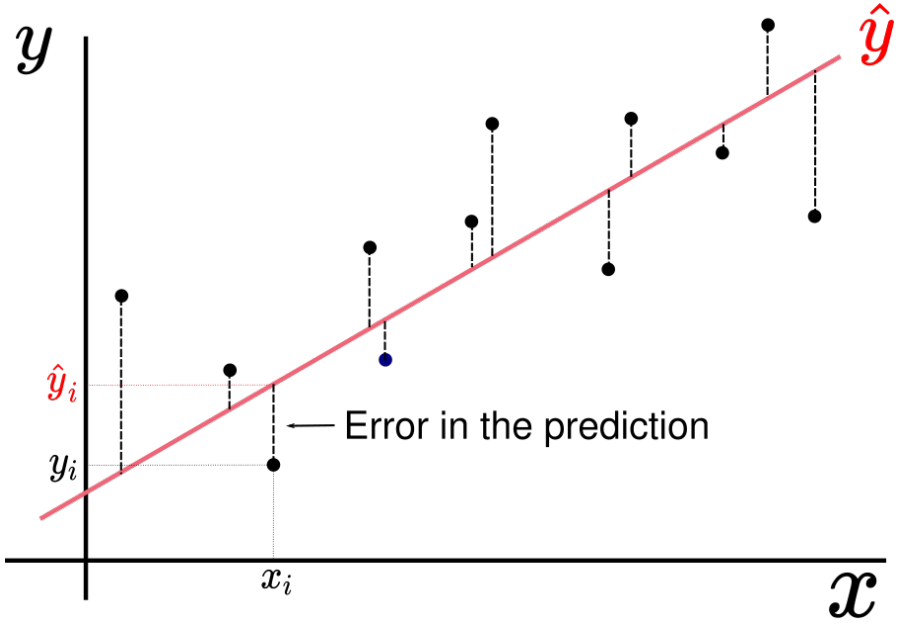


Error of the model:

$$\mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$$

Least Squares Method

Main idea: look at the error between true and predicted outcomes.



Error of the model:

$$\mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$$

Least Squares solution: learn the parameters by minimising the squared errors of prediction across data points:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

Some terminology

In general, a supervised learning task is carried out by minimising a **Loss Function,** here the **Mean Squared Error**:

$$L_{MSE}\big(y, f(x)\big) = \frac{1}{N}\sum_{i=1}^{N}\big(y^{(i)} - f(x^{(i)})\big)^2$$

Some terminology

In general, a supervised learning task is carried out by minimising a **Loss Function,** here the **Mean Squared Error**:

$$L_{MSE}\big(y, f(x)\big) = \frac{1}{N}\sum_{i=1}^{N}\big(y^{(i)} - f(x^{(i)})\big)^2$$

The N data points used to learn the parameters form the **Training Set** (i.e. used to 'train' the model).

Some terminology

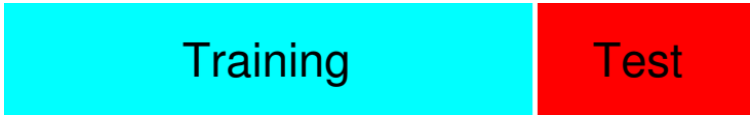
In general, a supervised learning task is carried out by minimising a **Loss Function,** here the **Mean Squared Error**:

$$L_{MSE}(y, f(x)) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - f(x^{(i)})\right)^2$$

The N data points used to learn the parameters form the **Training Set** (i.e. used to 'train' the model).

Once the model is trained, it can be evaluated on *new, unseen* data for which we don't have any given outcome variables. **Generalisability**: ability to perform well on unseen data. To measure it, we need a **Test Set:**

Divide data into:



$$\mathbb{E}\left[L\left(f(\{x^{(k)}\}), \{y^{(k)}\}\right)\right]_{k \in \text{test}}$$

should be small too

Least Squares solution: learn the parameters by minimising the squared errors of prediction across data points:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

1. Analytical, Exact Solution:

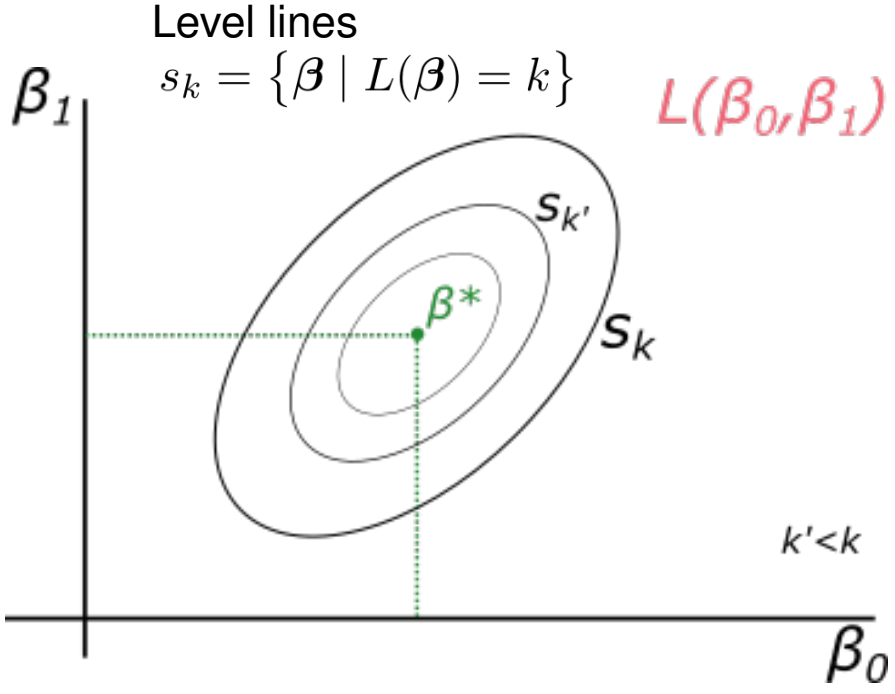
$$\boldsymbol{\beta}^\star = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^+\mathbf{y}$$

Moore-Penrose Pseudo-inverse

Found by carrying out explicitly the minimisation (see notes).

Minimum = Hessian positive definite

$$\nabla_{\boldsymbol{\beta}}(\nabla_{\boldsymbol{\beta}} L) = \frac{2}{N}\mathbf{X}^T\mathbf{X} = H$$



It's a multidimensional parabola, a **convex loss function**: guarantee that we have a global, unique minimum

Least Squares solution: learn the parameters by minimising the squared errors of prediction across data points:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

2. Numerical Solution: via Gradient Descent

Iteration by iteration, we update parameters:

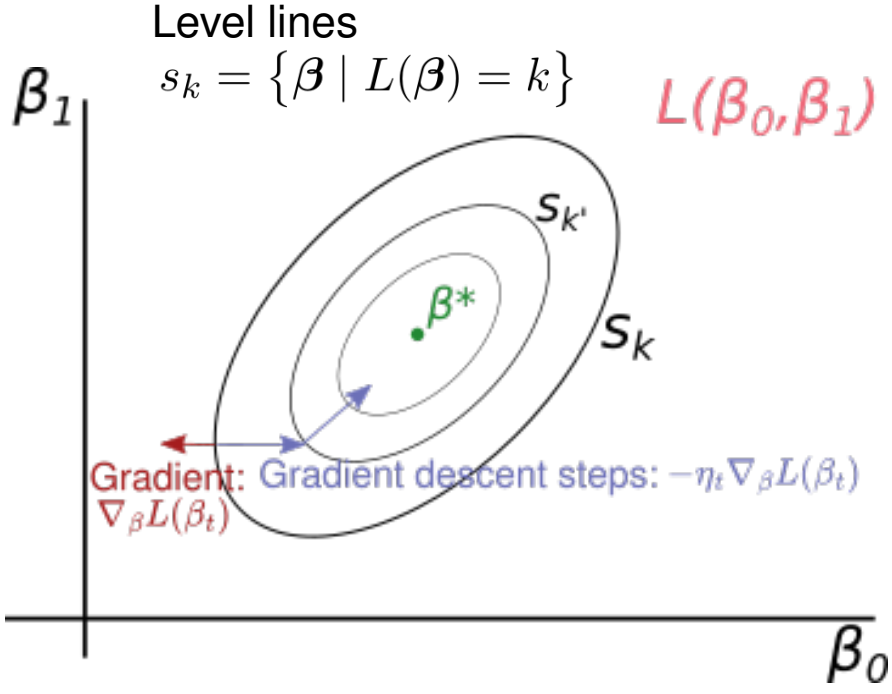
$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}_t)$$

Iteration index

Step size

MSE loss:

$$L(\boldsymbol{\beta}) = \frac{1}{N}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$



Simple idea: the gradient gives the direction of maximal variation; following the opposite of this direction leads to a minimum.

Least Squares solution: learn the parameters by minimising the squared errors of prediction across data points:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

2. Numerical Solution: via Gradient Descent

Iteration by iteration, we update parameters:

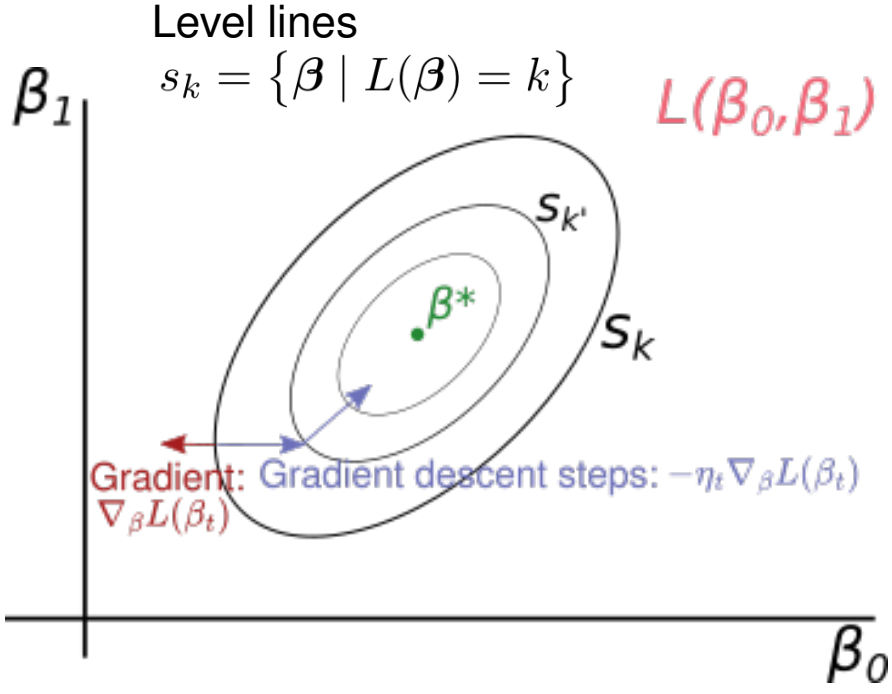
$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}_t)$$

Iteration index

Step size

MSE loss:

$$L(\boldsymbol{\beta}) = \frac{1}{N}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$



Not necessary for LR, but the only viable strategy for non-convex cases

Simple idea: the gradient gives the direction of maximal variation; following the opposite of this direction leads to a minimum.

Statistical Interpretation

The Least Squares solution is **equivalent** to assuming that deviations from the line across data are Gaussian distributed.

$$y^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \varepsilon^{(i)}$$

i.i.d, Gaussian: $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

Statistical Interpretation

The Least Squares solution is **equivalent** to assuming that deviations from the line across data are Gaussian distributed.

$$y^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \varepsilon^{(i)}$$

i.i.d, Gaussian: $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

I can write a likelihood of the data under the model:

$$\underset{\text{tot}}{\text{Lik}}(\mathbf{y} \mid \boldsymbol{\beta}) = \prod_{i=1}^{N} \text{Lik}(y^{(i)} \mid \boldsymbol{\beta})$$

with: $\text{Lik}(y^{(i)} \mid \boldsymbol{\beta}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-\left(y^{(i)} - (\beta_0 + \beta_1 x_1^{(i)})\right)^2}{2\sigma^2}$

and find the best model parameters by maximum likelihood:

$$\underbrace{-\frac{d\mathcal{L}_{\text{tot}}}{d\boldsymbol{\beta}}}_{\text{maximum likelihood}} \longleftrightarrow \underbrace{\frac{dL_{\text{MSE}}}{d\boldsymbol{\beta}}}_{\text{minimal loss (MSE)}}$$

Same solution!

Bias-Variance trade-off

Let's look at the learning problem from a statistical perspective.

$\boldsymbol{\beta}$: True parameter $\boldsymbol{\beta}^\star$: LS estimate from data

Many realisations of data sample = many $\boldsymbol{\beta}^\star$ with a distribution

Two key quantities to analyse the performance of an *estimator*:

$$\text{Bias: } \|\mathbb{E}[\boldsymbol{\beta}^\star] - \boldsymbol{\beta}\|$$

$$\text{Error Covariance Matrix: } \mathbb{E}\big[(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)^T\big]$$

Bias-Variance trade-off

Let's look at the learning problem from a statistical perspective.

$\boldsymbol{\beta}$: True parameter $\quad\quad$ $\boldsymbol{\beta}^\star$: LS estimate from data

Many realisations of data sample = many $\boldsymbol{\beta}^\star$ with a distribution

Two key quantities to analyse the performance of an *estimator*:

$$\text{Bias: } \|\mathbb{E}[\boldsymbol{\beta}^\star] - \boldsymbol{\beta}\|$$

$$\text{Error Covariance Matrix: } \mathbb{E}\left[(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)^T\right]$$

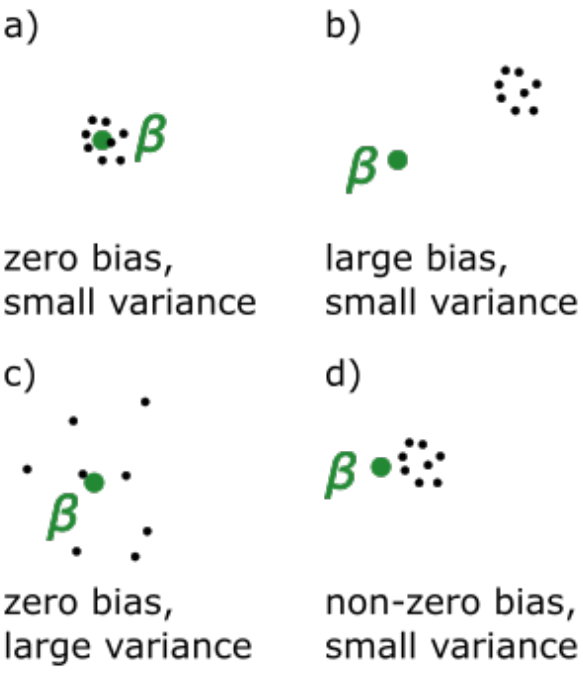
Related to the expected error of the estimate

Bias-variance trade-off

$$\text{Expected squared error} = \text{Variance} + \text{Bias}^2$$

Bias: difference between average estimated parameter and its true value

Variance: variation of estimated parameters on different data sets (i.e., sensitivity to the particular set, compromises stability, generalisability)



Bias-Variance trade-off

How can we improve the performance of the Least Squares estimator? We have:

$$\mathbb{E}\big[(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)^T\big] = \mathbb{E}\big[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \underbrace{\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T}_{\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T]=\sigma^2\mathbf{I}} \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\big] = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$$

$(\mathbf{X}^T\mathbf{X})$ can be badly conditioned due to dependencies among features, poor sampling

LS is unbiased!

$$\|\mathbb{E}[\boldsymbol{\beta}^\star] - \boldsymbol{\beta}\| = 0$$

For improvements, one needs to reduce the variance

Bias-Variance trade-off

How can we improve the performance of the Least Squares estimator? We have:

$$\mathbb{E}\left[(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)^T\right] = \mathbb{E}\left[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \underbrace{\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T}_{\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T]=\sigma^2\mathbf{I}} \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\right] = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$$

$(\mathbf{X}^T\mathbf{X})$ can be badly conditioned due to dependencies among features, poor sampling

LS is unbiased!

$$\|\mathbb{E}[\boldsymbol{\beta}^\star] - \boldsymbol{\beta}\| = 0$$

For improvements, one needs to reduce the variance

Strategies:

1. **Best-subset regression**: finds the subset of size k<p with the smallest regression error. The model has a reduced number of features, i.e. it's sparser.

Bias-Variance trade-off

How can we improve the performance of the Least Squares estimator? We have:

$$\mathbb{E}\left[(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)(\boldsymbol{\beta} - \boldsymbol{\beta}^\star)^T\right] = \mathbb{E}\left[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \underbrace{\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T}_{\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T]=\sigma^2\mathbf{I}} \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\right] = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$$

$(\mathbf{X}^T\mathbf{X})$ can be badly conditioned due to dependencies among features, poor sampling

LS is unbiased!

$$\|\mathbb{E}[\boldsymbol{\beta}^\star] - \boldsymbol{\beta}\| = 0$$

→ For improvements, one needs to reduce the variance

Strategies:

1. **Best-subset regression**: finds the subset of size k<p with the smallest regression error. The model has a reduced number of features, i.e. it's sparser.

2. **Shrinkage methods**: Ridge and LASSO regression.
Main idea: modify the loss function to make the size of learnt parameters small.

Gain: one has an estimator at lower variance, but biased;
one enforces a continuous version of spasity.

Ridge Regression

In Ridge regression, one adds a term penalising large values of the parameters

$$L_{\text{RIDGE}}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2$$

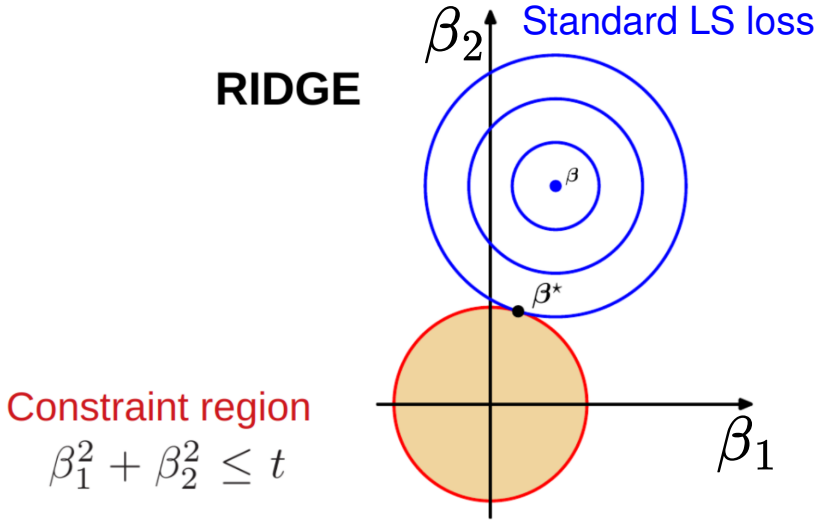
$\lambda > 0$ is the *penalty term*

$\|\boldsymbol{\beta}\|^2 = \sum_{i=1}^{p} |\beta_i|^2$

The solution is found by:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2 \quad \Leftrightarrow \quad \min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \text{ subject to } \|\boldsymbol{\beta}\|^2 \leq t$$

This equivalence allows us to rationalise a bit more geometrically the solution:



It can be obtained analytically:

$$\boldsymbol{\beta}^\star_{\text{RIDGE}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

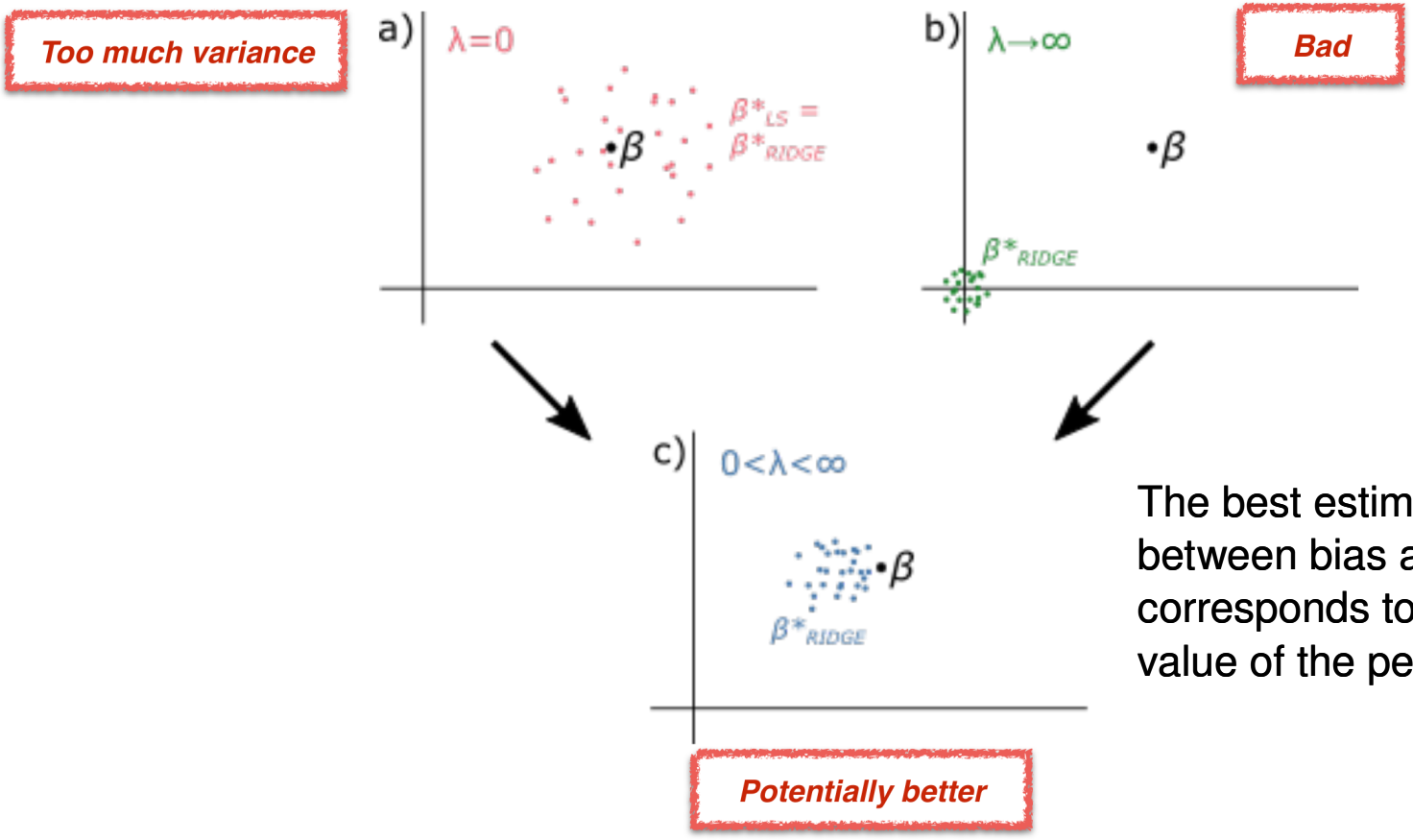
Ridge Regression

In Ridge regression, one adds a term penalising large values of the parameters

$$L_{\mathrm{RIDGE}}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2$$

$\lambda > 0$ is the *penalty term*

$\|\boldsymbol{\beta}\|^2 = \sum_{i=1}^{p} |\beta_i|^2$



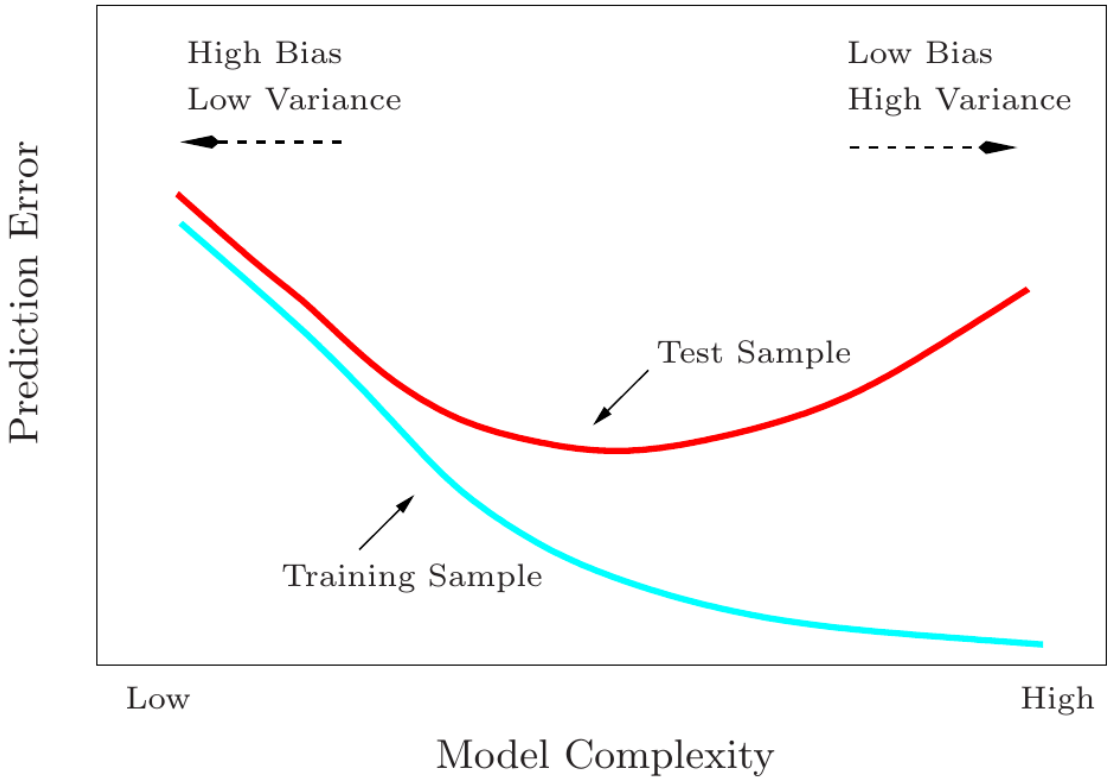
The best estimator strikes a balance between bias and variance, and corresponds to an intermediate value of the penalty term.

How do we set the right value of λ ?

λ is a hyper-parameter - we need a hyperparametric search

Region of underfitting

Region of overfitting



or 'Model capacity' (from flexible parametrization)

Large $\longleftarrow \lambda \longrightarrow$ Small

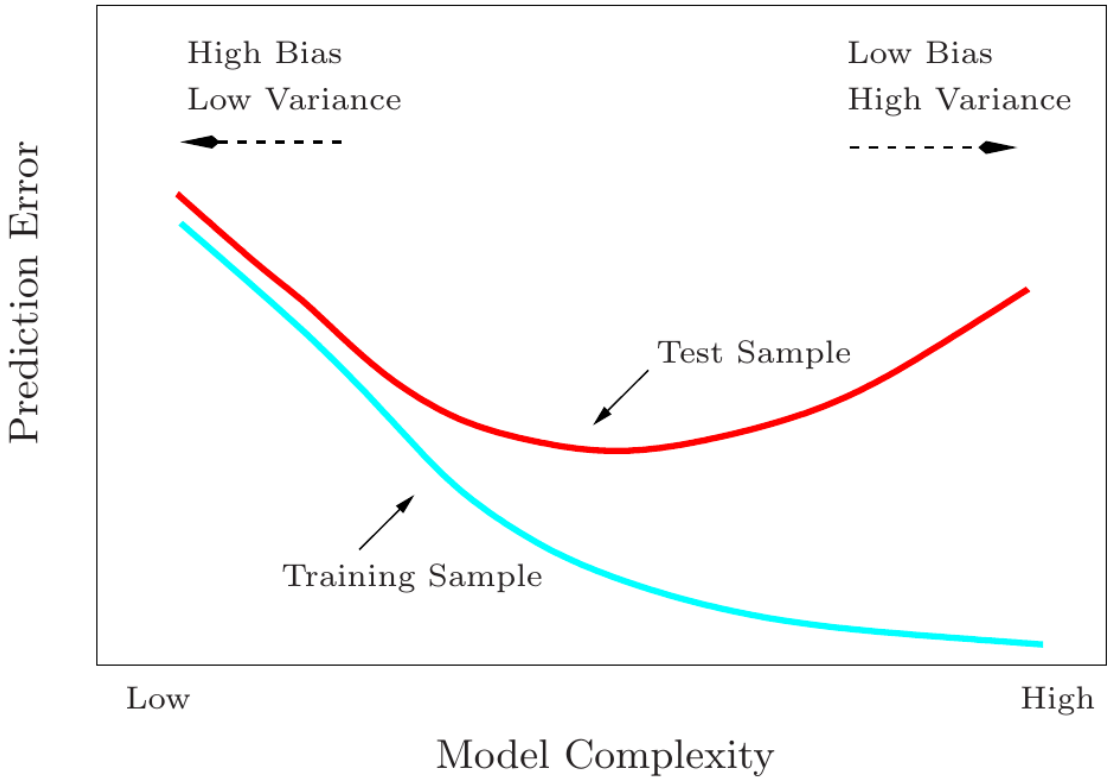
Plays the role of regularisation
(control overfitting via a penalty)

How do we set the right value of λ ?

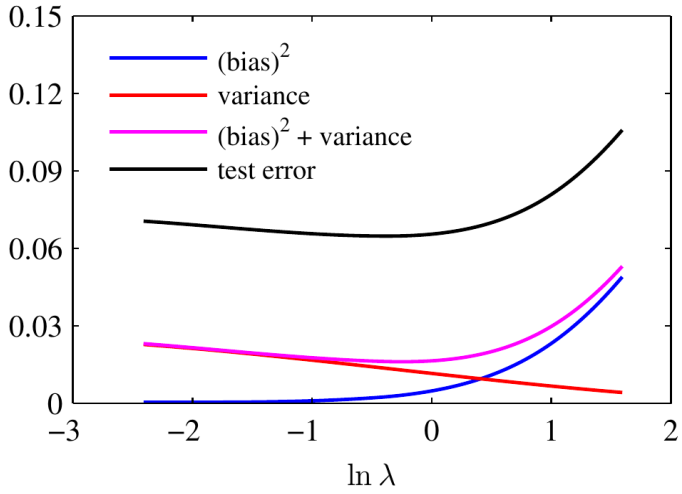
λ is a hyper-parameter - we need a hyperparametric search

Region of underfitting

Region of overfitting



or 'Model capacity' (from flexible parametrization)



Example on RIDGE regression (from Bishop, Pattern Recognition and Machine Learning, chap. 3)

Large

Small

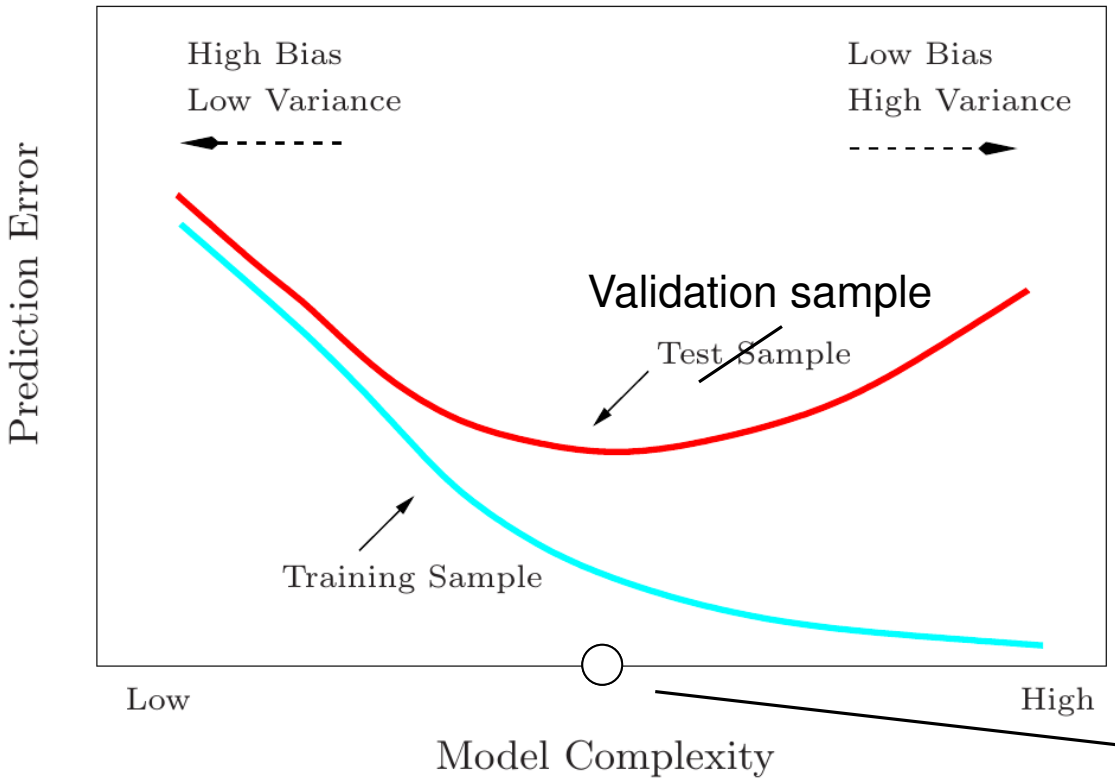
λ

How do we set the right value of λ ?

λ is a hyper-parameter - we need a hyperparametric search

Region of underfitting

Region of overfitting



Optimal value

or 'Model capacity' (from flexible parametrization)

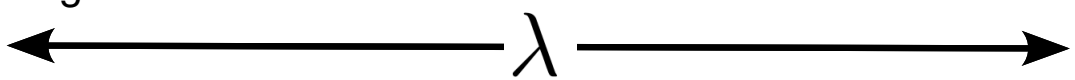
Cross-validation (next lecture)

Model selection
(hyperparametric search)

Model assessment
(out-of-sample performance)

Large

Small



LASSO Regression

Also in Lasso regression, one adds a term penalising large values of the parameters

$$L_{\text{LASSO}}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1$$

$\lambda > 0$ is the *penalty term*

$$\|\boldsymbol{\beta}\|_1 = \sum_{i=1}^{p} |\beta_i|$$

Compared to ridge, it does **not** have an analytical solution, but it is solvable numerically with standard convex optimisation techniques (you will see one in the coding notebook)

LASSO Regression

Also in Lasso regression, one adds a term penalising large values of the parameters

$$L_{\text{LASSO}}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1$$

$\lambda > 0$ is the *penalty term*

$$\|\boldsymbol{\beta}\|_1 = \sum_{i=1}^{p} |\beta_i|$$

Compared to ridge, it does **not** have an analytical solution, but it is solvable numerically with standard convex optimisation techniques (you will see one in the coding notebook)

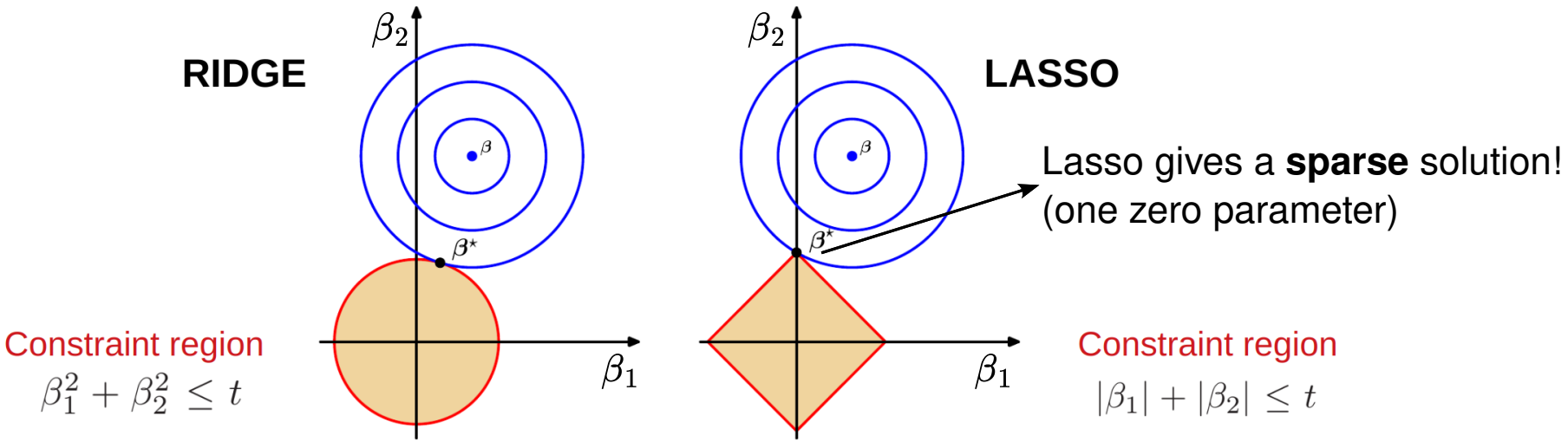
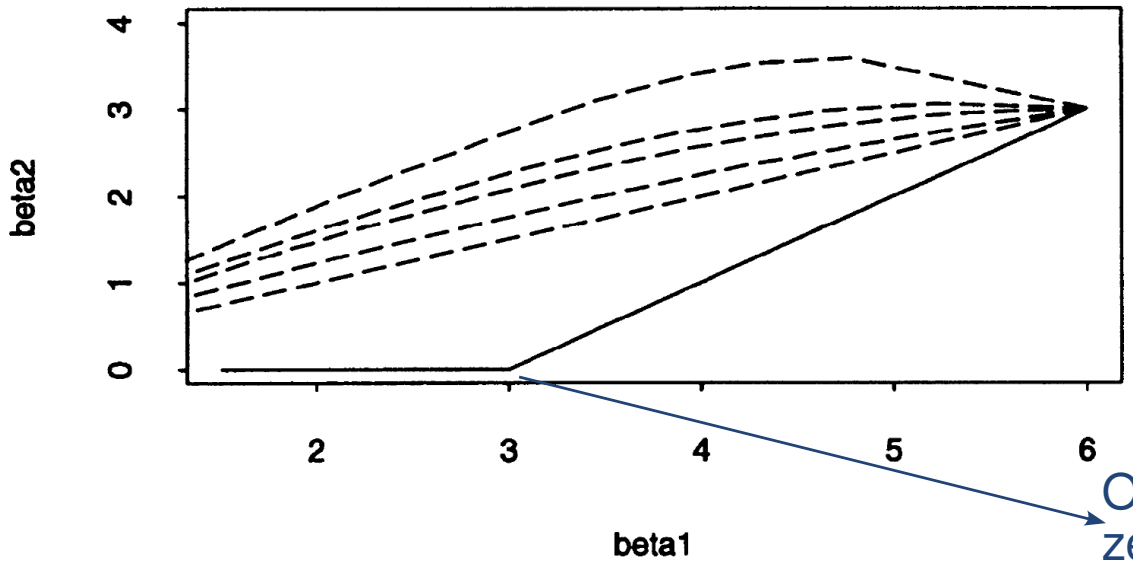


Figure adapted from Bishop, Pattern Recognition and Machine Learning, chap. 3

LASSO Regression

Compared to ridge, **LASSO induces a stronger sparsity**,
as was illustrated already in the original article introducing it:

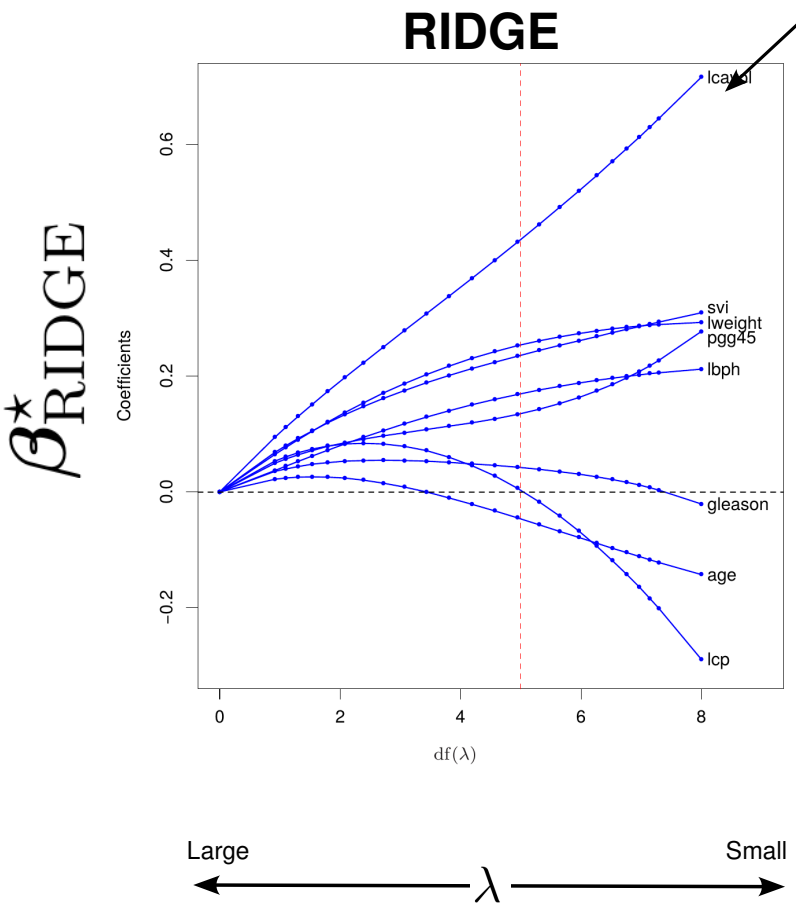


From Tibshirani R,
J. R. Statistic. Soc. B (1996),
see additional reading

Fig. 4. Lasso (———) and ridge regression (– – – –) for the two-predictor example: the curves show the (β_1, β_2) pairs as the bound on the lasso or ridge parameters is varied; starting with the bottom broken curve and moving upwards, the correlation ρ is 0, 0.23, 0.45, 0.68 and 0.90

Example of linear regression on a clinical dataset: the outcome is a marker of prostate cancer, the predictors are patients' clinical characteristics (from Hastie, Tibshirani, Friedman, Elements of statistical learning, chap. 3

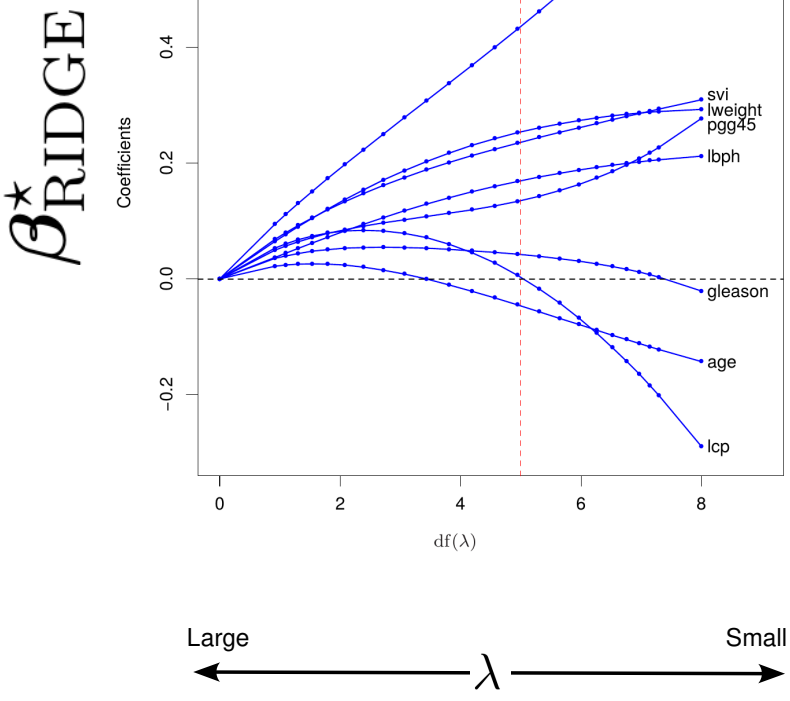
One coefficient per predictor (clinical charactertistic)



Example of linear regression on a clinical dataset: the outcome is a marker of prostate cancer, the predictors are patients' clinical characteristics (from Hastie, Tibshirani, Friedman, Elements of statistical learning, chap. 3

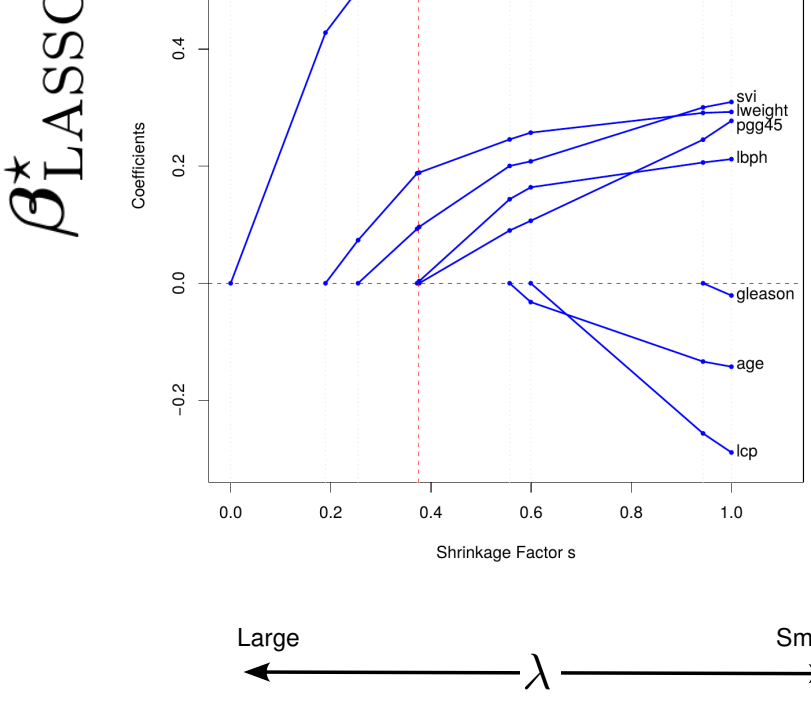
One coefficient per predictor (clinical charactertistic)

RIDGE



Large ← λ → Small

LASSO



Large ← λ → Small

Effective subset selection

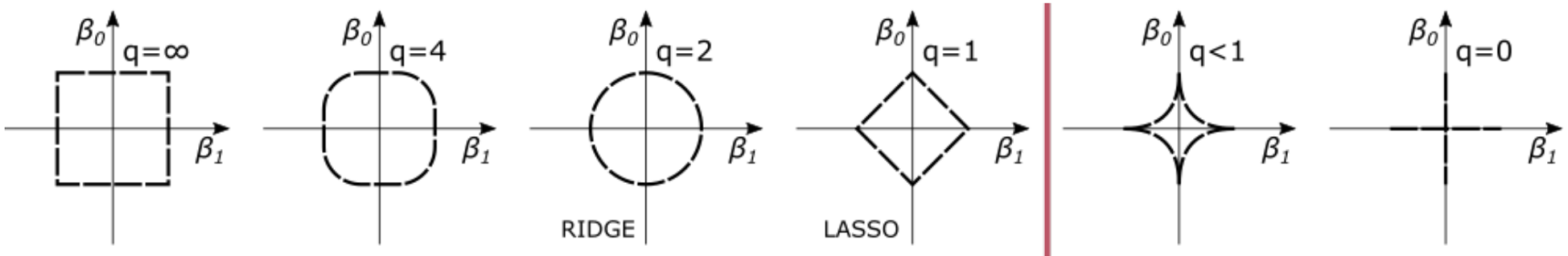
Generalisations

Elastic nets: combine benefits of both, but have an additional hyperparameter

$$L_{\text{EN}}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\left[\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|^2\right]$$

General variations involve regularisation terms containing the l_q-norm:

$$L_q(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_q^q \qquad \|\boldsymbol{\beta}\|_q^q = \sum_{i=1}^{p} |\beta_i|^q$$



Convex sets

Non-convex sets

Optimisation is doable

Optimisation is difficult

Summary I: regression & LS

Supervised learning: the basic process schematically is

(1) Start with data $(x^{(i)}, y^{(i)}), \quad i = 1, \ldots, N.$

(2) Decide: Classification or regression?

(3) Define a *loss function* $L(y, f(x))$

(4) Minimise the *mean sample loss*: $E(L) = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)}))$

(5) Try to also achieve a reasonable *expected test loss*: $E\big(L(y^{\text{in}}, f(x^{\text{in}}))\big)$

Summary I: regression & LS

Supervised learning: the basic process schematically is

(1) Start with data $(x^{(i)}, y^{(i)}), \quad i = 1, \ldots, N.$

(2) Decide: Classification or regression?

(3) Define a *loss function* $L(y, f(x))$

(4) Minimise the *mean sample loss*: $E(L) = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)}))$

(5) Try to also achieve a reasonable *expected test loss*: $E\big(L(y^{\text{in}}, f(x^{\text{in}}))\big)$

1. Linear Regression is the simplest form of regression, i.e. the supervised learning task of learning from data a model to predict a continuous outcome.

Summary I: regression & LS

Supervised learning: the basic process schematically is

(1) Start with data $(x^{(i)}, y^{(i)}), \quad i = 1, \ldots, N.$

(2) Decide: Classification or regression?

(3) Define a *loss function* $L(y, f(x))$

(4) Minimise the *mean sample loss*: $E(L) = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)}))$

(5) Try to also achieve a reasonable *expected test loss*: $E\big(L(y^{\text{in}}, f(x^{\text{in}}))\big)$

1. Linear Regression is the simplest form of regression, i.e. the supervised learning task of learning from data a model to predict a continuous outcome.

2. The vector of model's parameters is learnt from the data through LS, i.e., minimising the MSE (loss) between true/predicted outcomes (LS solution). There is a formula for the LS solution, but in general one solves the loss-optimisation problem numerically, via gradient descent methods.

Summary I: regression & LS

Supervised learning: the basic process schematically is

(1) Start with data $(x^{(i)}, y^{(i)}), \quad i = 1, \ldots, N.$

(2) Decide: Classification or regression?

(3) Define a *loss function* $L(y, f(x))$

(4) Minimise the *mean sample loss*: $E(L) = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)}))$

(5) Try to also achieve a reasonable *expected test loss*: $E\big(L(y^{\text{in}}, f(x^{\text{in}}))\big)$

1. Linear Regression is the simplest form of regression, i.e. the supervised learning task of learning from data a model to predict a continuous outcome.

2. The vector of model's parameters is learnt from the data through LS, i.e., minimising the MSE (loss) between true/predicted outcomes (LS solution). There is a formula for the LS solution, but in general one solves the loss-optimisation problem numerically, via gradient descent methods.

3. The LS estimator is unbiased, but can have large variance.

Summary II: shrinkage meth.

1. Shrinkage methods (Ridge and LASSO) modify the linear-regression loss adding a term to penalise large values of the learnt parameters.

Summary II: shrinkage meth.

1. Shrinkage methods (Ridge and LASSO) modify the linear-regression loss adding a term to penalise large values of the learnt parameters.

2. Shrinkage methods help improve generalisability: shrinking or setting some coefficients to zero (sparsity) controls overfitting.

Summary II: shrinkage meth.

1. Shrinkage methods (Ridge and LASSO) modify the linear-regression loss adding a term to penalise large values of the learnt parameters.

2. Shrinkage methods help improve generalisability: shrinking or setting some coefficients to zero (sparsity) controls overfitting.

3. LASSO induces stronger sparsity, which might favour **interpretability**: With a large number of predictors, it might be difficult to inspect and understand which ones determine the strongest effects. Parsimony of parameters and predictors might help identify e.g. the main determinants of a certain outcome, so the model is more interpretable.

Summary II: shrinkage meth.

1. Shrinkage methods (Ridge and LASSO) modify the linear-regression loss adding a term to penalise large values of the learnt parameters.

2. Shrinkage methods help improve generalisability: shrinking or setting some coefficients to zero (sparsity) controls overfitting.

3. LASSO induces stronger sparsity, which might favour **interpretability**: With a large number of predictors, it might be difficult to inspect and understand which ones determine the strongest effects. Parsimony of parameters and predictors might help identify e.g. the main determinants of a certain outcome, so the model is more interpretable.

4. The penalty coefficient should be optimised in a hyperparametric search.

Summary II: shrinkage meth.

1. Shrinkage methods (Ridge and LASSO) modify the linear-regression loss adding a term to penalise large values of the learnt parameters.

2. Shrinkage methods help improve generalisability: shrinking or setting some coefficients to zero (sparsity) controls overfitting.

3. LASSO induces stronger sparsity, which might favour **interpretability**: With a large number of predictors, it might be difficult to inspect and understand which ones determine the strongest effects. Parsimony of parameters and predictors might help identify e.g. the main determinants of a certain outcome, so the model is more interpretable.

4. The penalty coefficient should be optimised in a hyperparametric search.

Regularisations, bias-variance trade-off, hyperparametric search are more general than linear models and regression (e.g. k-NN, deep neural networks)!