

Lecture 5. Regularisation

COMP90051 Statistical Machine Learning

Lecturer: Feng Liu



This lecture

- How irrelevant features make optimisation ill-posed
- Regularising linear regression
 - * Ridge regression
 - * The lasso
 - * Connections to Bayesian MAP
- Regularising non-linear regression
- Bias-variance

Regularisation

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

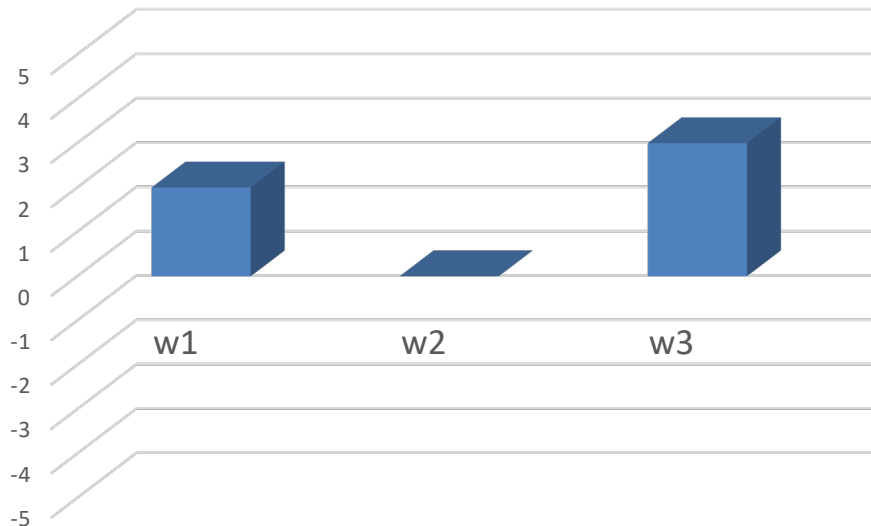
- Major technique & theme, throughout ML
- Addresses one or more of the following related problems
 - * Avoids ill-conditioning (a computational problem)
 - * Avoids overfitting (a statistical problem)
 - * Introduce prior knowledge into modelling
- This is achieved by augmenting the objective function
- In this lecture: we cover the first two aspects. We will cover more of regularisation throughout the subject

The Problem with Irrelevant Features

Linear regression on rank-deficient data.

Example 1: Feature importance

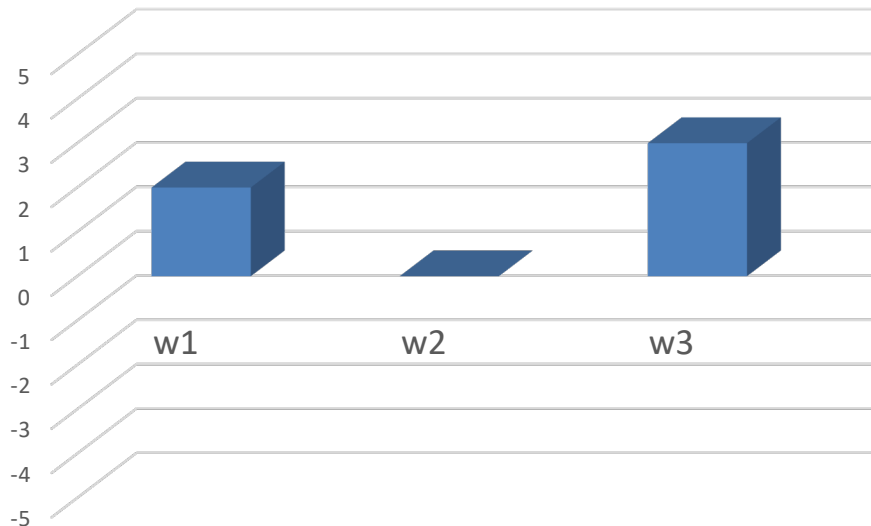
- Linear model on three features
 - * \mathbf{X} is matrix on $n = 4$ instances (rows)
 - * Model: $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$



Question: Which feature is more important?

Example 1: Feature importance

- Linear model on three features
 - * \mathbf{X} is matrix on $n = 4$ instances (rows)
 - * Model: $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$

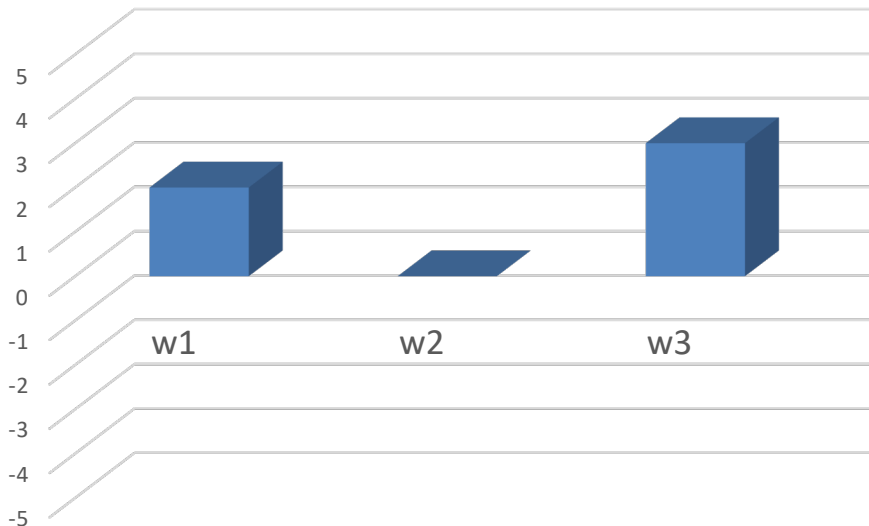


Example 1: Irrelevant features

- Linear model on three features, first two same

- * \mathbf{X} is matrix on $n = 4$ instances (rows)
- * Model: $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$
- * First two columns of \mathbf{X} identical
- * Feature 2 (or 1) is irrelevant

3	3	7
6	6	9
21	21	79
34	34	2



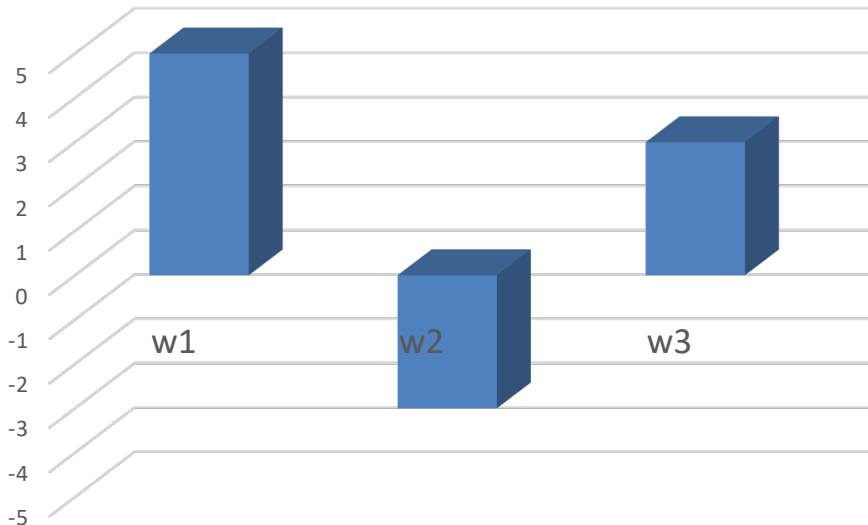
- Effect of perturbations on model predictions?

- * Add Δ to w_1
- * Subtract Δ from w_2

Example 1: Irrelevant features

- Linear model on three features, first two same
 - * \mathbf{X} is matrix on $n = 4$ instances (rows)
 - * Model: $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$
 - * First two columns of \mathbf{X} identical
 - * Feature 2 (or 1) is **irrelevant**

3	3	7
6	6	9
21	21	79
34	34	2



- Effect of perturbations on model predictions?
 - * Add Δ to w_1
 - * Subtract Δ from w_2

Problems with irrelevant features

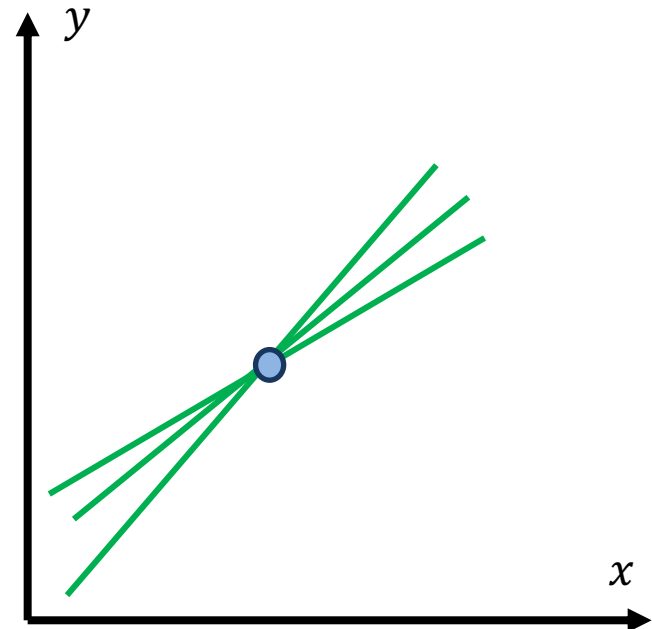
- In example, suppose $[\hat{w}_0, \hat{w}_1, \hat{w}_2, \hat{w}_3]'$ is “optimal”
- For any δ new $[\hat{w}_0, \hat{w}_1 + \delta, \hat{w}_2 - \delta, \hat{w}_3]'$ get
 - * *Same* predictions!
 - * *Same* sum of squared errors!
- Problems this highlights
 - * The solution is not **unique**
 - * Lack of **interpretability**
 - * Optimising to learn parameters is **ill-posed problem**

Irrelevant (co-linear) features in general

- Extreme case: features complete clones
- For linear models, more generally
 - * Feature $\mathbf{X}_{.j}$ is irrelevant if
 - * $\mathbf{X}_{.j}$ is a **linear combination** of other columns
$$\mathbf{X}_{.j} = \sum_{l \neq j} \alpha_l \mathbf{X}_{.l}$$
... for some scalars α_l . Also called **multicollinearity**
 - * Equivalently: Some eigenvalue of $\mathbf{X}'\mathbf{X}$ is zero
- Even **near-irrelevance**/colinearity can be problematic
 - * V small eigenvalues of $\mathbf{X}'\mathbf{X}$
- Not just a pathological extreme; ***easy to happen!***

Example 2: Lack of data

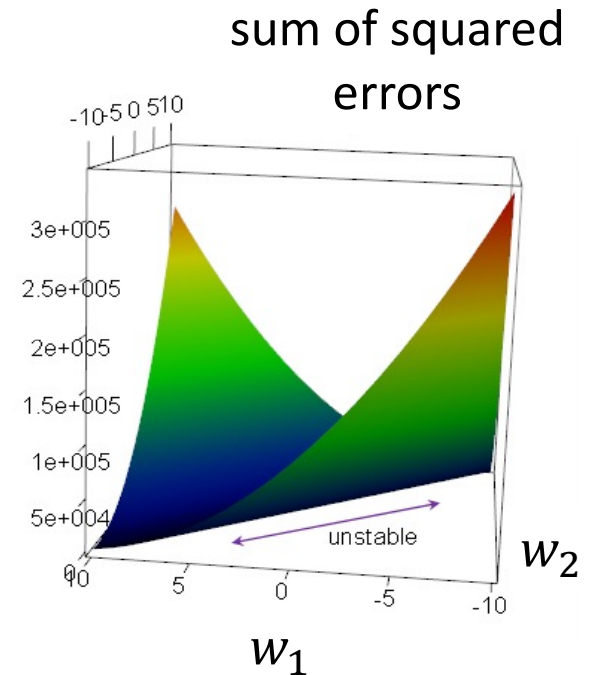
- Extreme example:
 - * Model has two parameters (slope and intercept)
 - * Only one data point
- Underdetermined system



Ill-posed problems

- In both examples, finding the best parameters becomes an **ill-posed problem**
- This means that the problem solution is not defined
 - * In our case w_1 and w_2 cannot be uniquely identified
- Remember normal equations solution of linear regression:

$$\hat{\mathbf{w}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$
- With irrelevant/multicollinear features, matrix $\mathbf{X}'\mathbf{X}$ has **no inverse**



convex, but not strictly convex

Mini Summary

- Irrelevant features as collinearity
- Leads to
 - * Ill-posed optimisation for linear regression
 - * Broken interpretability
- Multiple intuitions: algebraic, geometric
- Next: Regularisation to the rescue!

Regularisation in Linear Models

Ridge regression and the Lasso

Re-conditioning the problem

- Regularisation: introduce an **additional condition** into the system

- The original problem is to minimise $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$

- The regularised problem is to minimise

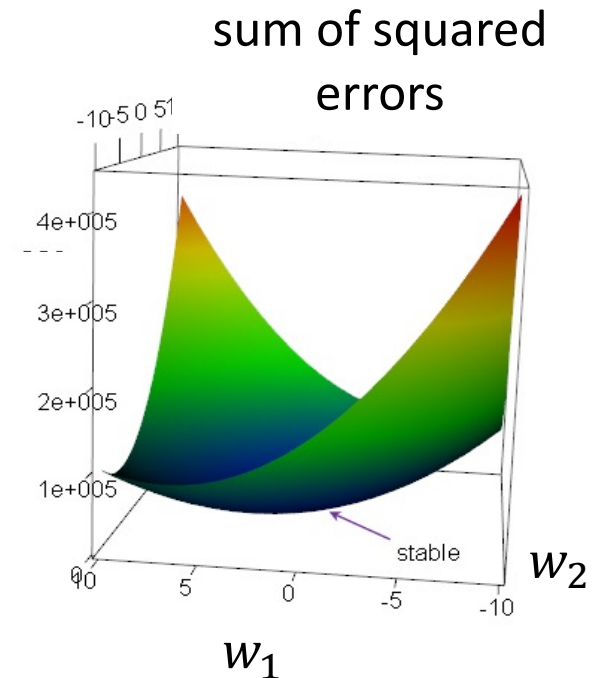
$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \text{ for } \lambda > 0$$

- The solution is now

$$\hat{\mathbf{w}} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}'\mathbf{y}$$



- This formulation is called **ridge regression**
 - * Turns the ridge into a deep, singular valley
 - * Adds λ to eigenvalues of $\mathbf{X}'\mathbf{X}$: makes invertible



strictly convex

Regulariser as a prior

- Without regularisation, parameters found based entirely on the information contained in the training set \mathbf{X}
 - * Regularisation introduces **additional information**
- Recall our probabilistic model $Y = \mathbf{x}'\mathbf{w} + \varepsilon$
 - * Here Y and ε are random variables, where ε denotes noise
- Now suppose that \mathbf{w} is also a random variable (denoted as \mathbf{W}) with a Normal **prior distribution**

$$\mathbf{W} \sim \mathcal{N}(0, 1/\lambda)$$

- * I.e. we expect small weights and that no one feature dominates
- * Is this always appropriate? E.g. data centring and scaling
- * We could encode much more elaborate problem knowledge

Computing posterior using Bayes rule

- The prior is then used to compute the posterior

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$

- Instead of maximum likelihood (MLE), take *maximum a posteriori* estimate (MAP)
- Apply log trick, so that

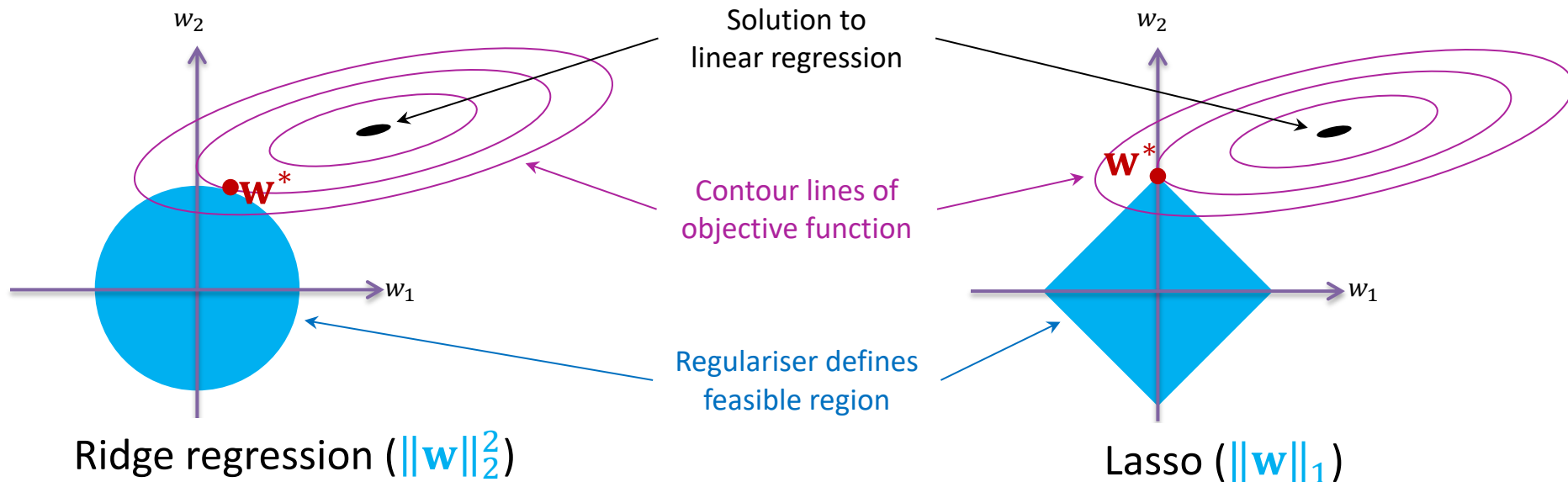
$$\log(\text{posterior}) = \log(\text{likelihood}) + \log(\text{prior}) - \log(\text{marg})$$
- Arrive at the problem of minimising

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

this term doesn't
affect optimisation

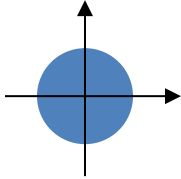
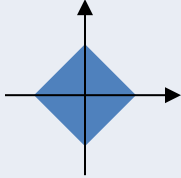
Regulariser as a constraint

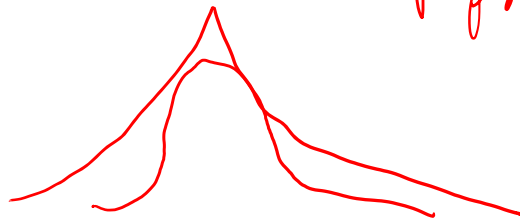
- For illustrative purposes, consider a *modified problem*:
minimise $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$ subject to $\|\mathbf{w}\|_2^2 \leq \lambda$ for $\lambda > 0$



- Lasso (L_1 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	$\ \mathbf{y} - \mathbf{X}\mathbf{w}\ _2^2$	None	$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ (if inverse exists)
Ridge regression	$\ \mathbf{y} - \mathbf{X}\mathbf{w}\ _2^2 + \lambda\ \mathbf{w}\ _2^2$ <i>Gaussian prior</i>	L ₂ norm 	$(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$
Lasso	$\ \mathbf{y} - \mathbf{X}\mathbf{w}\ _2^2 + \lambda\ \mathbf{w}\ _1$ <i>Laplace prior</i>	L ₁ norm 	No closed-form, but solutions are sparse and suitable for high-dim data



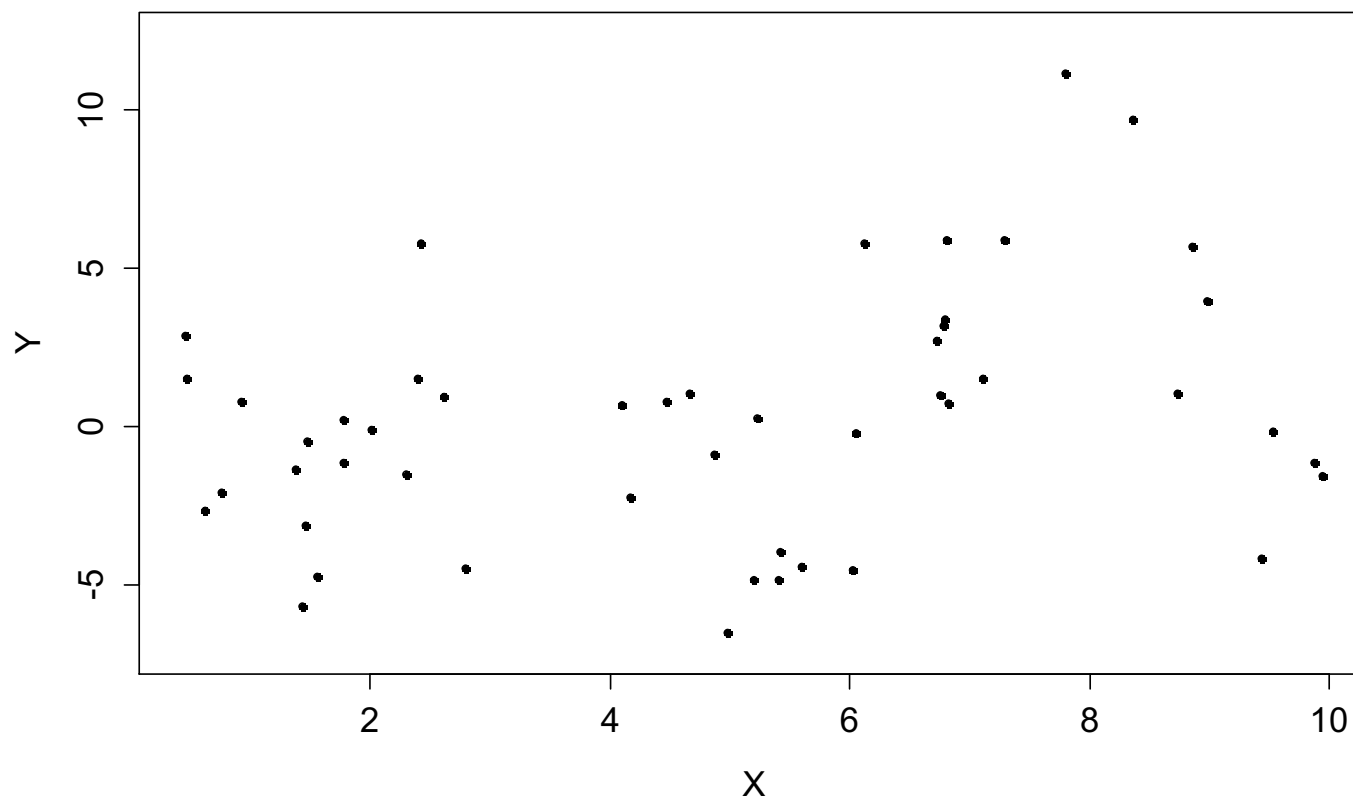
Mini Summary

- L_2 regularisation: Ridge regression
 - * Re-conditions the optimisation
 - * Equivalent to MAP with Gaussian prior on weights
- L_1 regularisation: The Lasso
 - * Particularly favoured in high-dim, low-example regimes
- Next: Regularisation and non-linear regression

Regularisation in Non-Linear Models

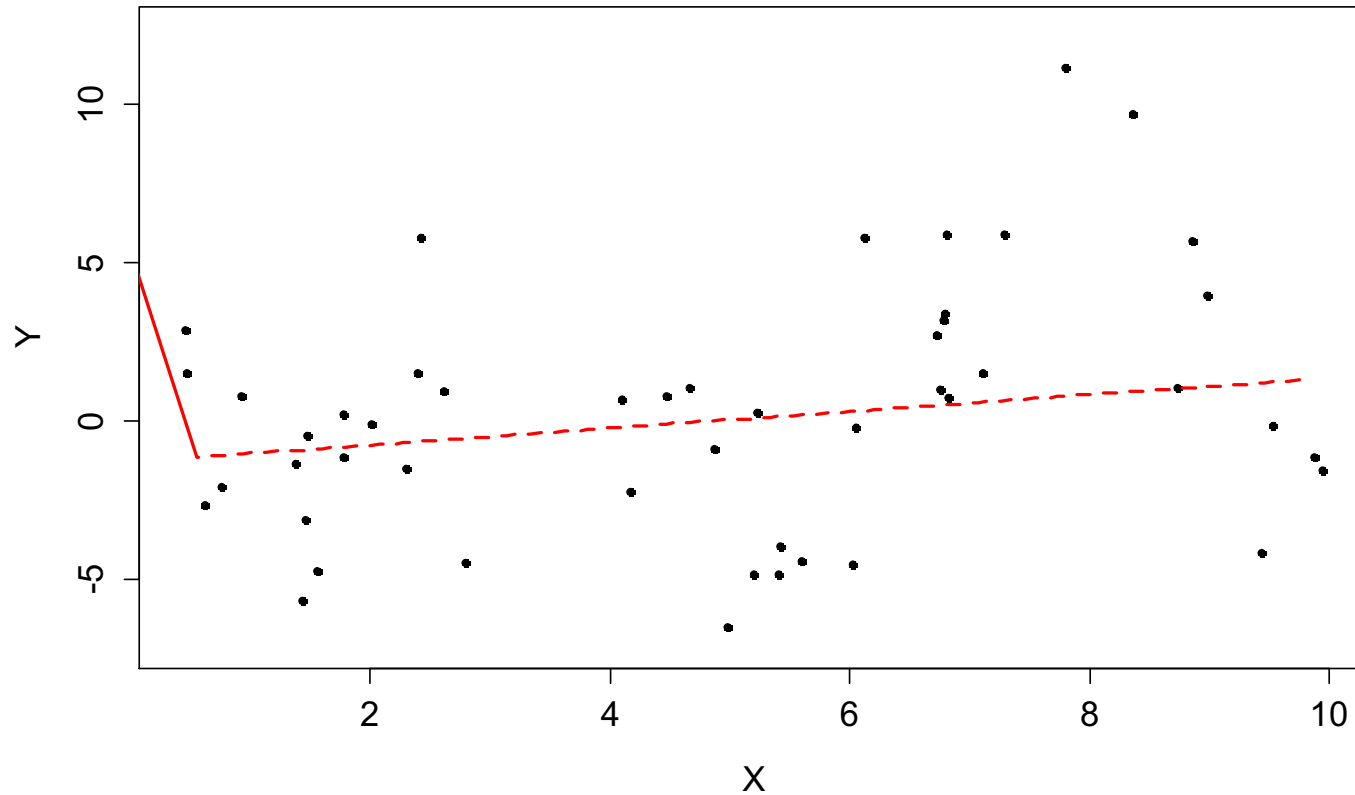
Model selection in ML

Example regression problem



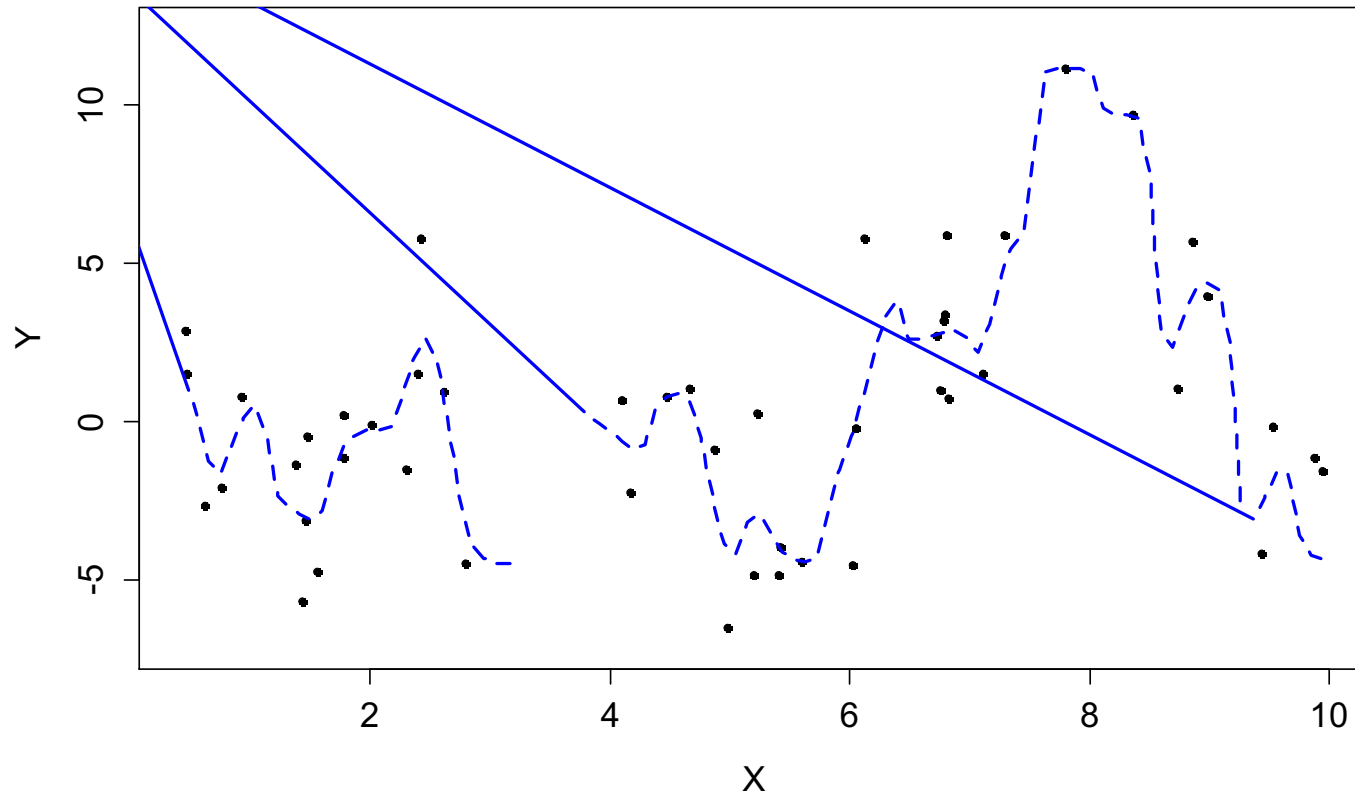
How complex a model should we use?

Underfitting (linear regression)



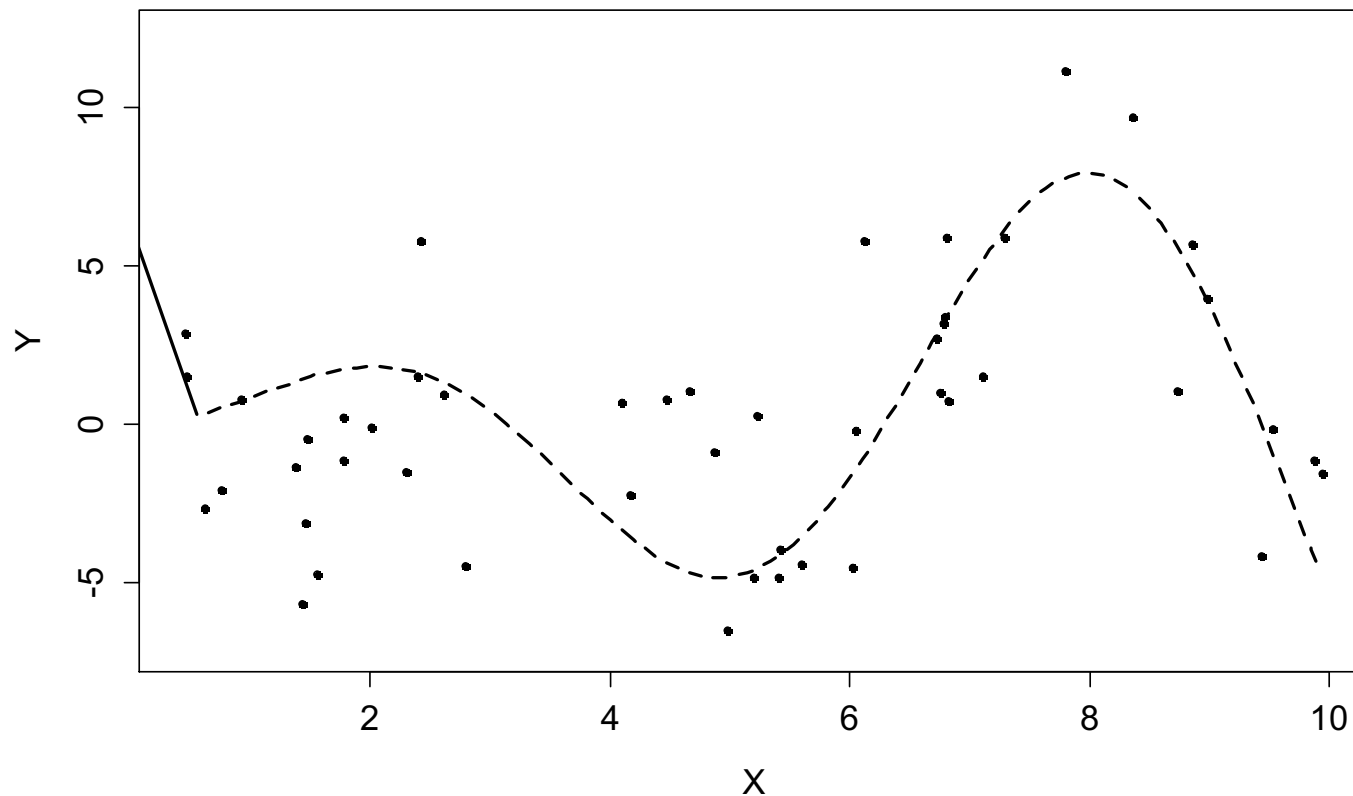
Model class Θ can be **too simple** to possibly fit true model.

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.

Approach: 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

Approach: Regularisation

- Augment the problem:

$$\hat{\boldsymbol{\theta}} \in \operatorname{argmin}_{\boldsymbol{\theta} \in \Theta} (L(\text{data}, \boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta}))$$

- E.g., ridge regression

$$\hat{\mathbf{w}} \in \operatorname{argmin}_{\mathbf{w} \in W} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

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

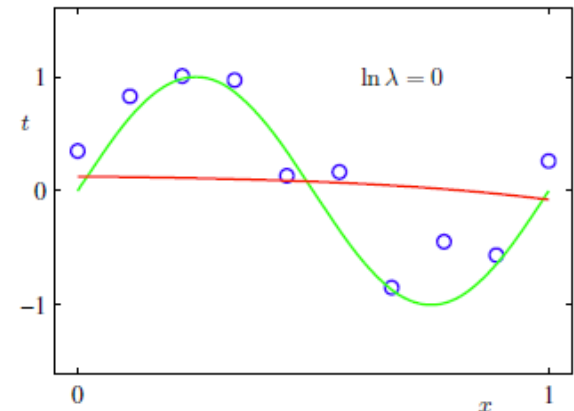
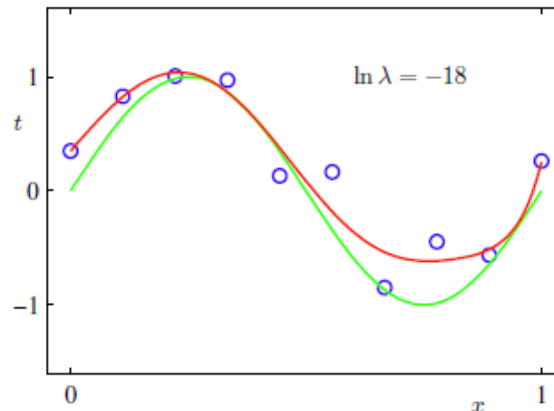
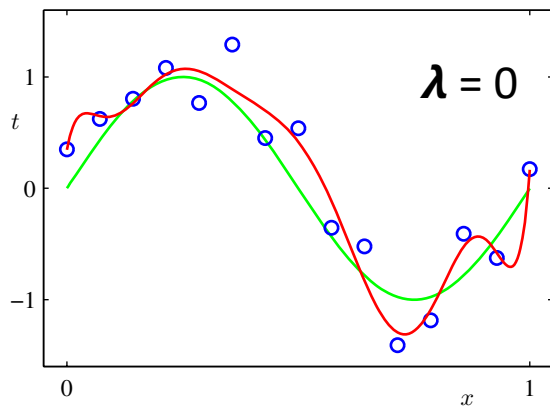
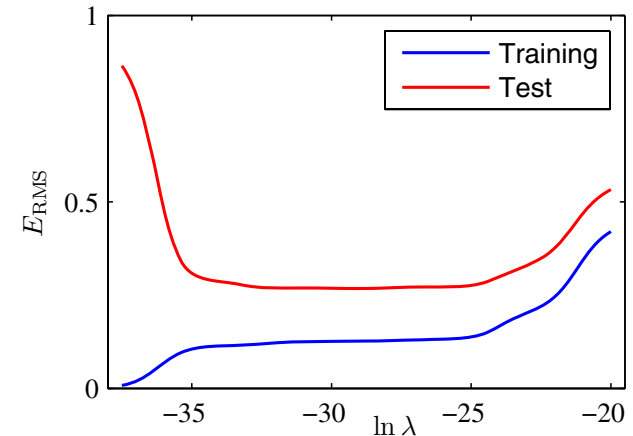
Example: Polynomial regression

- 9th-order polynomial regression

- * model of form

$$\hat{f} = w_0 + w_1 x + \dots + w_9 x^9$$

- * regularised with $\lambda \|\mathbf{w}\|_2^2$ term



Mini Summary

- Overfitting vs underfitting
- Effect of regularisation on nonlinear regression
 - * Controls balance of over- vs underfitting
 - * Controlled in this case by the penalty hyperparameter
- Next: Bias-variance view for regression

Bias-variance trade-off

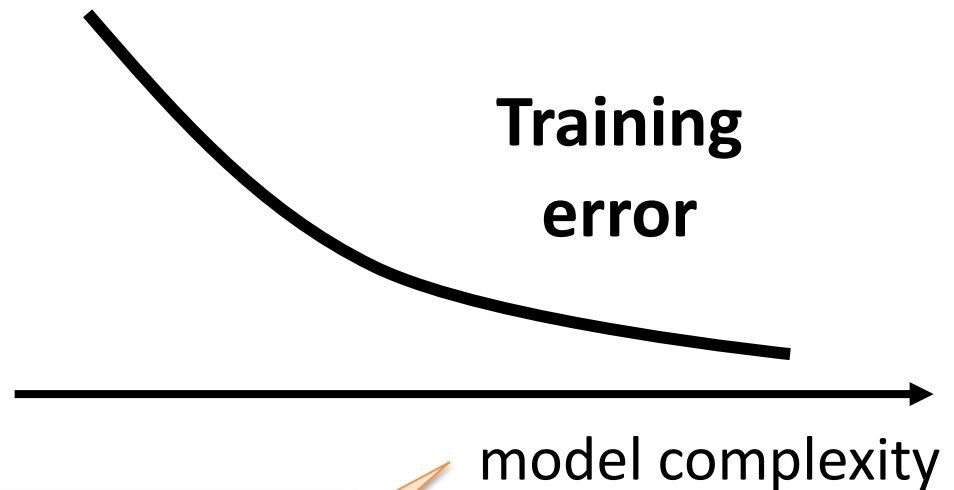
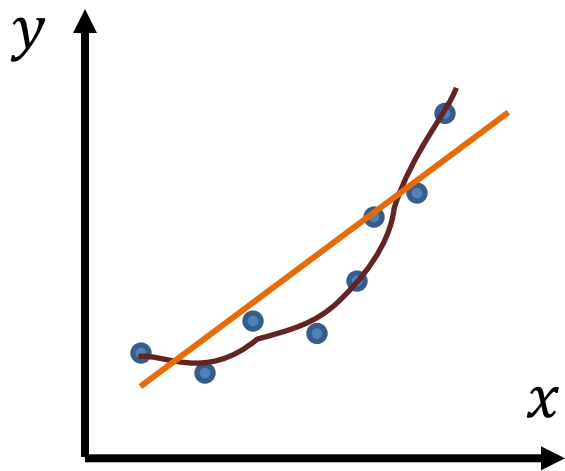
Train error, test error and
model complexity in
supervised regression

Assessing generalisation

- Supervised learning: train the model on existing data, then make predictions on new data
- Training the model: ERM / minimisation of training error
- Generalisation capacity is captured by risk / test error
- Model complexity is a major factor that influences the ability of the model to generalise (**vague still**)
- In this section, our aim is to explore error in the context of supervised regression. One way to decompose it.

Training error and model complexity

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



What is this?

(Another) Bias-variance decomposition

- Squared loss for **supervised-regression** predictions

$$l(Y, \hat{f}(\mathbf{X}_0)) = (Y - \hat{f}(\mathbf{X}_0))^2$$

Classification
later on

- Lemma: Bias-variance decomposition

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

Risk /
test error
for x_0

(bias)²

variance

irreducible
error

* Prediction randomness comes from randomness in test features AND training data

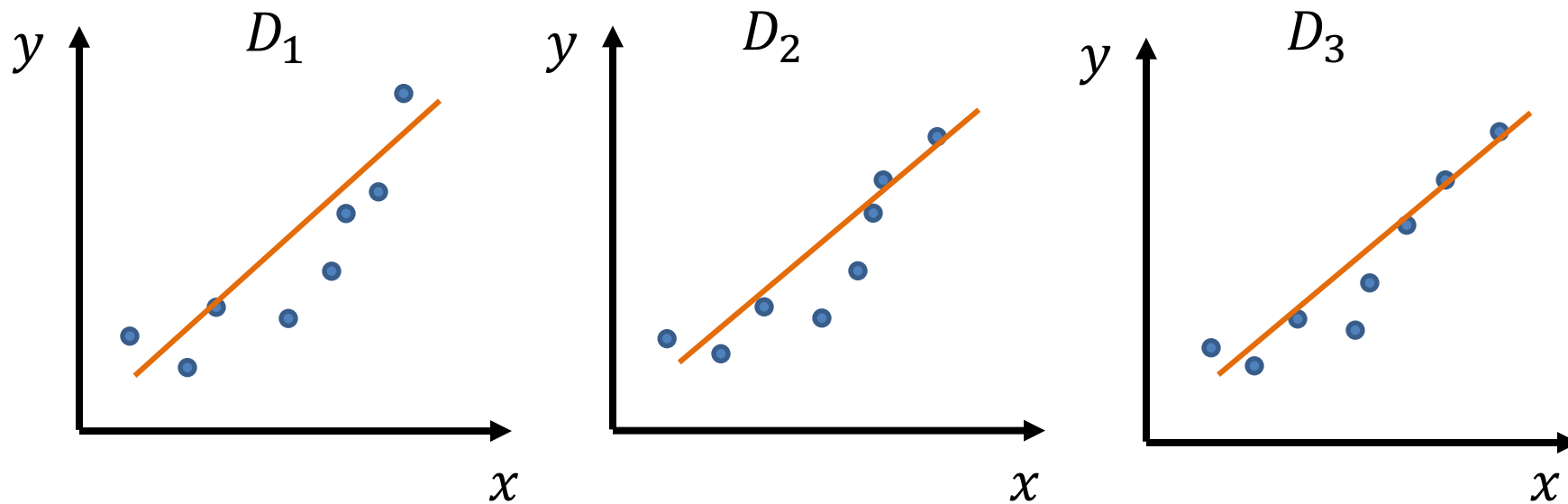
Decomposition proof sketch

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

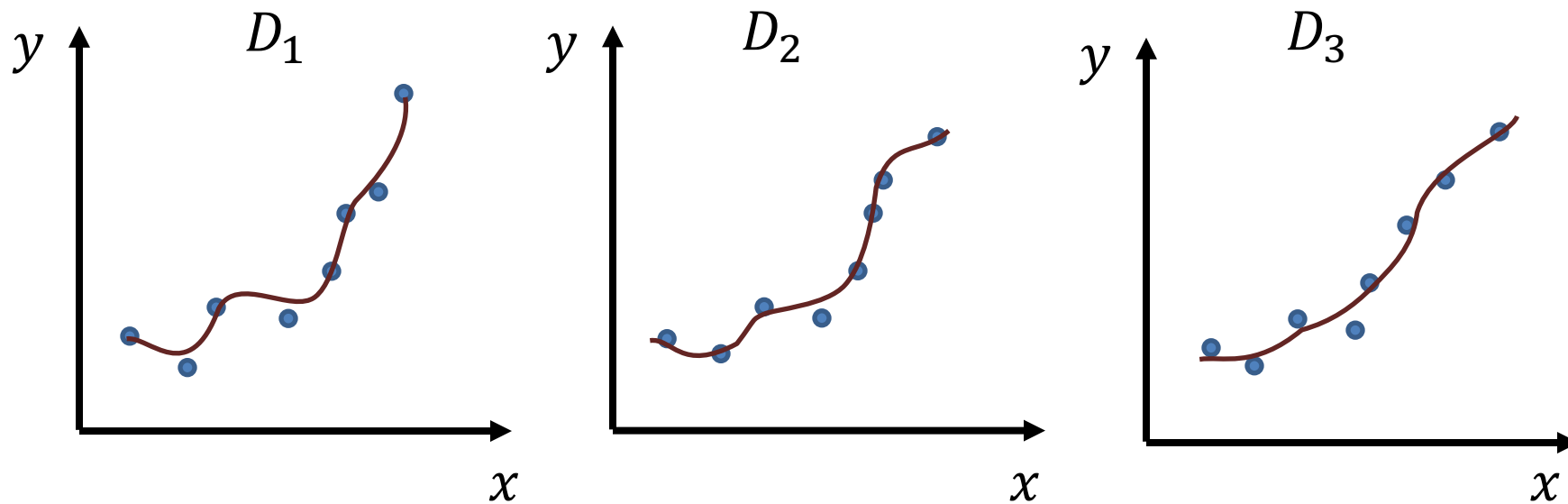
A key formula used here:
 $\text{Var}[Y] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2$



Training data as a random variable

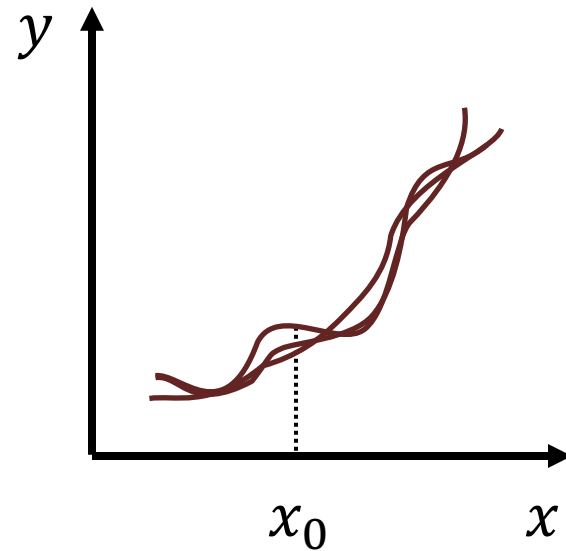
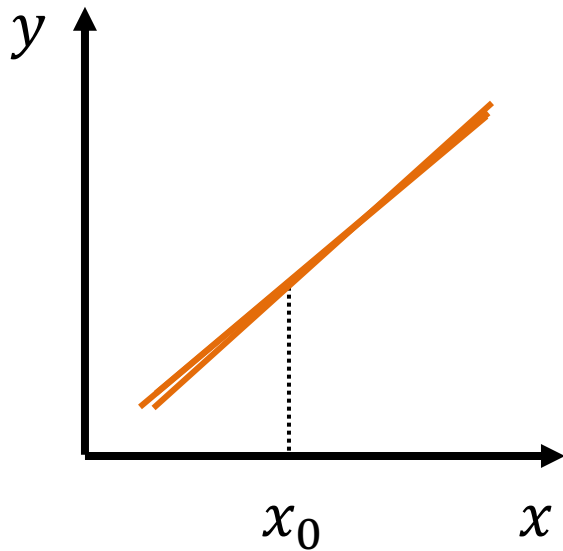


Training data as a random variable



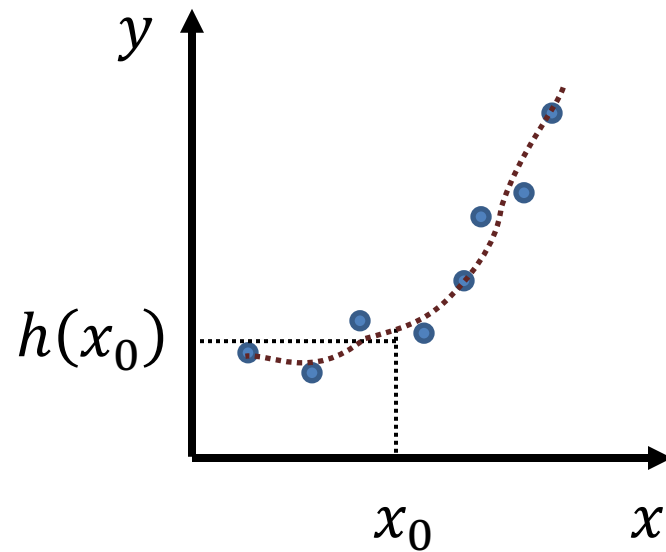
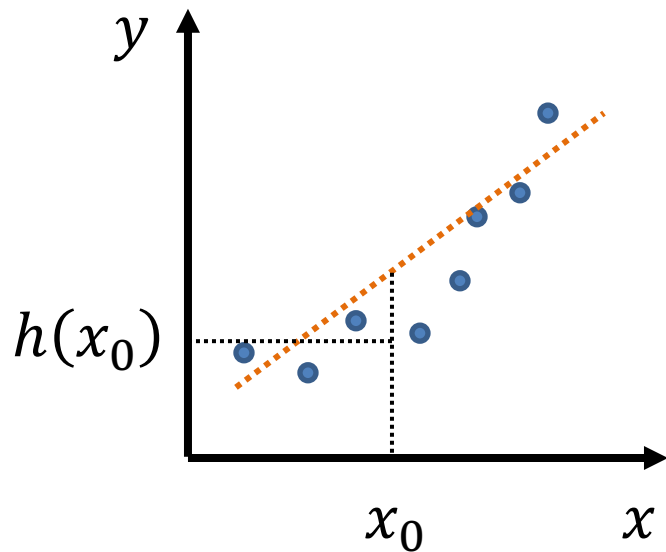
Intuition: Model complexity and variance

- simple model \rightarrow low variance
- complex model \rightarrow high variance



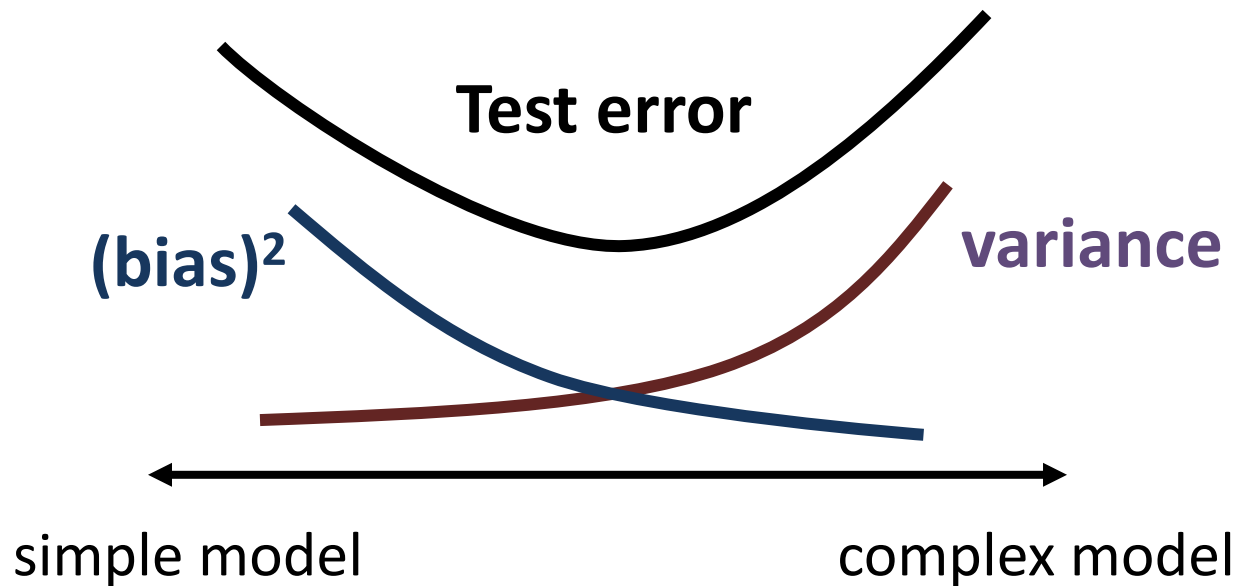
Intuition: Model complexity and variance

- simple model \rightarrow high bias
- complex model \rightarrow low bias

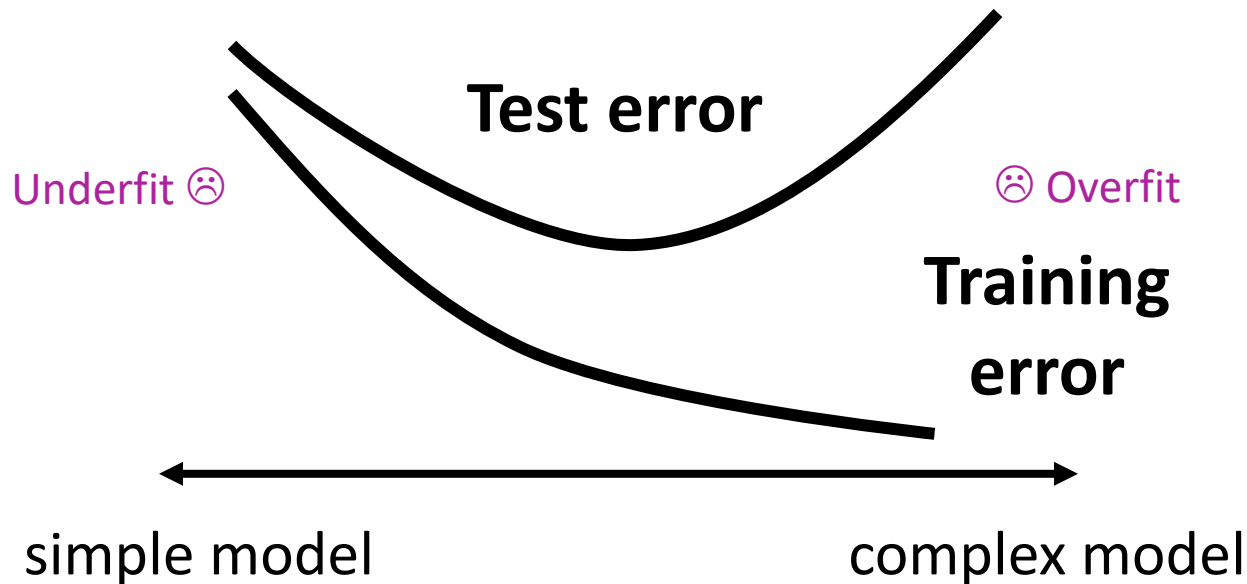


Bias-variance trade-off

- simple model \rightarrow high bias, low variance
- complex model \rightarrow low bias, high variance



Test error and training error



But how to measure
model family complexity?

Mini Summary

- Supervised regression: square-loss risk decomposes to bias, variance and irreducible terms
- This trade-off mirrors under/overfitting
- Controlled by “model complexity”
 - * But we’ve been vague about what this means!?
- Next lectures: Bounding generalisation error in ML