Lecture 5. Optimisation. Regularisation

COMP90051 Statistical Machine Learning

Semester 2, 2017
Lecturer: Andrey Kan

This lecture

- Iterative optimisation
  - Loss functions
  - Coordinate descent
  - Gradient descent
- Regularisation
  - Model complexity
  - Constrained modelling
  - Bias-variance trade-off

Iterative Optimisation

A very brief summary of a few basic optimisation methods

Supervised learning*

1. Assume a model (e.g., linear model)
   * Denote parameters of the model as $\boldsymbol{\theta}$
   * Model predictions are $\hat{f}(\boldsymbol{x}, \boldsymbol{\theta})$
2. Choose a way to measure discrepancy between predictions and training data
   * E.g., sum of squared residuals $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$
3. Training = parameter estimation = optimisation

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \, L(data, \boldsymbol{\theta})$$

*This is the setup of what's called *frequentist supervised learning*. A different view on parameter estimation/training will be presented later in the subject.

Loss functions: Measuring discrepancy

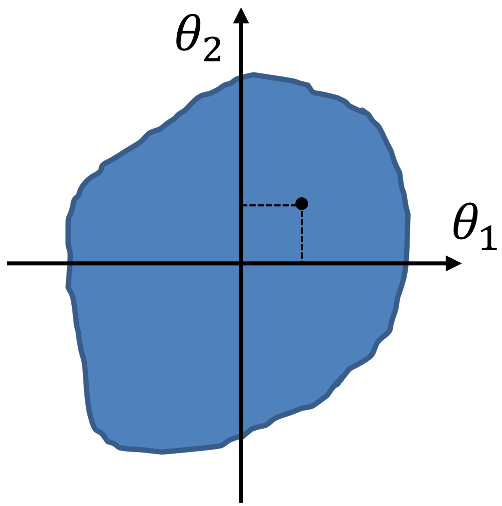
- For a single training example the discrepancy between prediction and label is measured using a *loss function*

- Examples:
  - squared loss $l_{sq} = \left(y - \hat{f}(\boldsymbol{x}, \boldsymbol{\theta})\right)^2$
  - absolute loss $l_{abs} = \left|y - \hat{f}(\boldsymbol{x}, \boldsymbol{\theta})\right|$
  - Perceptron loss (next lecture)
  - Hinge loss (later in the subject)

Solving optimisation problems

- Analytic (aka closed form) solution
  - Known only in limited number of cases
  - Use necessary condition: $\frac{\partial L}{\partial \theta_1} = \cdots = \frac{\partial L}{\partial \theta_p} = 0$
- Approximate iterative solution
  1. Initialisation: choose starting guess $\boldsymbol{\theta}^{(1)}$, set $i = 1$
  2. Update: $\boldsymbol{\theta}^{(i+1)} \leftarrow SomeRule\left[\boldsymbol{\theta}^{(i)}\right]$, set $i \leftarrow i + 1$
  3. Termination: decide whether to Stop
  4. Go to Step 2
  5. Stop: return $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$

Finding the deepest point

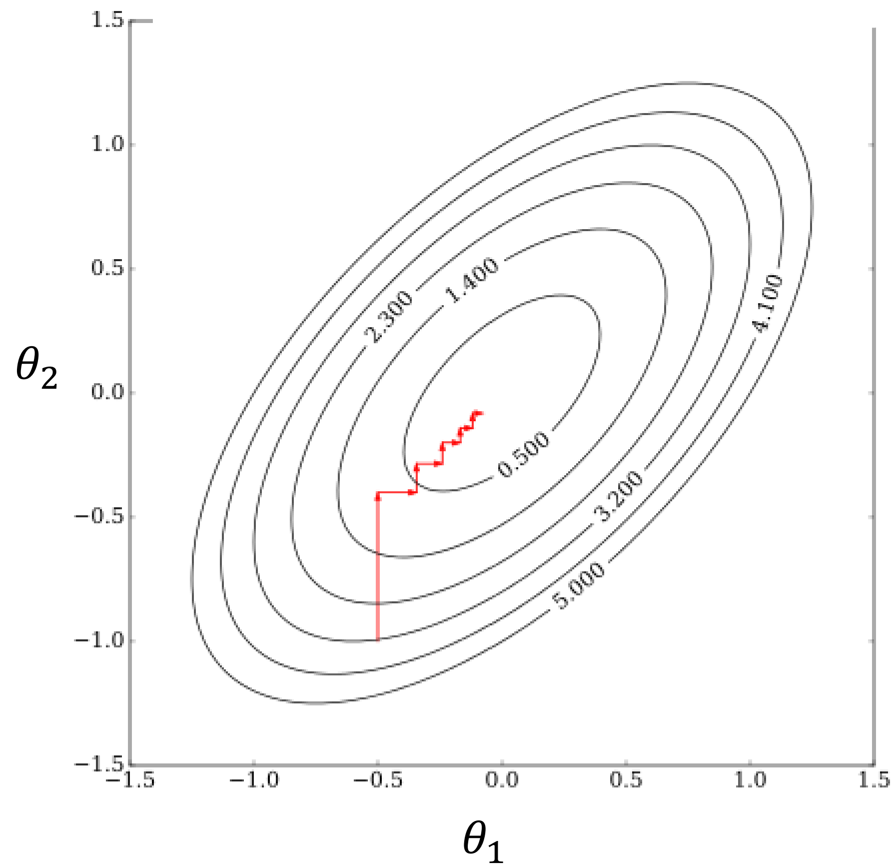


- In this example, a model has 2 parameters
- Each location corresponds to a particular combination of parameter values
- Depth indicates discrepancy between model using those values and data

Coordinate descent

- Suppose $\boldsymbol{\theta} = [\theta_1, \dots, \theta_K]'$

1. Choose $\boldsymbol{\theta}^{(1)}$ and some T
2. For i from 1 to T*
   1. $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)}$
   2. For j from 1 to K
      1. Fix components of $\boldsymbol{\theta}^{(i+1)}$, expect j-th component
      2. Find $\hat{\theta}_j^{(i+1)}$ that minimises $L\left(\theta_j^{(i+1)}\right)$
      3. Update j-th component of $\boldsymbol{\theta}^{(i+1)}$
3. Return $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$



*Other stopping criteria can be used

Wikimedia Commons. Author: Nicoguaro (CC4)

Gradient

- Gradient (at point $\boldsymbol{\theta}$) is defined as $\left[\frac{\partial L}{\partial \theta_1}, \dots, \frac{\partial L}{\partial \theta_p}\right]'$ computed at point $\boldsymbol{\theta}$
- One can show that gradient points to the direction of maximal change of $L(\boldsymbol{\theta})$ when departing from point $\boldsymbol{\theta}$
- Shorthand notation
  - $\nabla L \stackrel{\text{def}}{=} \left[\frac{\partial L}{\partial \theta_1}, \dots, \frac{\partial L}{\partial \theta_p}\right]'$ computed at point $\boldsymbol{\theta}$
  - Here ∇ is the nabla symbol

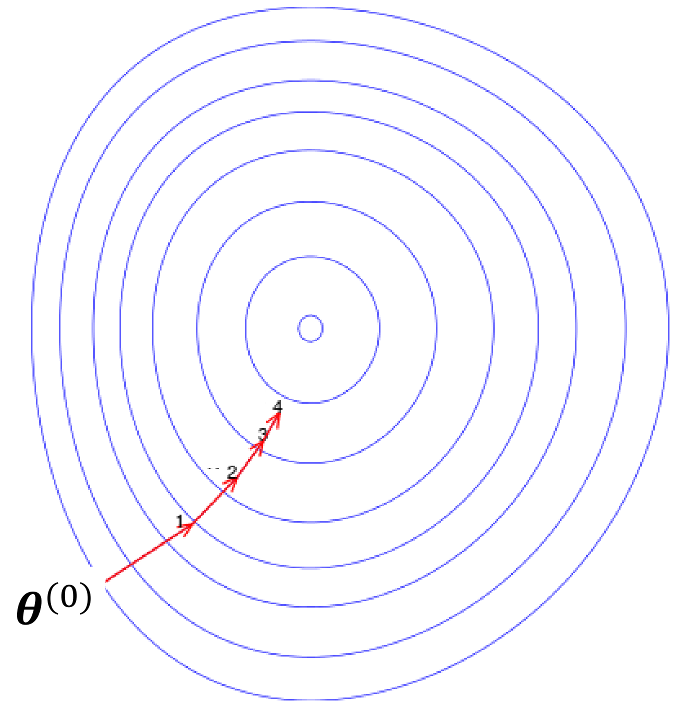
Harps, p. 984.

Gradient descent

1. Choose $\boldsymbol{\theta}^{(1)}$ and some T
2. For i from 1 to T*
   1. $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \nabla L(\boldsymbol{\theta}^{(i)})$
3. Return $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$

- Note: η is dynamically updated in each step

We assume L is differentiable



*Other stopping criteria can be used

Wikimedia Commons. Authors: Olegalexandrov, Zerodamage

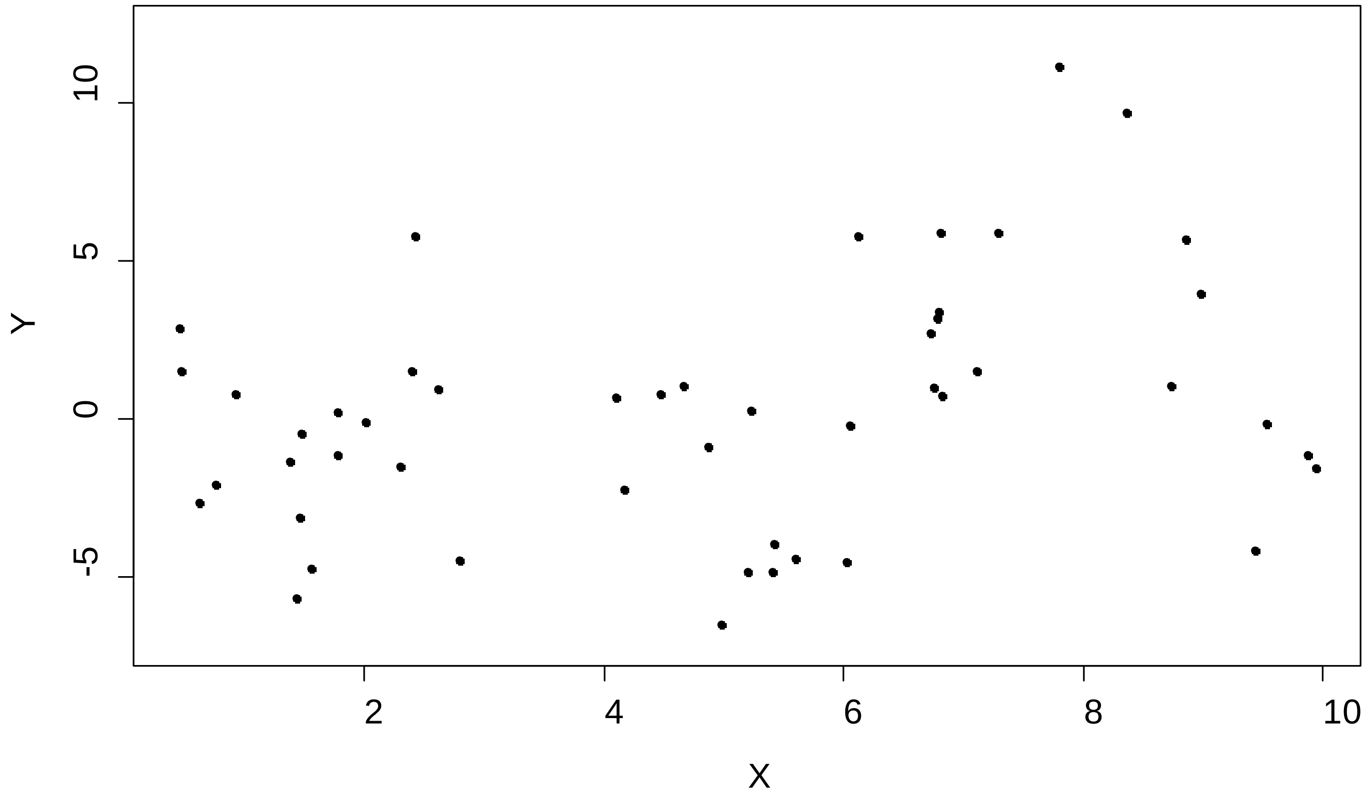
Regularisation

Process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting (*Wikipedia*)

Previously: Regularisation

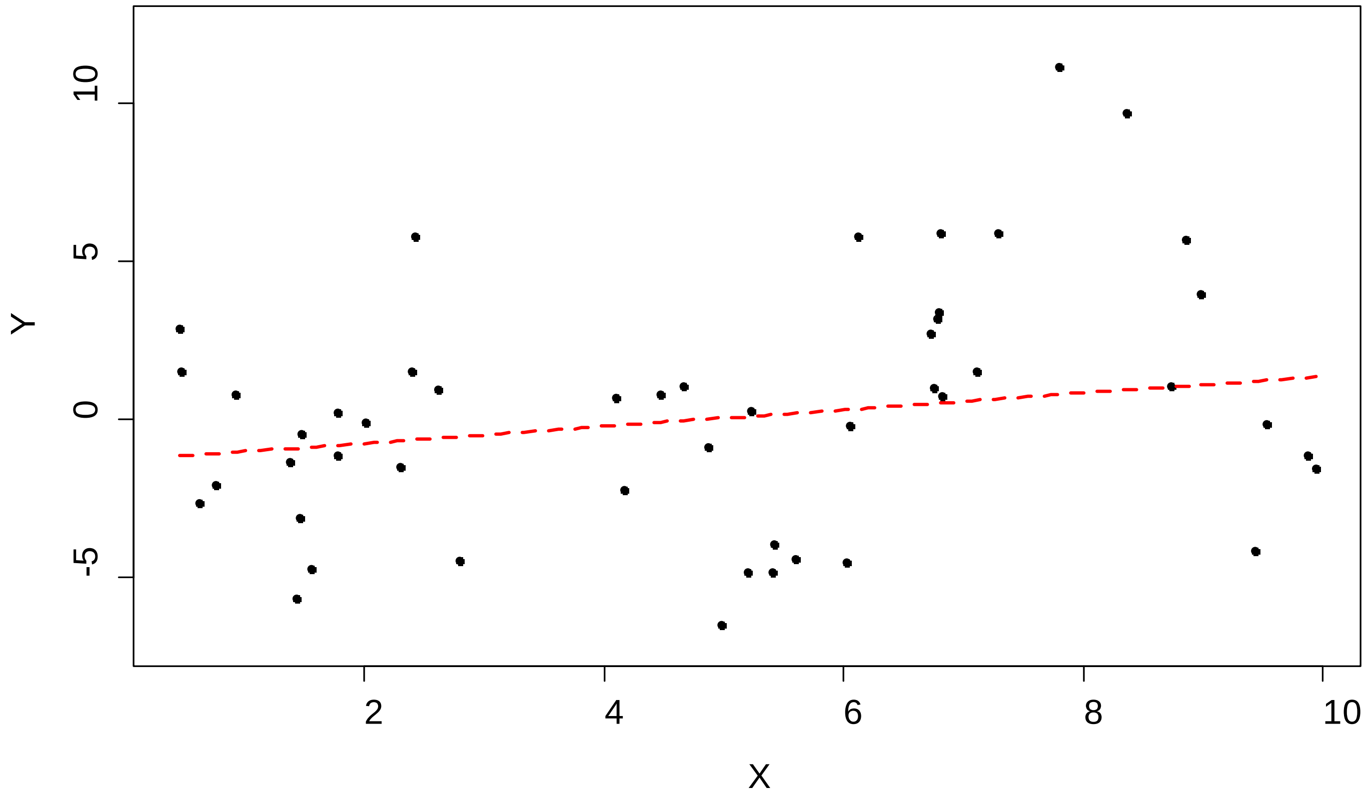
- Major technique, common in Machine Learning
- Addresses one or more of the following related problems
  * Avoid ill-conditioning
  * Introduce prior knowledge
  * Constrain modelling
- This is achieved by augmenting the objective function
- <u>Not just for linear methods</u>.

Example regression problem



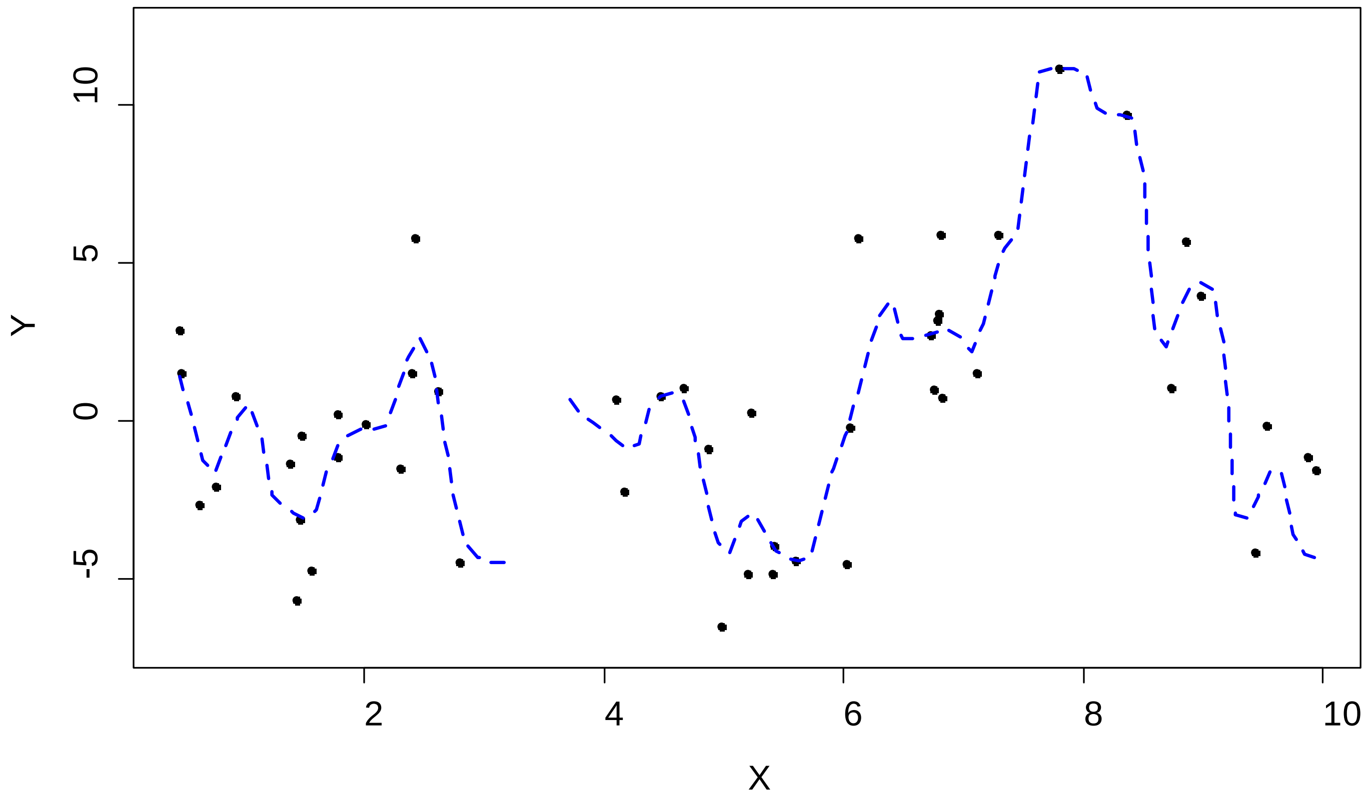
How complex a model should we use?

Underfitting (linear regression)



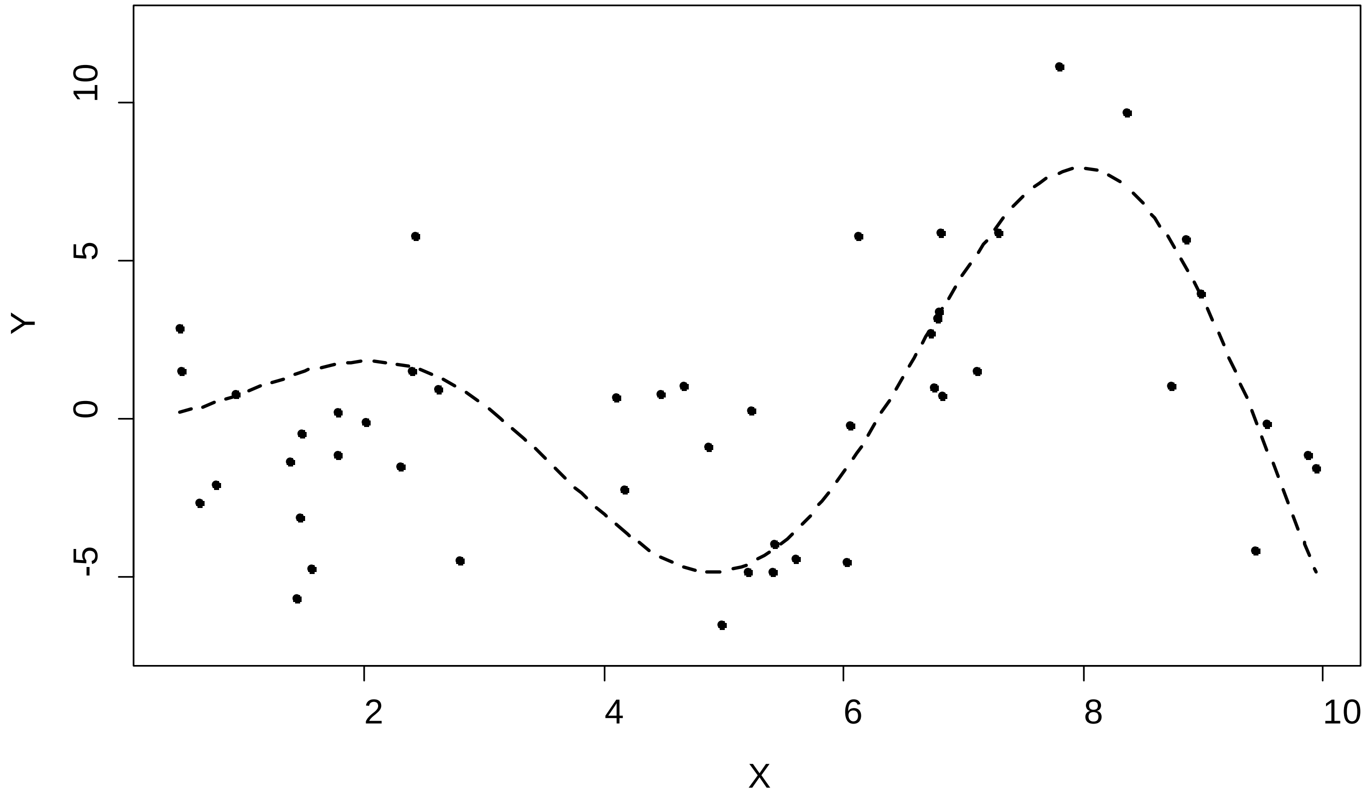
Model class Θ can be **too simple** to possibly fit true model.

Overfitting (non-parametric smoothing)



Model class Θ can be **so complex** it can fit true model + noise

Actual model ($x \sin x$)



The **right model class** Θ will sacrifice some training error, for test error.

How to “vary” model complexity

- Method 1: Explicit model selection
- Method 2: Regularisation
- Usually, method 1 can be viewed a special case of method 2

1. Explicit model selection

- Try different classes of models. Example, try polynomial models of various degree d (linear, quadratic, cubic, …)
- Use held out validation (cross validation) to select the model

1. Split training data into D_{train} and $D_{validate}$ sets
2. For each degree d we have model f_d
   1. Train f_d on D_{train}
   2. Test f_d on $D_{validate}$
3. Pick degree $\hat{d}$ that gives the best test score
4. Re-train model $f_{\hat{d}}$ using all data

2. Vary complexity by regularisation

- Augment the problem:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \big(L(data, \boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta}) \big)$$

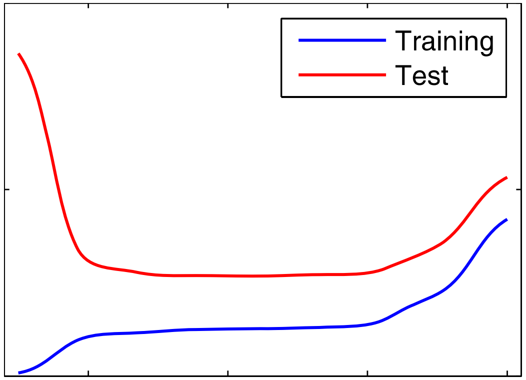
- E.g., ridge regression

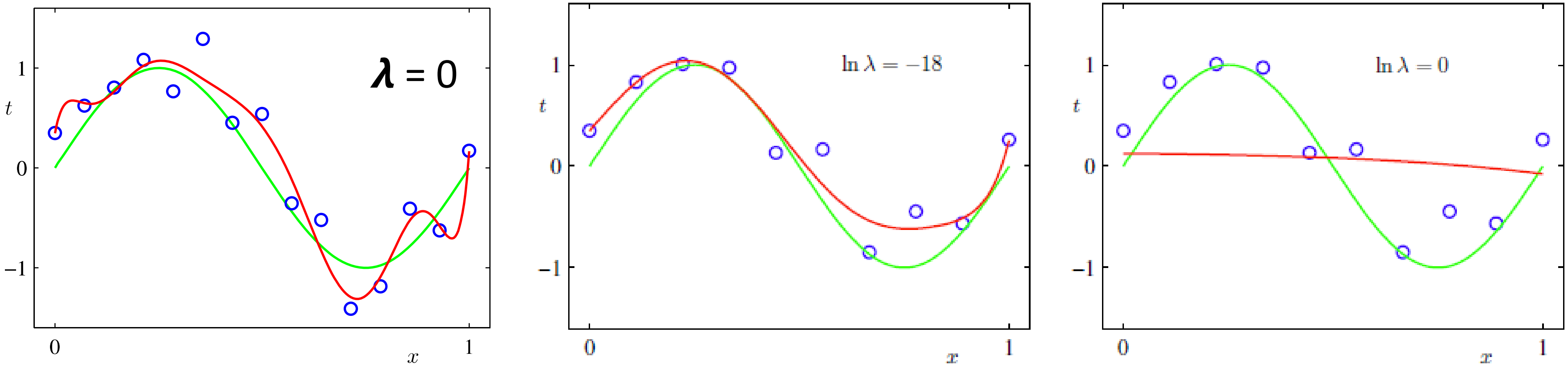
$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w} \in W}{\operatorname{argmin}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$$

- Note that $R(\boldsymbol{\theta})$ does not depend on data
- Use held out validation/cross validation to choose λ

Example: polynomial regression

- 9^{th} order polynomial regression
  * model of form
    $$\hat{f} = w_0 + w_1 x + \ldots + w_9 x^9$$
  * regularised with $\lambda\|\boldsymbol{w}\|_2^2$ term



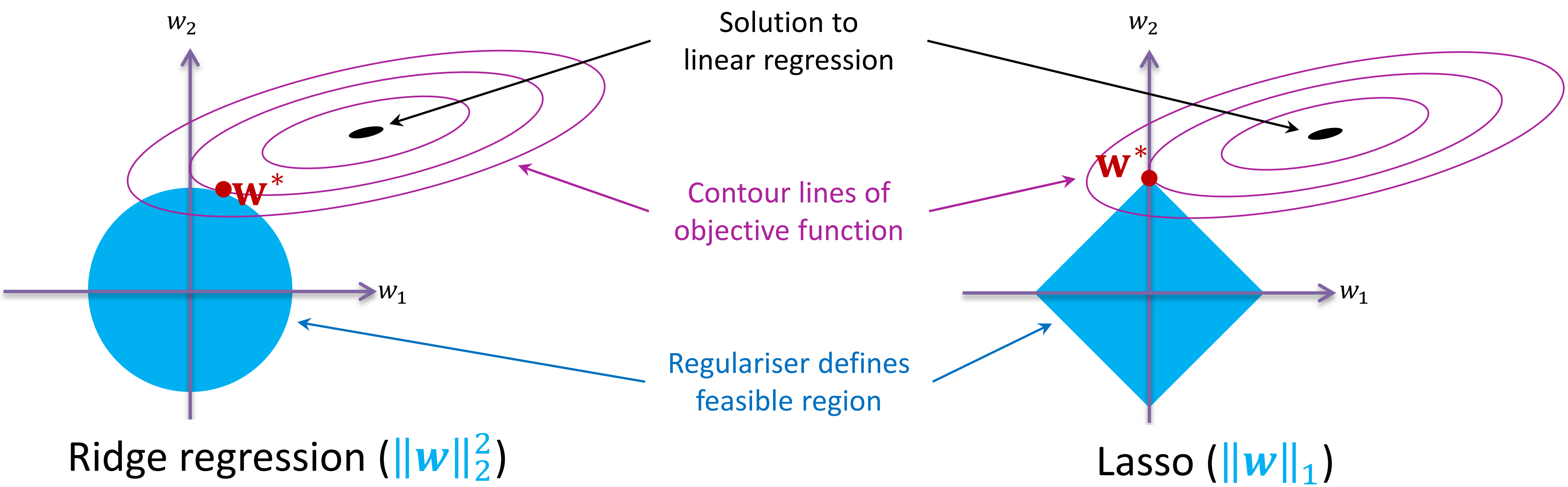


Figures from Bishop pp7-9,150

Regulariser as a constraint

- For illustrative purposes, consider a *modified problem*:

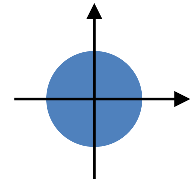
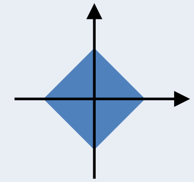
  minimise $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$ *subject to* $\|\boldsymbol{w}\|_2^2 \leq \lambda$ for $\lambda > 0$



- Lasso (L1 regularisation) encourages solutions to sit on the axes

  → Some of the weights are set to zero → Solution is sparse

Regularised linear regression

Algorithm	Minimises	Regulariser	Solution
Linear regression	$\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2$	None	$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ (if inverse exists)
Ridge regression	$\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2+\lambda\|\boldsymbol{w}\|_2^2$	L2 norm 	$(\mathbf{X}'\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$
Lasso	$\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2+\lambda\|\mathbf{w}\|_1$	L1 norm 	No closed-form, but solutions are sparse and suitable for high-dim data

Bias-variance trade-off

Analysis of relations between train error, test error and model complexity

Assessing generalisation capacity

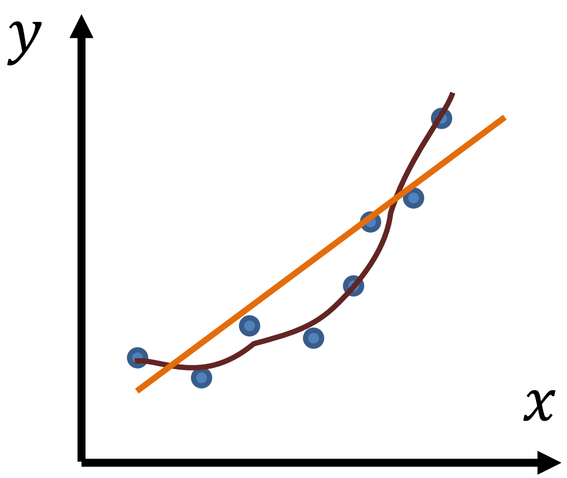
- Supervised learning: train the model on existing data, then make predictions on new data
    - Generalisation capacity of the model is an important consideration
- Training the model: minimisation of training error
- Generalisation capacity is captured by the test error
- Model complexity is a major factor that influences the ability of the model to generalise
- In this section, our aim is to explore relations between training error, test error and model complexity

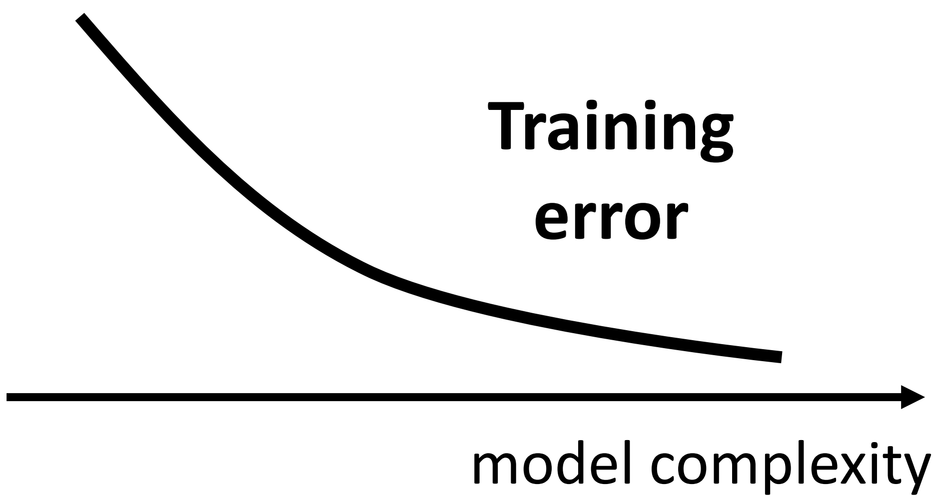
Training error

- Suppose training data is $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$
- Let $\hat{f}(\boldsymbol{x})$ denote predicted value for $\boldsymbol{x}$ from the model trained on D
- Let $l\left(y_i, \hat{f}(\boldsymbol{x}_i)\right)$ denote loss on the i-th training example
- Training error: $\frac{1}{n}\sum_{i=1}^{n} l\left(y_i, \hat{f}(\boldsymbol{x}_i)\right)$

Training error and model complexity

- More complex model → training error goes down
- Finite number of points → usually can reduce training error to 0 (is it always possible?)





Test error

- Assume $\mathcal{Y} = h(\boldsymbol{x}_0) + \varepsilon$
  * $\boldsymbol{x}_0$ is a fixed instance
  * $h(\boldsymbol{x})$ is an unknown true function
  * ε is noise, $\mathbb{E}[\varepsilon] = 0$ and $Var[\varepsilon] = \sigma^2$
- Treat training data as a random variable $\mathcal{D}$
  * Draw $D \sim \mathcal{D}$, train model on D, make predictions
  * Prediction $\hat{f}(\boldsymbol{x}_0)$ is a random variable, despite $\boldsymbol{x}_0$ fixed
- Test error for $\boldsymbol{x}_0$: $\mathbb{E}\left[l\left(\mathcal{Y}, \hat{f}(\boldsymbol{x}_0)\right)\right]$
  * The expectation is taken with respect to $\mathcal{D}$ and ε
  * $\mathcal{D}$ and ε are assumed to be independent

Bias-variance decomposition

- For the following analysis, we consider squared loss as an important special case

$$l\left(\mathcal{Y}, \hat{f}(\boldsymbol{x}_0)\right) = \left(\mathcal{Y} - \hat{f}(\boldsymbol{x}_0)\right)^2$$

- Lemma: Bias-Variance Decomposition

$$\mathbb{E}\left[l\left(\mathcal{Y}, \hat{f}(\boldsymbol{x}_0)\right)\right] = \left(\mathbb{E}[\mathcal{Y}] - \mathbb{E}[\hat{f}]\right)^2 + Var[\hat{f}] + Var[\mathcal{Y}]$$

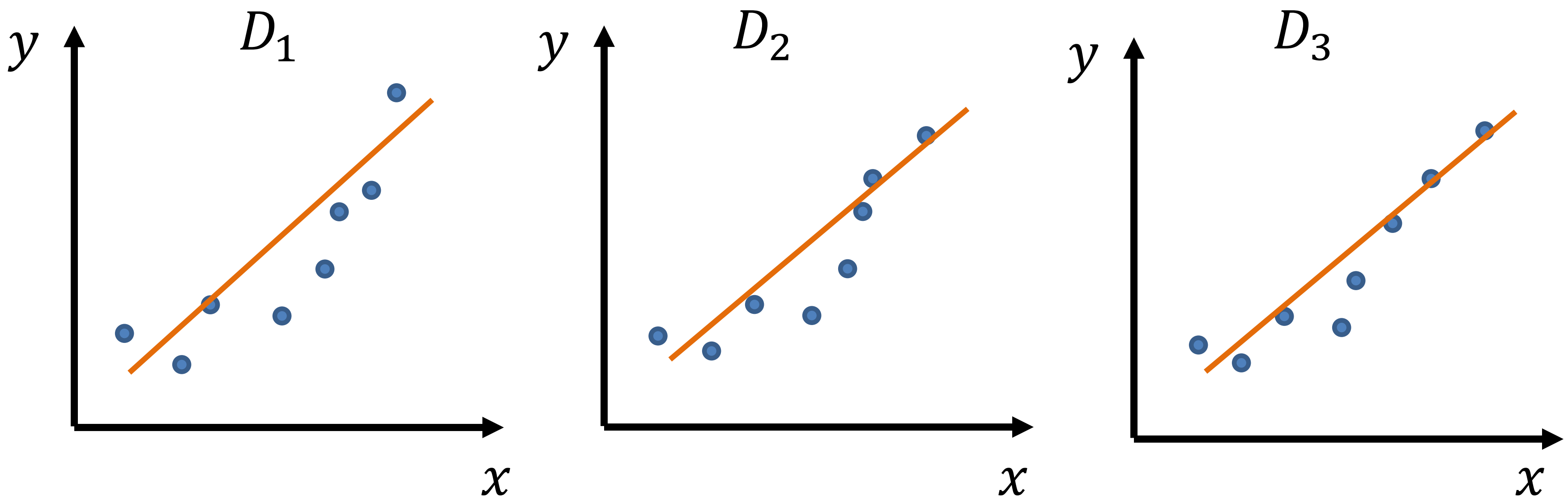
test error for $\boldsymbol{x}_0$ **(bias)^2** **variance** **irreducible error**

Decomposition proof sketch

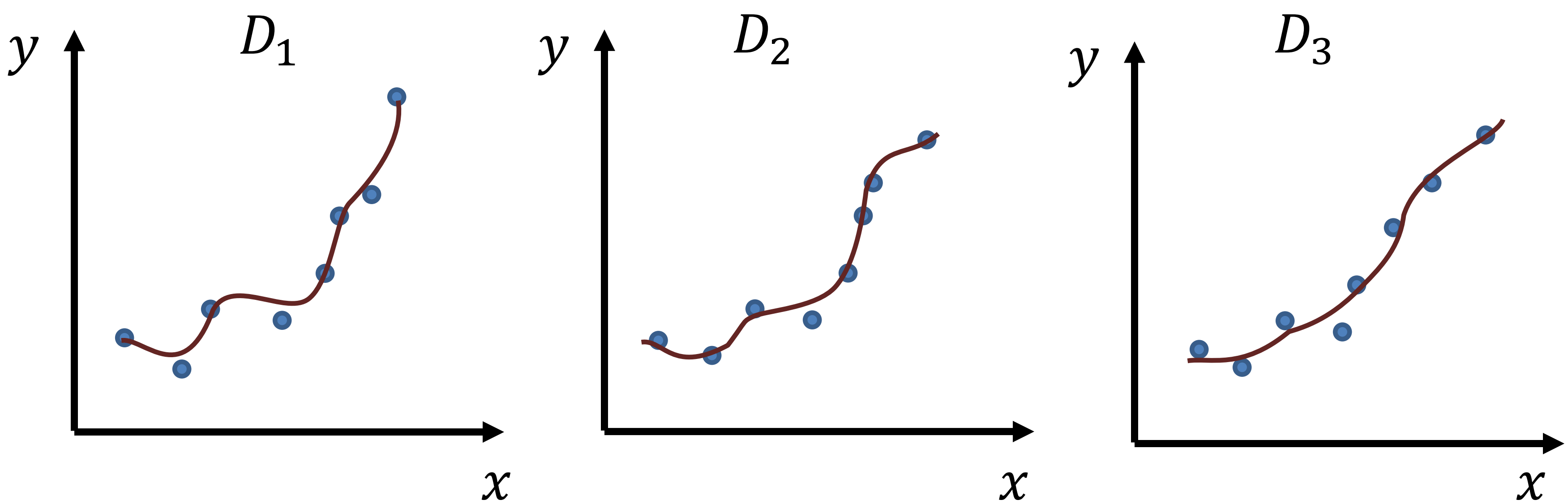
- Here $(\boldsymbol{x})$ is omitted to de-clutter notation
- $\mathbb{E}\left[\left(\mathcal{Y}-\hat{f}\right)^2\right] = \mathbb{E}\left[\mathcal{Y}^2 + \hat{f}^2 - 2\mathcal{Y}\hat{f}\right]$
- $= \mathbb{E}[\mathcal{Y}^2] + \mathbb{E}\left[\hat{f}^2\right] - \mathbb{E}\left[2\mathcal{Y}\hat{f}\right]$
- $= Var[\mathcal{Y}] + \mathbb{E}[\mathcal{Y}]^2 + Var\left[\hat{f}\right] + \mathbb{E}\left[\hat{f}\right]^2 - 2\mathbb{E}[\mathcal{Y}]\mathbb{E}\left[\hat{f}\right]$
- $= Var[\mathcal{Y}] + Var\left[\hat{f}\right] + \left(\mathbb{E}[\mathcal{Y}]^2 - 2\mathbb{E}[\mathcal{Y}]\mathbb{E}\left[\hat{f}\right] + \mathbb{E}\left[\hat{f}\right]^2\right)$
- $= Var[\mathcal{Y}] + Var\left[\hat{f}\right] + \left(\mathbb{E}[\mathcal{Y}] - \mathbb{E}\left[\hat{f}\right]\right)^2$

* Green slides are non-examinable

Training data as a random variable

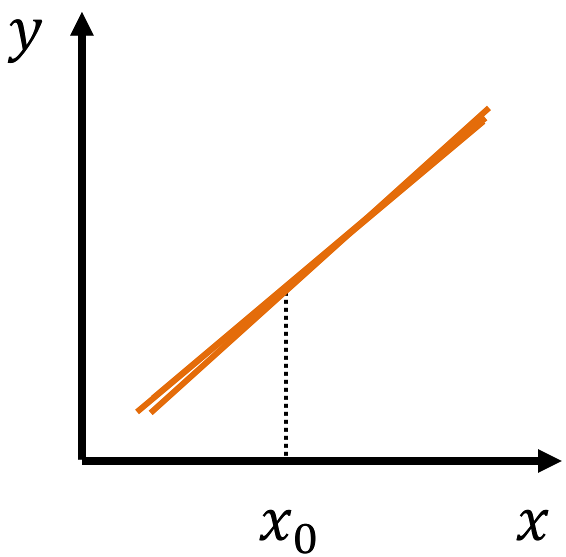


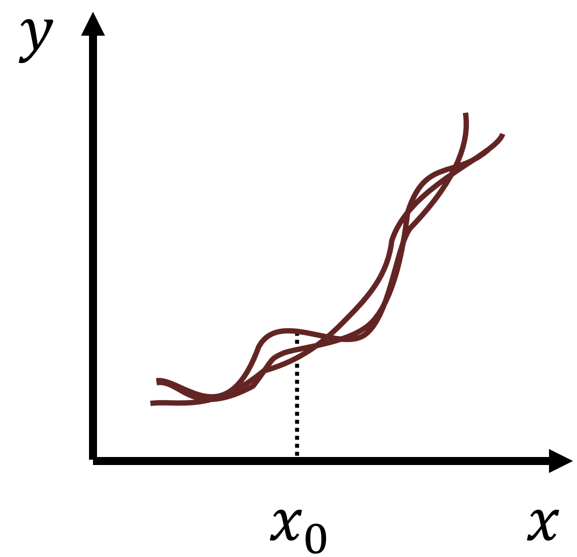
Training data as a random variable



Model complexity and variance

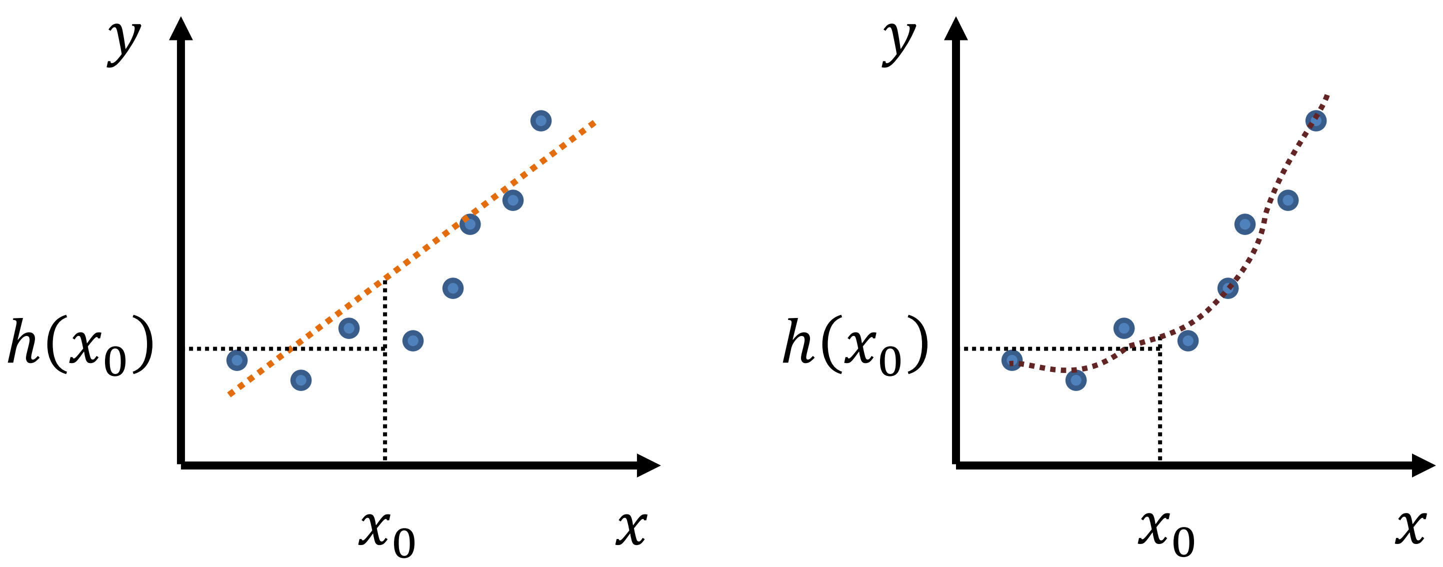
- simple model ➔ low variance
- complex model ➔ high variance





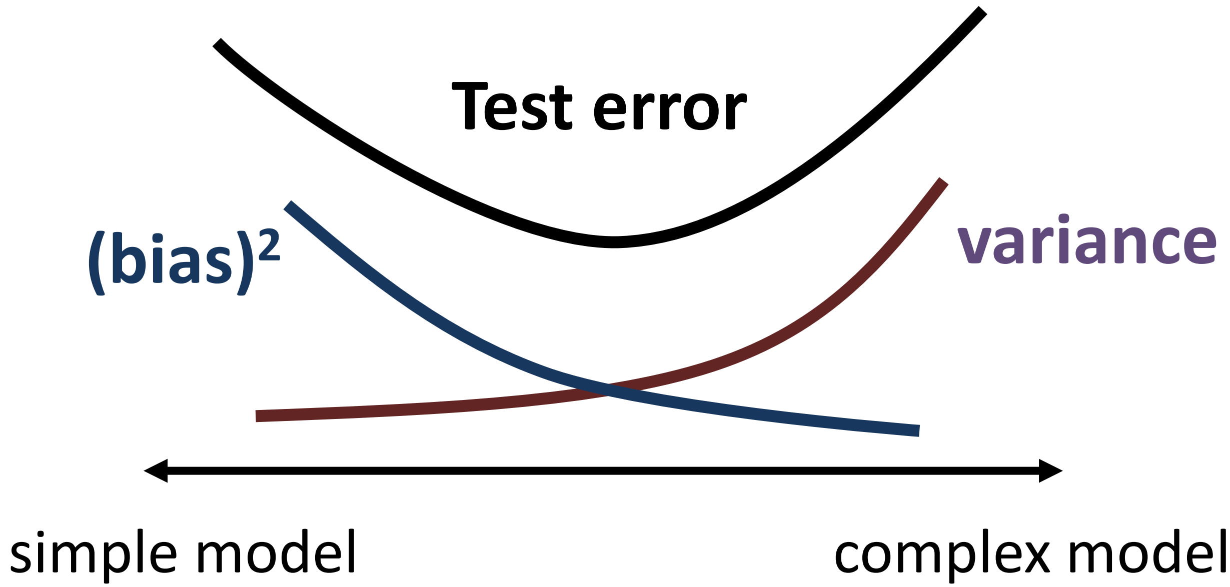
Model complexity and bias

- simple model ➔ high bias
- complex model ➔ low bias

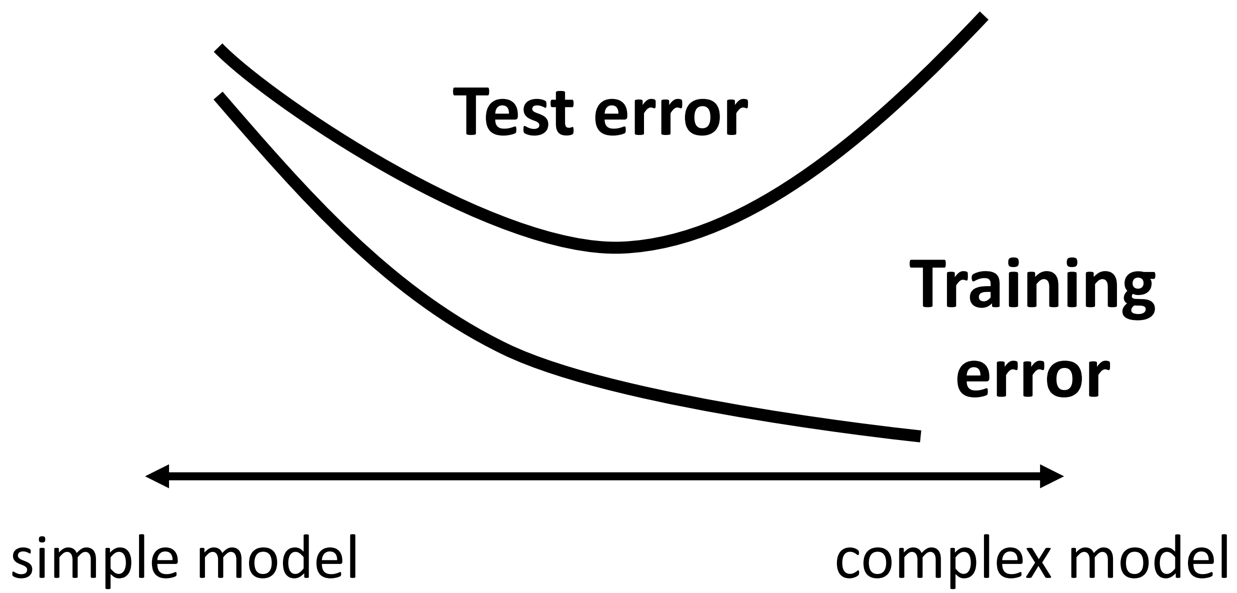


Bias-variance trade-off

- simple model ➔ high bias, low variance
- complex model ➔ low bias, high variance



Test error and training error



This lecture

- Iterative optimisation
  - Loss functions
  - Coordinate descent
  - Gradient descent
- Regularisation
  - Model complexity
  - Constrained modelling
  - Bias-variance trade-off