# QBUS3820: Statistical Learning and Data Mining

Lecture 3: Linear Regression, K-Nearest Neighbours and Model Selection

---

Semester 1, 2019

Discipline of Business Analytics, The University of Sydney Business School

# Lecture 3: Linear Regression, K-Nearest Neighbours and Model Selection

1. Statistical properties of OLS
2. The Gaussian MLR model
3. K-Nearest Neighbours
4. Comparison with linear regression
5. Model selection: simulated example
6. Validation and cross-validation
7. Analytical criteria

# Statistical properties of OLS

# Least squares (reminder)

Training data $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^n = \{(y_i, x_{i1}, \ldots, x_{ip})\}_{i=1}^n$ satisfies:

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \varepsilon_i, \quad i = 1, \ldots, n$$

For each candidate coefficient vector $\tilde{\boldsymbol{\beta}}$ we define the residual sum of squares:

$$\text{RSS}(\tilde{\boldsymbol{\beta}}) = \sum_{i=1}^{n} \Big( y_i - [\tilde{\beta}_0 + \tilde{\beta}_1 x_{i1} + \tilde{\beta}_2 x_{i2} + \ldots + \tilde{\beta}_p x_{ip}] \Big)^2$$

The ordinary least squares (OLS) method selects the coefficients that minimise the residual sum of squares:

$$\widehat{\boldsymbol{\beta}} \;=\; \text{minimizer of RSS}$$

In order to obtain a solution to the OLS minimisation problem, we need linear algebra.

# OLS estimator in matrix form

**Additional Material to Lecture 3 (see Canvas)** shows that the OLS estimator of the linear regression coefficient vector $\beta$ is

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

where

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix} \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix}$$

# Sampling distribution of an estimator

In classical statistics, the population parameter $\beta$ is fixed and the data is a random sample from the population. We estimate $\beta$ by applying an **estimator** $\widehat{\beta}$ to data (in our case the OLS estimator).

As the sample on which estimator $\widehat{\beta}$ is computed is random, the value of the estimator is a random variable, and the distribution of this random variable is called the **sampling distribution**.

We can study the uncertainty of an estimate by understanding the sampling distribution of the estimator.

# Sampling distribution of an estimator

Suppose that we draw a large number of different datasets $\mathcal{D}^{(s)}$ $(s = 1, \ldots, S)$ from the population. Each has sample size $n$.

On each of these datasets, we apply the estimator $\widehat{\beta}$ and obtain a set of estimates $\{\widehat{\beta}(\mathcal{D}^{(s)})\}_{s=1}^{S}$. The sampling distribution can be thought of as the distribution of these estimates as we let $S \to \infty$.

This concept may seem nonintuitive as it refers to hypothetical datasets rather than the data that we do have.

# Mean and variance of the OLS estimator

To simplify the exposition, we will treat the predictor values as given, rather than as realizations of a random variable.

It follows from the MLR assumptions ($E[\varepsilon] = 0$, $\text{Var}[\varepsilon] = \sigma^2$) and the OLS formula we derived earlier that the mean and variance of the OLS estimator are:

$$E(\widehat{\boldsymbol{\beta}}) = \boldsymbol{\beta} \quad \text{(no bias)}$$
$$\text{Var}\left(\widehat{\boldsymbol{\beta}}\right) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$$

The formulas are left without proof

# Variance

We can obtain a more interpretable expression for the variance of the individual coefficients:

$$\mathsf{Var}(\widehat{\beta}_j) = \frac{\sigma^2}{\left(1 - \mathsf{R}_j^2\right)(n-1)s_{x_j}^2}$$

$\mathsf{R}_j^2$ is the R-squared from the linear regression of predictor $j$ on all other predictors.

$s_{x_j}^2$ is the sample variance of predictor $j$.

# Variance and SD

$$\text{Var}(\widehat{\beta}_j) = \frac{\sigma^2}{\left(1 - \text{R}_j^2\right)(n-1)s_{x_j}^2}$$

**Interpretation:**

- The variance of $\widehat{\beta}_j$ is proportional to the error variance, $\sigma^2$.
- The variance of $\widehat{\beta}_j$ decreases as $n$ increases.
- The higher the correlation of the $j$-th predictor with other predictors, the higher the variance of $\widehat{\beta}_j$.
- The variance of $\widehat{\beta}_j$ decreases as the sample variance of the $j$-th predictor increases.

Note that $SD\left(\widehat{\beta}_j\right) = \sqrt{\text{Var}(\widehat{\beta}_j)}$

# Standard Errors of the estimated coefficients

The formula for $SD\left(\widehat{\beta}_j\right)$ relies on the knowledge of the error variance ($\sigma^2$).

An unbiased estimator of $\sigma^2$ is:

$$\widehat{\sigma}^2 = \frac{\sum_{i=1}^{n} e_i^2}{n-p-1}$$

When we replace $\sigma$ with $\widehat{\sigma}$ in the formula for the $SD\left(\widehat{\beta}_j\right)$, we get the formula for the standard error: $SE\left(\widehat{\beta}_j\right)$, which we can actually calculate from the data.

# Predictions

The OLS method leads to unbiased predictions. To get a feel for its variance, we consider predictions on the training data.

The OLS prediction for the $i$th individual is:

$$\widehat{f}(\boldsymbol{x}_i) = \widehat{\beta}_0 + x_{i1}\widehat{\beta}_1 + \ldots + x_{ip}\widehat{\beta}_p.$$

It can be shown that the average variance of predictions on the training set is:

$$\frac{1}{n}\sum_{i=1}^{n} \mathsf{Var}\left[\widehat{f}(\boldsymbol{x}_i)\right] = \frac{p+1}{n}\sigma^2$$

Think about how each component $(n,\, \sigma^2,\, p)$ affects the average variance of the predictions on the training data.

# The Gaussian MLR model

# The Gaussian MLR model

- Our analysis so far did not make any distributional assumptions about the regression errors. We made assumptions, such as $E[\varepsilon] = 0$ and $\text{Var}[\varepsilon] = \sigma^2$, but left the probability distribution of $\varepsilon$ unspecified.

- We managed to learn a lot from these minimal assumptions. For example, the mean and variance of the OLS estimator.

- But we may want to learn more. For example, what is the full sampling distribution of the OLS estimator? Knowing this distribution is necessary for making probability statements about the uncertainty in this estimator.

# The Gaussian MLR model

We now add the Gaussian assumption on the distribution of the error terms: $\varepsilon \sim N(0, \sigma^2)$, which means $\varepsilon$ has Normal distribution with mean $0$ and variance $\sigma^2$.

As a consequence of the Gaussian MLR model, we get the exact sampling distribution of the OLS estimator:

$$\widehat{\boldsymbol{\beta}} \sim N\left(\boldsymbol{\beta}, \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\right)$$

where

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}$$

# Sampling distribution

Another consequence of the Gaussian MLR model is:

$$\frac{\widehat{\beta}_j - \beta_j}{\mathsf{SE}(\widehat{\beta}_j)} \sim t_{n-p-1}$$

where $t_{n-p-1}$ denotes the $t$-distribution with $n - p - 1$ degrees of freedom. We rely on this fact for confidence intervals and hypothesis testing.

# Confidence interval (CI)

Recall the basic structure:

$$\textbf{estimate} \ \pm \ \textbf{critical value} \ \times \ \textbf{standard error}$$

An approximate $100 \times (1-\alpha)\%$ confidence interval for $\beta_j$ is:

$$\widehat{\beta}_j \pm t_{n-p-1,\alpha/2} \times \mathsf{SE}(\widehat{\beta}_j)$$

where $t_{n-p-1,\alpha/2}$ is the $100 \times (1-\alpha/2)$ percentile of the $t_{n-p-1}$ distribution.

In the most common case of the $95\%$ CI, $t_{n-p-1,\alpha/2} \approx 2$, provided $n-p-1$ is not too small.

# Hypothesis Testing

$H_0 : \beta_j = 0$ vs $H_1 : \beta_j \neq 0$

Test statistic

$$t_{\text{stat}} = \frac{\widehat{\beta}_j}{SE(\widehat{\beta}_j)}$$

Under $H_0$ we have $t_{\text{stat}} \sim t_{n-p-1}$

Statistical software typically reports the $p$-value:

$$p\text{-value} = P(t_{n-p-1} > |t_{\text{stat}}|)$$

The smaller this $p$-value, the more the evidence against $H_0$.

Common threshold: $0.05$

# K-Nearest Neighbours

# Introduction

In our first lecture we discussed the distinction between parametric and nonparametric models:

- **parametric models** assume that regression function $f$ has a fixed number of parameters. Parametric models are faster to use and more interpretable, but make stronger assumptions about the data;

- in **nonparametric models** the number of estimated parameters grows with the size of the training data. These methods are more flexible, but have larger variance.

We will now consider K-nearest neighbours regression, which is a nonparametric approach.

# K-nearest neighbours

Given training data $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^n$ and an input point $\boldsymbol{x}$,

**K-nearest neighbours** (KNN) regression makes the following prediction at $\boldsymbol{x}$:

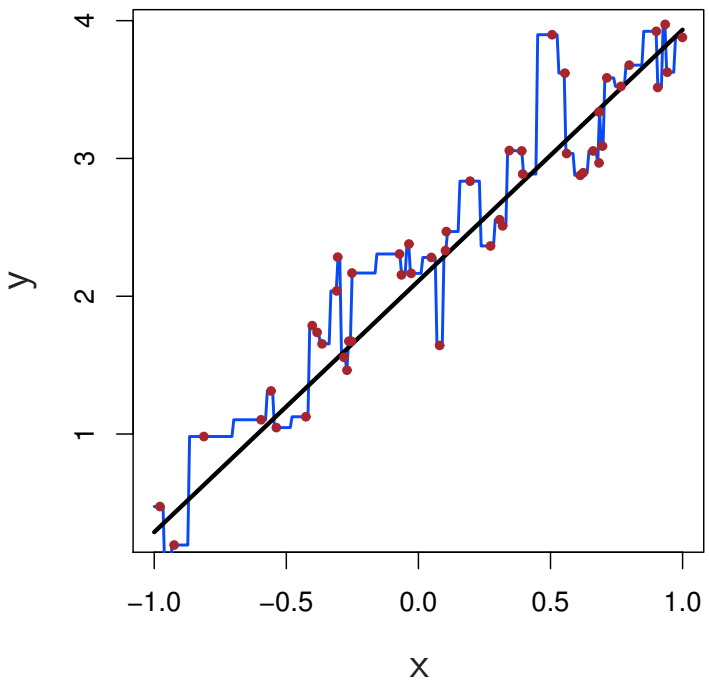
$$\widehat{f}(\boldsymbol{x}) = \text{Average}\Big[\, y_i \,\Big|\, \boldsymbol{x}_i \text{ is in } N_k(x) \,\Big]$$

Here we average those $y_i$ whose $\boldsymbol{x}_i$ lie in the neighborhood $N_k(\boldsymbol{x})$ containing the closest $k$ data points to $\boldsymbol{x}$.

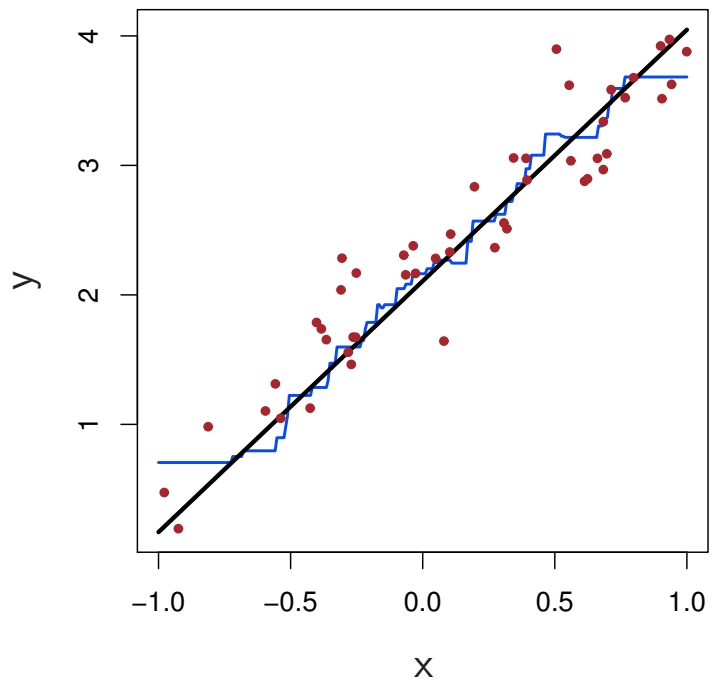
# Illustration

Black: true regression function; Red: simulated training data

Blue: KNN fit for $k = 1$




Blue: KNN fit for $k = 9$




(Figure from ISL)

# K-nearest neighbours

$$\widehat{f}(\boldsymbol{x}) = \text{Average}\Big[\, y_i \,\Big|\, \boldsymbol{x}_i \text{ is in } N_k(\boldsymbol{x}) \,\Big]$$

Recall that we want to estimate the true regression function $f(\boldsymbol{x}) = E(Y|X = \boldsymbol{x})$.

In KNN:

- Expected value $E(Y|X = \boldsymbol{x})$ is approximated by an average.
- Conditioning on a point $X = \boldsymbol{x}$ is relaxed to conditioning on a neighbourhood $N_k(\boldsymbol{x})$ that is close to $\boldsymbol{x}$.

# K-nearest neighbours

Under mild regularity assumptions on the distribution of the data, the following holds for KNN:

$$\widehat{f}(\boldsymbol{x}) \to f(\boldsymbol{x}) \quad \text{as} \quad n, k \to \infty \quad \text{and} \quad (k/n) \to 0$$

In other words, provided there is enough data, the KNN method can potentially approximate any regression function $f$ without making assumptions about the form of this function.

# Modelling choices

$$\widehat{f}(\boldsymbol{x}) = \text{Average}\Big[\, y_i \,|\, \boldsymbol{x}_i \text{ is in } N_k(x) \,\Big]$$

The KNN method requires us to specify:

1. The number of neighbours.
2. The predictors.
3. A distance metric.

# Choosing the number of neighbours

- The number of neighbours $k$ is a **tuning parameter**: we need to specify it prior to the learning process.
- We cannot use the training data to pick $k$ since we would then always choose $k = 1$ and fit the data perfectly!
- Instead, we select $k$ based on bias-variance trade-off considerations.
- This is an instance of **model selection**. We can estimate the test error for each candidate value of $k$. We will then select the value of $k$ with the lowest error according to this criterion.

# Bias-variance decomposition

Recall the following expression for the expected prediction error:

$$\mathrm{E}\left[(Y_0 - \widehat{f}(\boldsymbol{x}_0))^2 | X = \boldsymbol{x}_0\right] = \sigma^2 + \mathrm{Bias}^2\left(\widehat{f}(\boldsymbol{x}_0)\right) + \mathrm{Var}\left(\widehat{f}(\boldsymbol{x}_0)\right)$$

For the KNN method, the expression has the following simple form:

$$\sigma^2 + \left[f(\boldsymbol{x}_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(\boldsymbol{x}_{(\ell)})\right]^2 + \frac{\sigma^2}{k}$$

where $\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(k)}$ denote the $k$ nearest neighbours of $\boldsymbol{x}_0$

# Bias-variance decomposition

$$\mathrm{E}\left[(Y_0 - \widehat{f}(\boldsymbol{x}_0))^2 | X = \boldsymbol{x}_0\right] = \sigma^2 + \left[f(\boldsymbol{x}_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(\boldsymbol{x}_{(\ell)})\right]^2 + \frac{\sigma^2}{k}$$

- For small $k$ the bias will be relatively small, because the regression function evaluated at the neighbours will be close to $f(\boldsymbol{x}_0)$. However, when $k$ is small we average only a few observations, which leads to high variance.

- As we increase $k$ we reduce the variance, but at the cost of higher bias.

# Illustration

Play the animation to see how the estimates change as we vary $k$.

# Curse of dimensionality

- The KNN method is subject to a **curse of dimensionality**: it breaks down with high-dimensional inputs (large $p$).
- The reason is that as we increase the number of predictors, it becomes exponentially more difficult to find training observations that are reasonably close to $\boldsymbol{x}$.
- The curse of dimensionality is a prevalent problem with nonparametric methods.

# Distance

A common distance measure is the Euclidean distance:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_l) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{lj})^2}$$

The Euclidean distance only makes sense if the predictors are on the same scale. An alternative is to use the normalised Euclidean distance:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_l) = \sqrt{\sum_{j=1}^{p} \left( \frac{x_{ij} - x_{lj}}{s_{x_j}} \right)^2},$$

where $s_{x_j}$ is the sample standard deviation of predictor $j$ in the training sample.

# Mahalanobis distance

A more complicated measure that often works better in practice is the Mahalanobis distance

$$d(\boldsymbol{x}_i, \boldsymbol{x}_l) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T S^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)},$$

where $S$ is the sample covariance matrix of the predictors.

# Computational considerations

- Generating KNN predictions is computationally costly. For each new input point, we need to compute distances to all the training points, and sort these values. This differs from the linear regression approach, where computing predictions is cheap.

- The KNN method is a memory intensive method. It requires us to keep the entire training sample in the memory for computing predictions.

# Comparison with linear regression

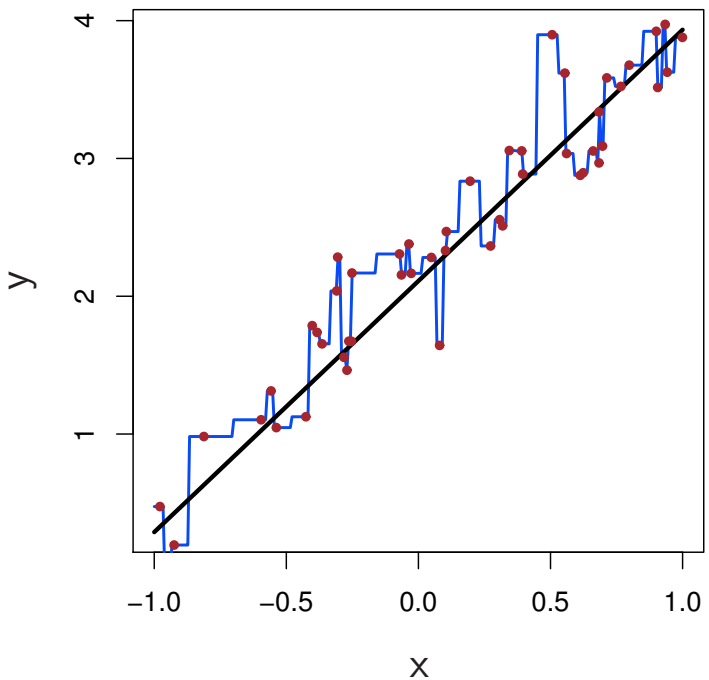
# Comparison with linear regression

- Linear regression and KNN methods represent two highly different approaches to supervised learning.
- Linear regression assumes a linear form for the regression function $f$. This assumption leads to stable predictions $\widehat{f}(\boldsymbol{x})$, but the model can be highly inaccurate (high bias) if the assumption of linearity in the parameters is incorrect.
- KNN makes no structural assumptions about $f$, leading to low bias. But its predictions can be very unstable (high variance), since only a few training observations contribute to each prediction.
- In general, a parametric approach outperforms a nonparametric approach if the parametric form assumed for the regression function is close to the true form of $f$.

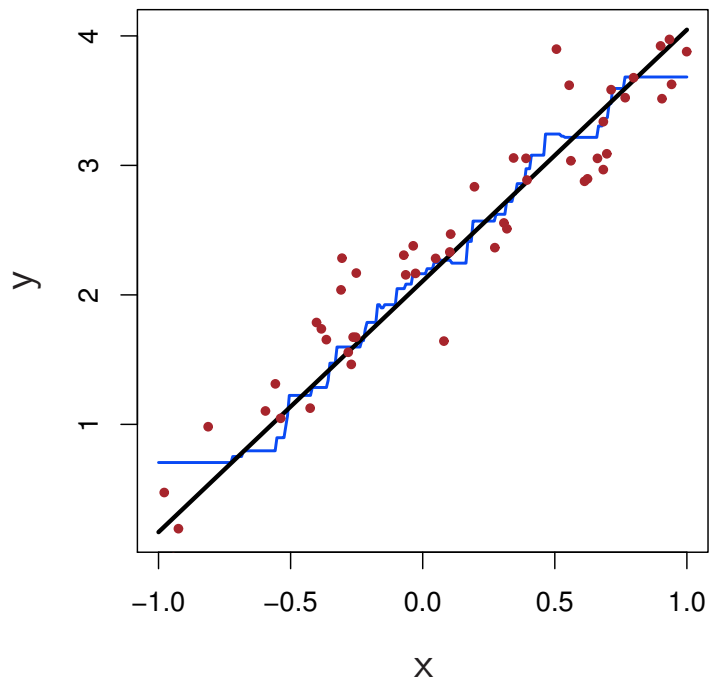
# Illustration revisited

Black: true regression function; Red: simulated training data

Blue: KNN fit for $k = 1$



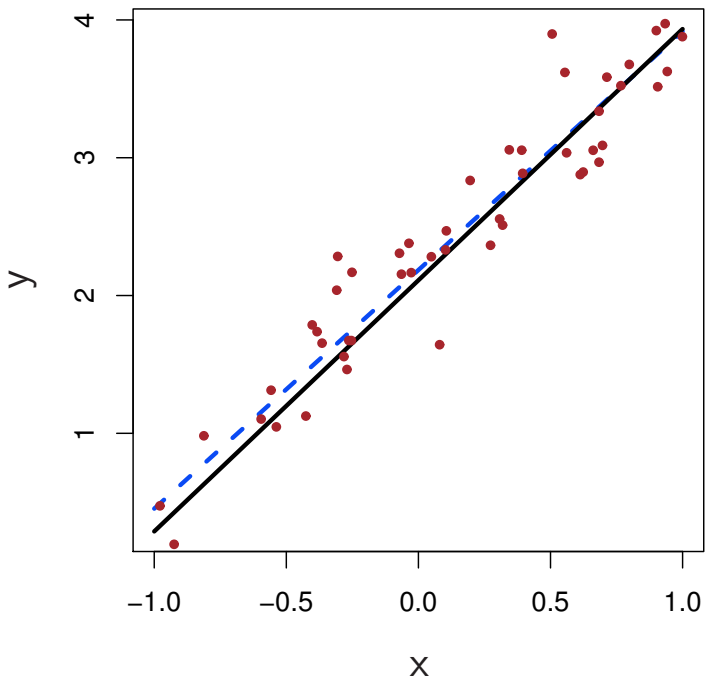

Blue: KNN fit for $k = 9$

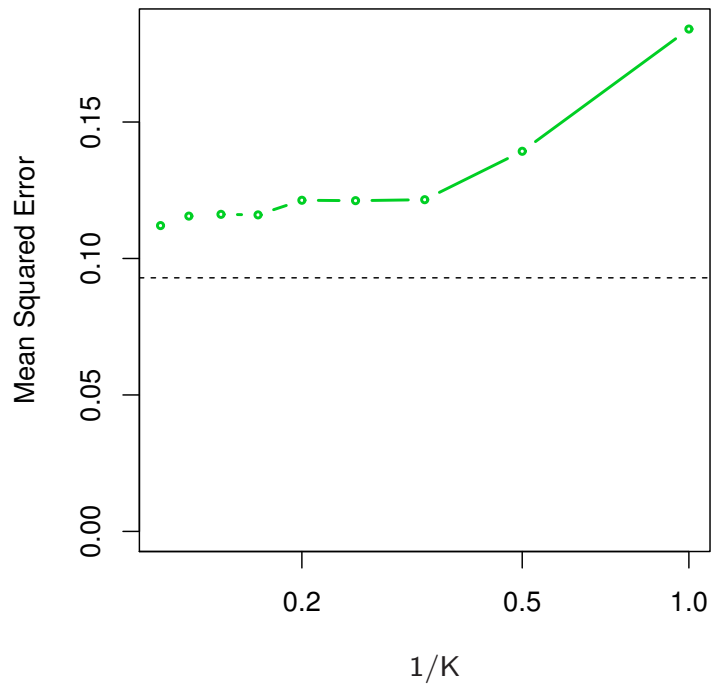



(Figure from ISL)

# Linear regression and k-NN: Illustration

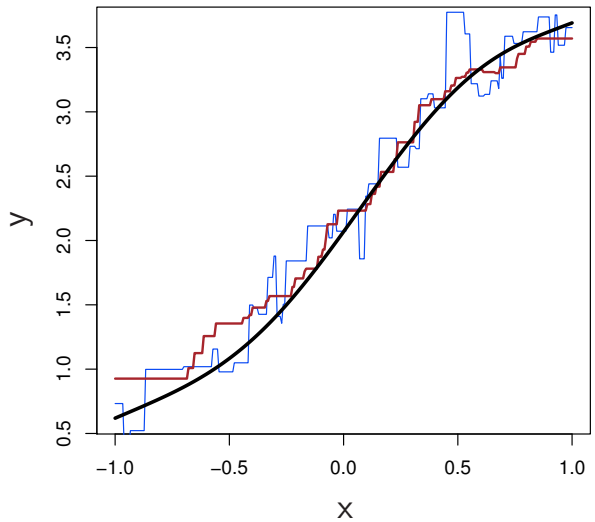
Blue dashed: OLS fit  Green: KNN test MSE; dashed: OLS

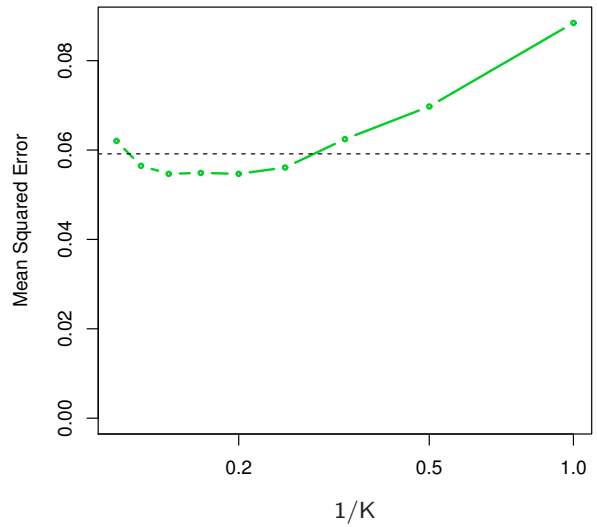


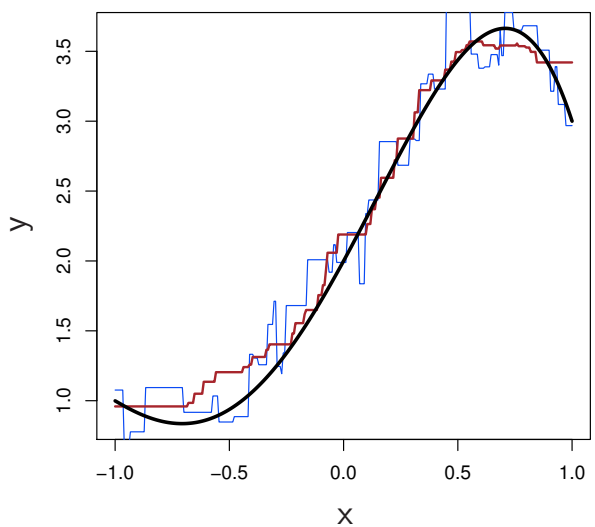



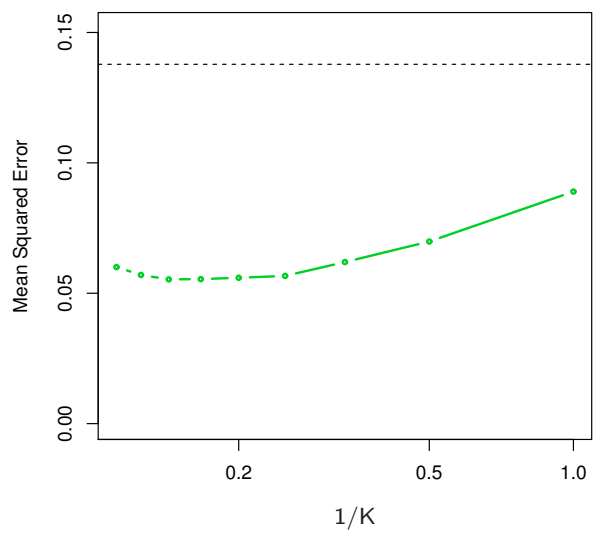

(Figure from ISL)

# Same as before, but a nonlinear true regression function



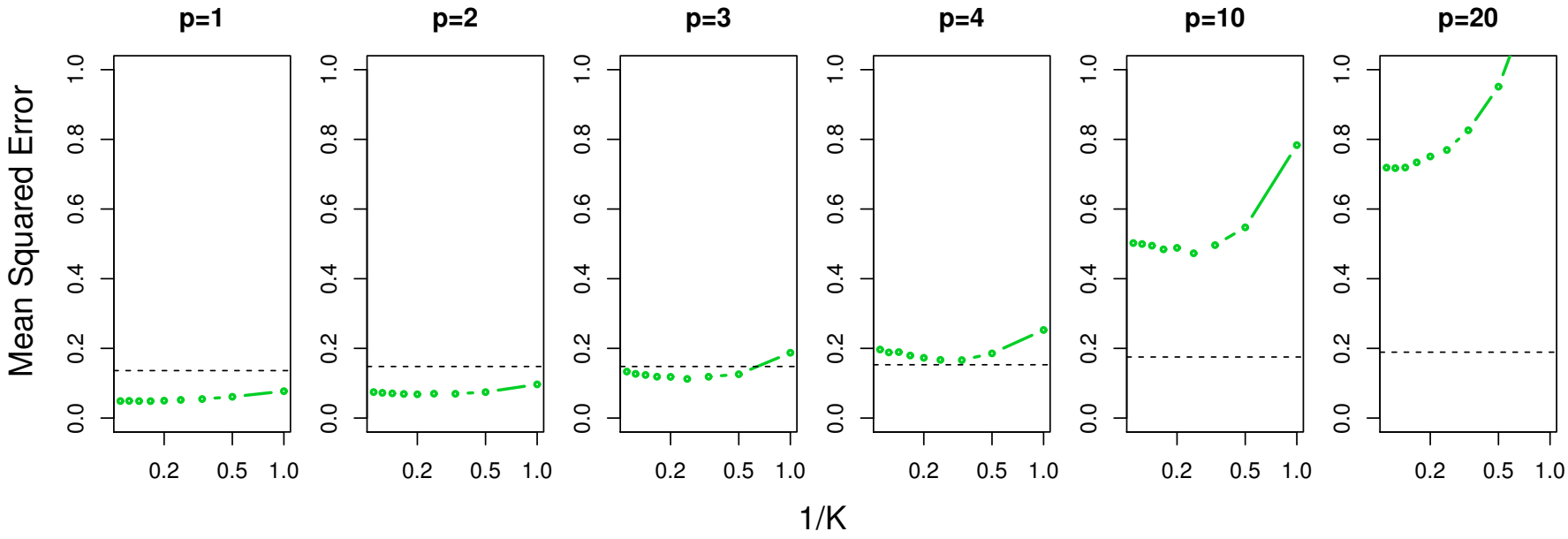



# Many predictors

Only the first predictor is in the true model, the rest are noise; true function is as in the last example above




(Figure from ISL)

# Model selection: simulated example

# Model selection

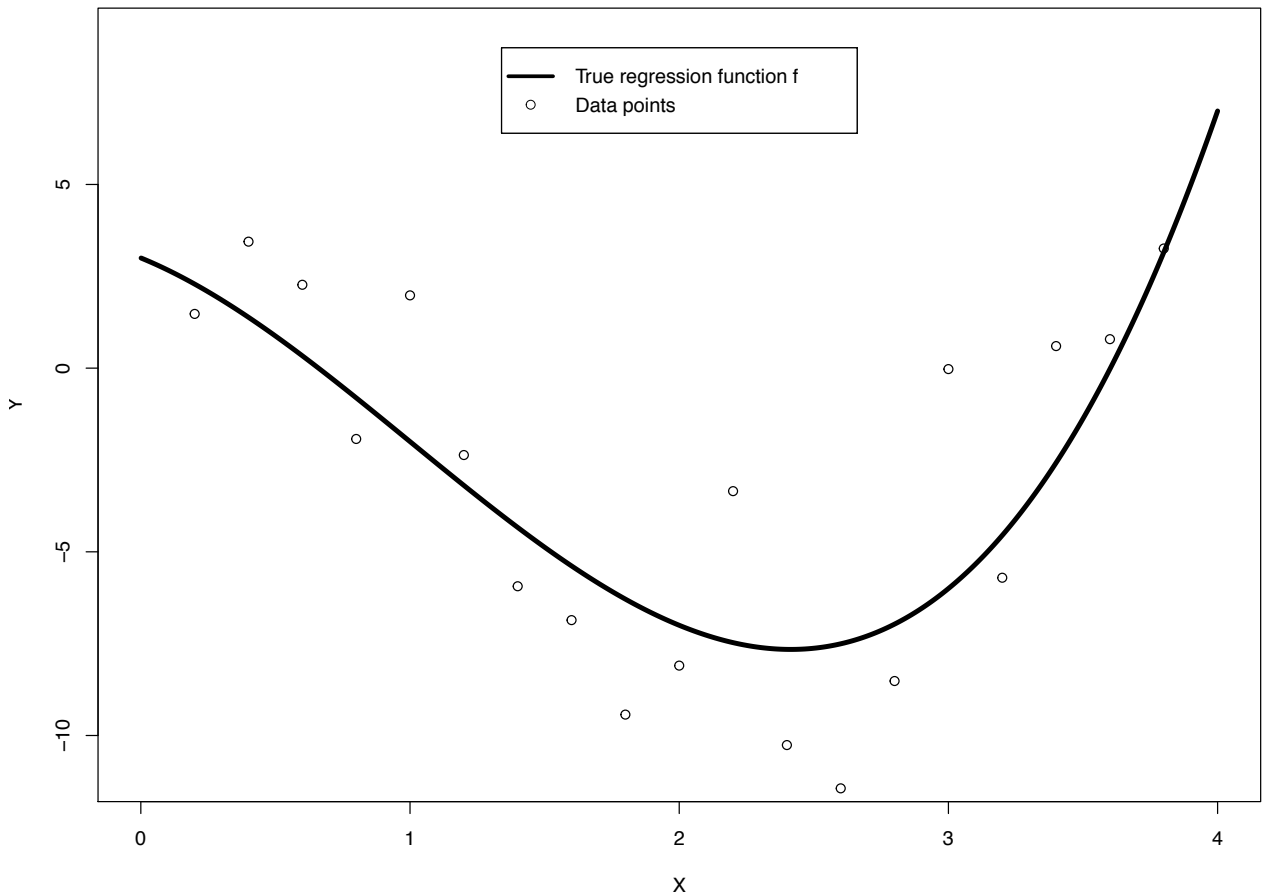
**Model selection** methods estimate the expected test error of a model based on the training data, allowing us to choose between models with different degrees of complexity.

We select the model that is estimated to have the best predictive ability.

# Simulated Example

The data is generated from the following (true) model:

$$Y = 3 - 3X - 3X^2 + X^3 + \varepsilon \qquad \text{where } \varepsilon \sim N(0, 6)$$



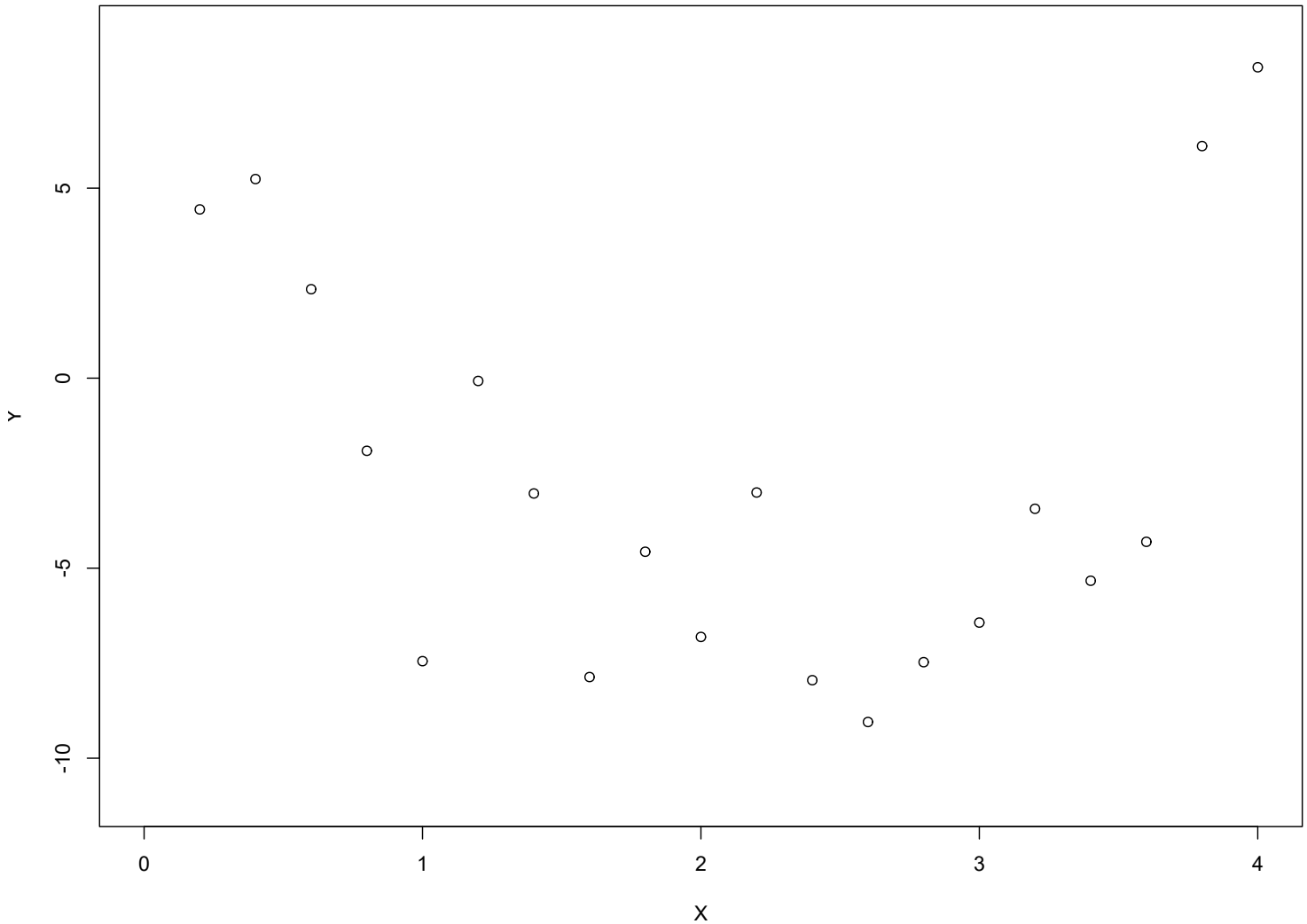
# Simulated Example

The following three models are estimated using OLS:

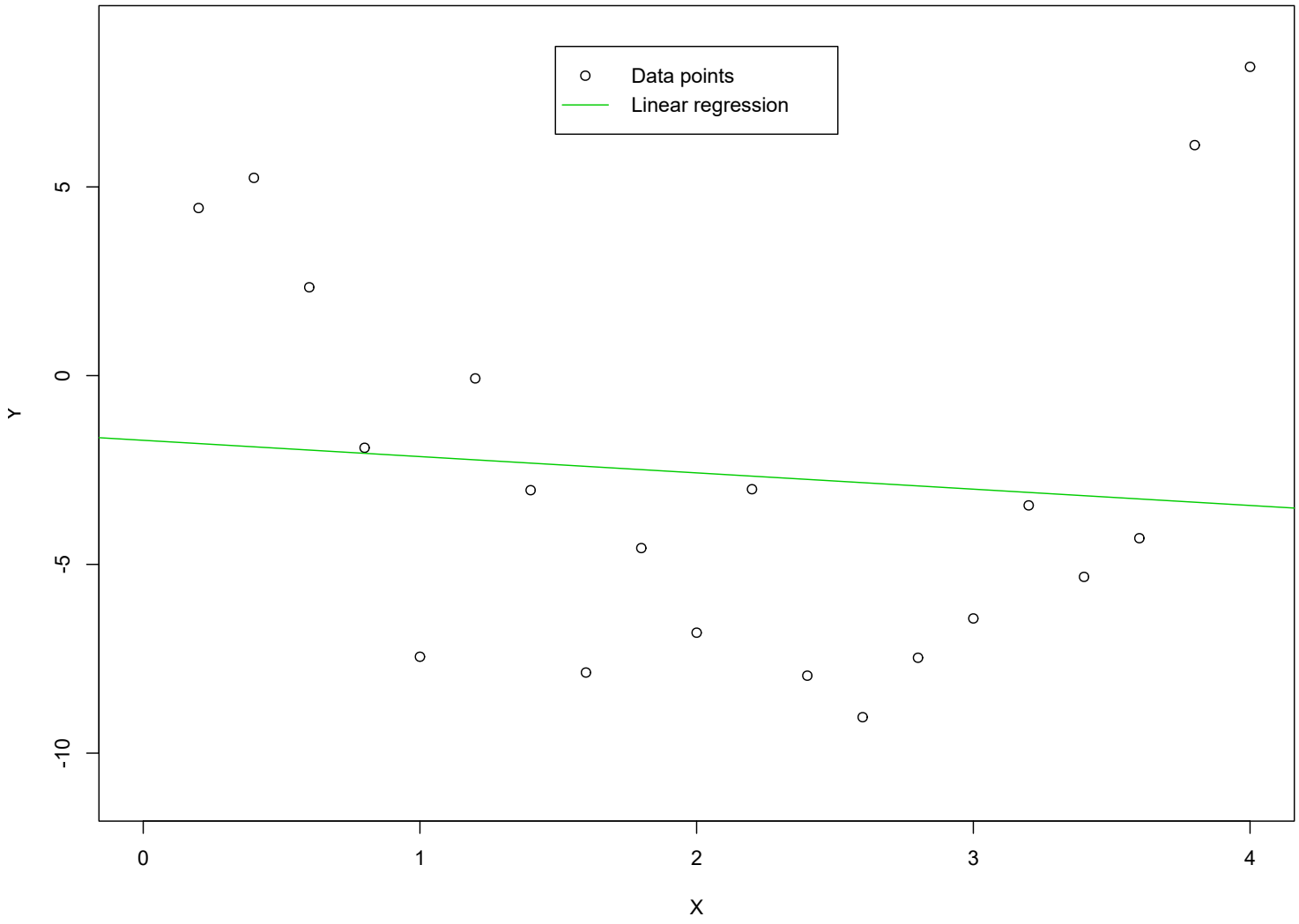
- Linear regression ($X$ is the only predictor).
- Cubic regression (predictors: $X$, $X^2$ and $X^3$).
- 12-degree polynomial regression (predictors: $X$, $X^2$,...,$X^{12}$).

We will compare the estimated regression functions $\widehat{f}(x)$ to the true one: $f(x) = 3 - 3x - 3x^2 + x^3$.

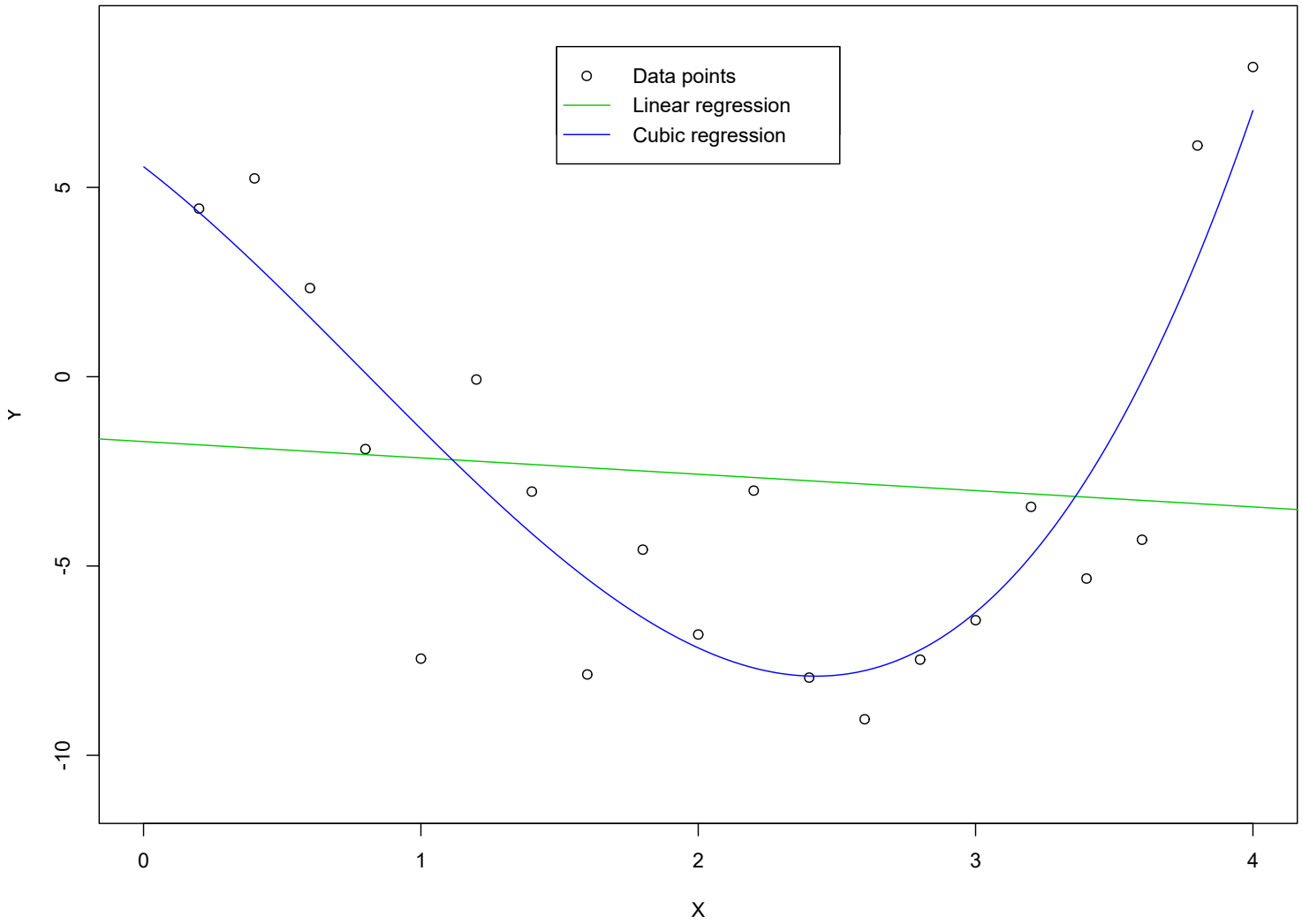
# The Data



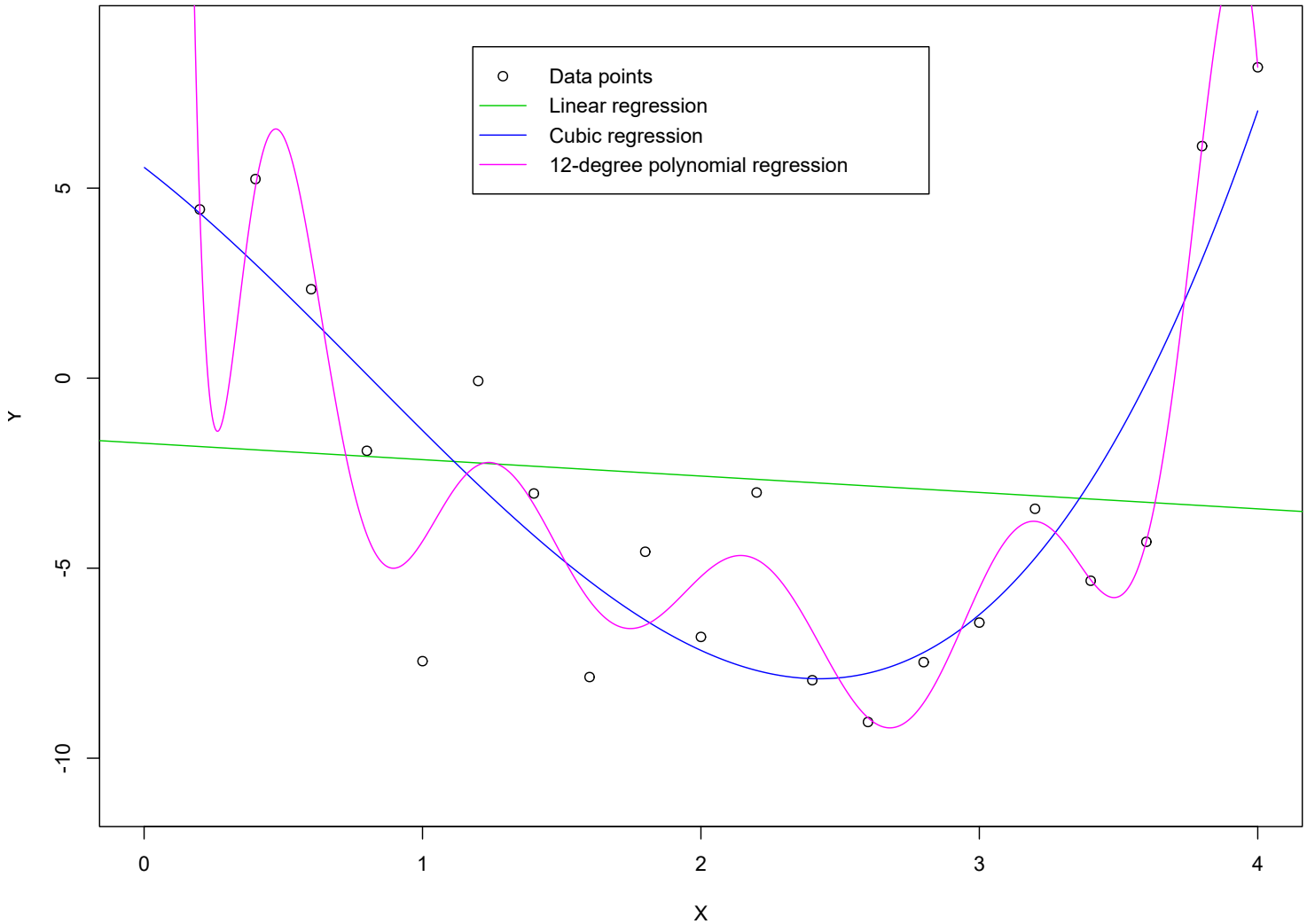
# Linear fit



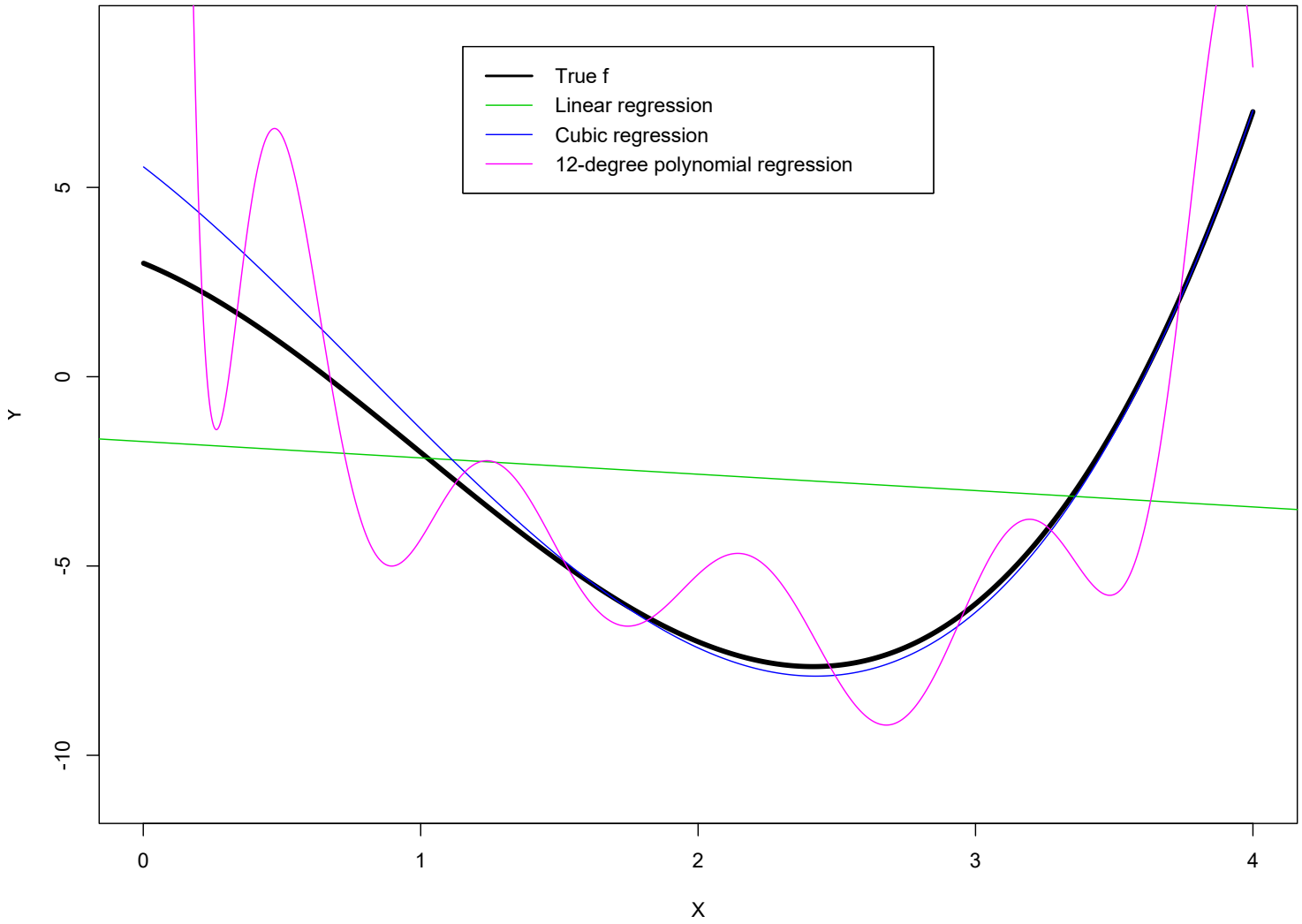
# Linear and Cubic



# All three fits to the data



# Comparison with the true regression function $f$



# Prediction error

We now focus on the prediction at $X = 2$.

The reducible part of the expected prediction error splits into squared bias and variance:

$$\begin{aligned}
\text{Reducible Error} &= E\left[(f(2) - \widehat{f}(2))^2\right] \\
&= \left(E[\widehat{f}(2)] - f(2)\right)^2 + E\left[\left(\widehat{f}(2) - E[\widehat{f}(2)]\right)^2\right] \\
&= \text{Bias}^2\left(\widehat{f}(2)\right) \quad + \quad \text{Var}\left(\widehat{f}(2)\right)
\end{aligned}$$

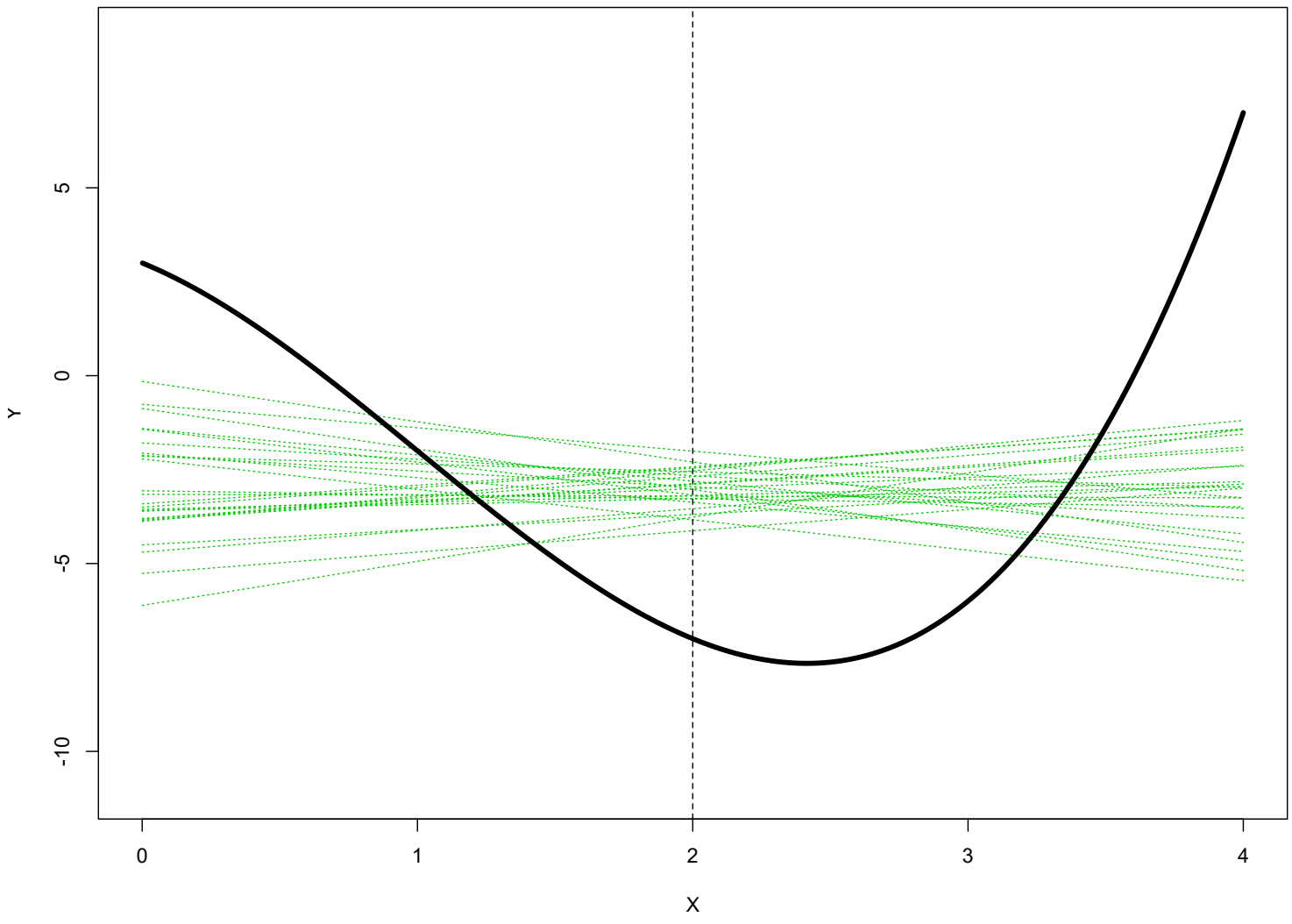
# Prediction error

$$\begin{aligned}\text{Reducible Error} &= \left(E[\widehat{f}(2)] - f(2)\right)^2 + E\left[\left(\widehat{f}(2) - E[\widehat{f}(2)]\right)^2\right] \\ &= \text{Bias}^2\left(\widehat{f}(2)\right) \quad + \quad \text{Var}\left(\widehat{f}(2)\right)\end{aligned}$$

We can approximate the Bias by generating many datasets from the true model and averaging the $\left[\widehat{f}(2) - f(2)\right]$ values produced for the different datasets.

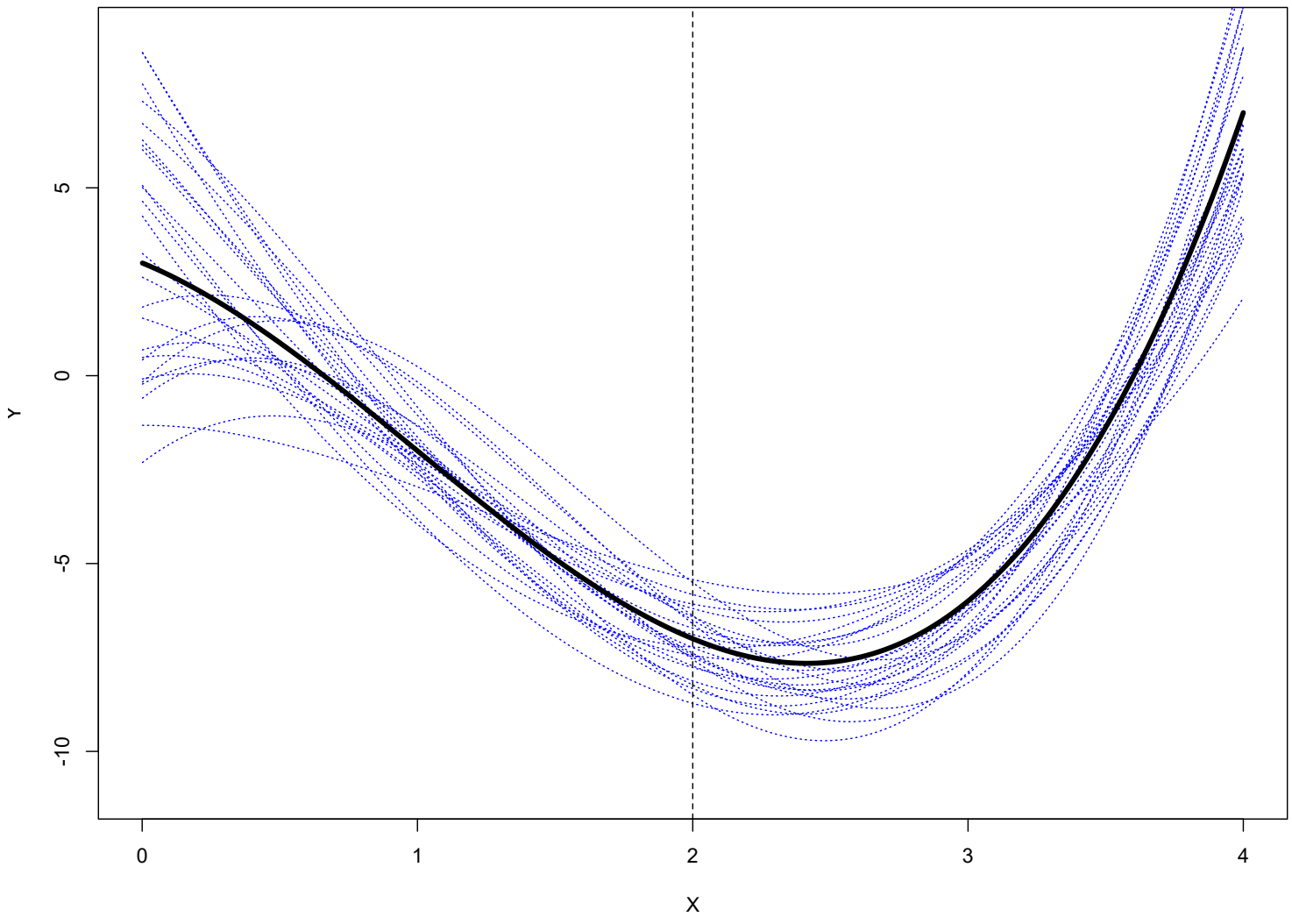
We can similarly approximate the variance by calculating the variance of $\widehat{f}(2)$ values produced for the different datasets.

# Linear fit for $25$ simulated datasets



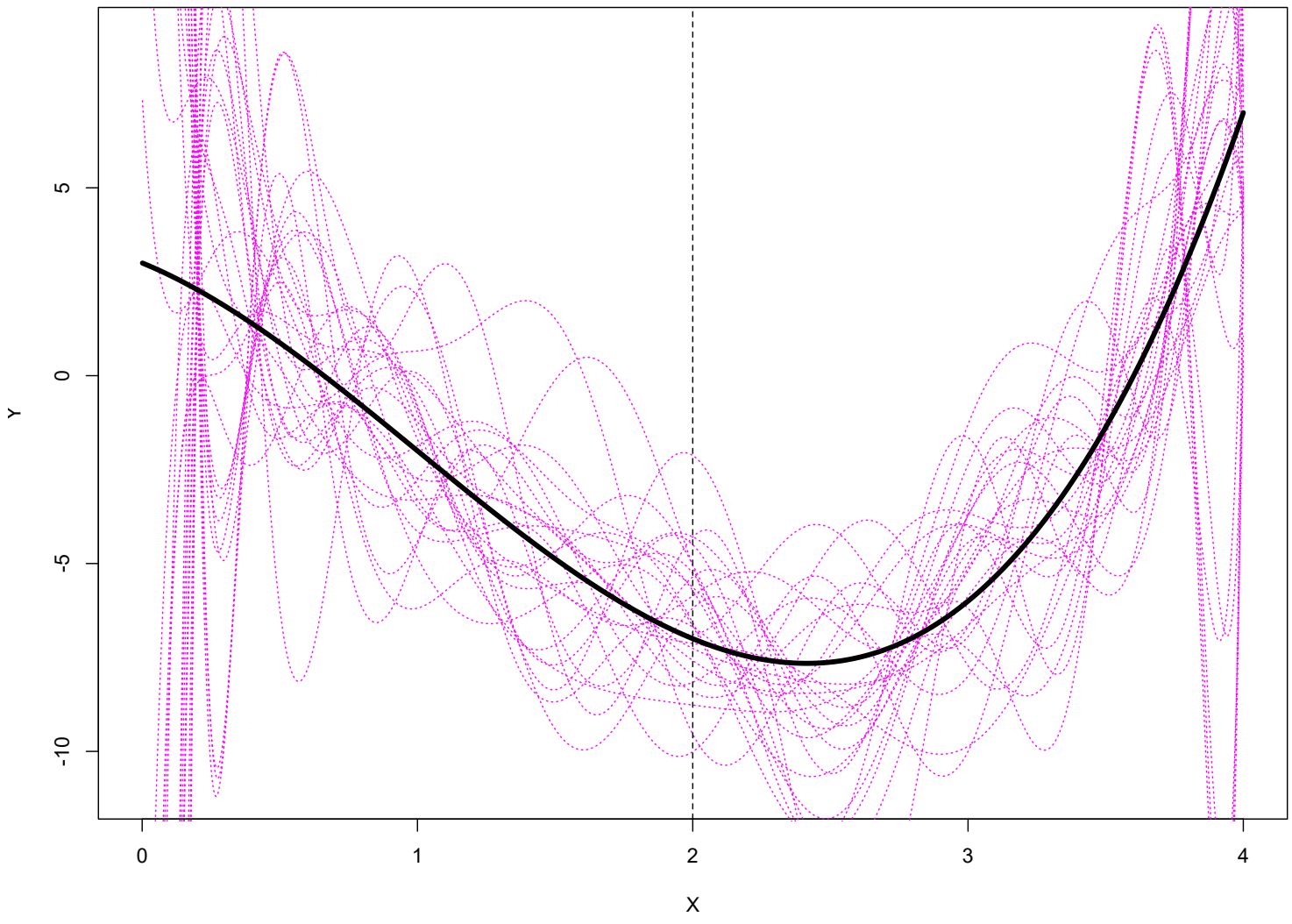

we see that $\widehat{f}(2)$ has high bias but low variance

# Cubic fit for $25$ simulated datasets




no more bias, but variance of $\widehat{f}(2)$ is somewhat higher

# 12-degree polynomial fit for $25$ simulated datasets




still no bias, but the variance of $\widehat{f}(2)$ is now quite high

# Prediction at $X = 2$

For illustration, we only repeated the estimation process $25$ times, so the resulting approximations to the bias and variances are somewhat rough. Generating a larger number of datasets would improve the approximations.

| | Linear | Cubic | 12-degree |
|---|---|---|---|
| $\approx$ Bias$^2$ | 15.79 | 0.27 | 0.87 |
| $\approx$ Variance | 0.27 | 0.67 | 2.07 |
| $\approx$ Reducible Error | 16.06 | 0.94 | 2.94 |

Note that the actual (theoretical) bias in the cubic and 12-degree polynomial models is zero, because both models include all the true predictors and use OLS, which is unbiased.

# Validation and cross-validation

# Approaches to model selection

**Validation set**. Randomly splits the training set into two: one set for training the model, and another for testing the predictive performance and selecting the model complexity (this set is called a *validation* set).

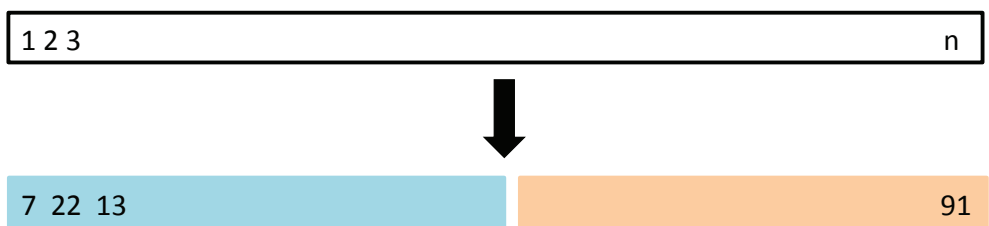
**Cross-validation**. An efficient extension of the validation set approach that is based on multiple splits of the data into training and validation sets.

**Analytical criteria**. Uses analytical results to penalise the training error to account for overfitting.

# Validation set

In the **validation set** approach, we randomly split the *training* data into two parts: another training set and a set we can use for testing the performance of the models (this is the *validation* set).

We select the model with the best predictive performance on the validation set.



## Training, validation, and test split

Now, instead of just training and test data we split our data into 3 parts: training, validation and test. Here is how they are used:

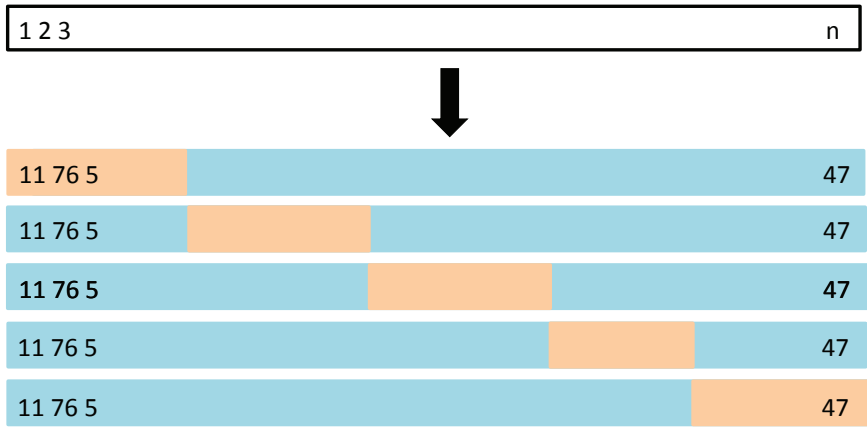
1. Estimate different models on the training data.
2. Make predictions on the validation set.
3. Select the model with the best validation set performance.
4. Re-estimate the selected model by combining the training and validation sets (i.e. going back to the original training data).
5. Assess the performance of the selected model by making predictions on the test data.

# Cross-validation

The validation set approach has serious limitations when the size of the training data is not large. The model may not have enough data to train on, and there may not be enough cases in the validation set to reliably estimate the expected prediction error.

**Cross-validation** methods are based on multiple random training/validation splits. Unlike the validation set approach, each observation gets a turn at being predicted.

# K-fold cross-validation




1. Randomly split the training sample into $K$ **folds** of roughly equal size.
2. For each fold $k \in \{1, \ldots, K\}$, estimate the model on all other folds combined, then use the estimated model to make predictions and compute errors for all observations in fold $k$.
3. The cross-validation error is the average error across all the observations in the training sample.

# K-fold cross-validation

**5-fold and 10-fold CV**. $K = 5$ or $K = 10$ folds are the most common choices for cross-validation.

**Leave one out cross-validation**. If we set $K = n$, this is called leave one out cross-validation, or **LOOCV**. For each observation $i$, we train the model on all other observations, and predict $i$.

# Leave one out CV

**Algorithm** Leave one out CV for regression

1: **for** i=1:n **do**
2: Assign observation $i$ to the validation set.
3: Assign observations $1, \ldots, i-1, i+1 \ldots, n$ to the training set $\mathcal{D}_{-i}$.
4: Estimate the model using the training set $\mathcal{D}_{-i}$.
5: Use the model to compute the prediction $\widehat{f}^{-i}(\boldsymbol{x}_i)$.
6: Compute the squared error $(y_i - \widehat{f}^{-i}(\boldsymbol{x}_i))^2$.
7: **end for**

8: Compute the leave-one-out MSE:

$$\mathsf{MSE}_{\mathsf{CV}_n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}^{-i}(\boldsymbol{x}_i))^2$$

# LOOCV vs K-fold cross-validation

**LOOCV**. Approximately unbiased estimator of the expected prediction error. However, it can have high variance in some settings since the training sets are very similar for every prediction.

Furthermore, it can have a high computational cost as it requires us to fit the model $n$ times (except in special cases).

**K-fold**. Lower computational cost and lower variance. It is subject to some bias because the training sets have sizes smaller than $n$.

# Leave one out CV for linear regression

LOOCV estimate for the test error (reminder):

$$\mathsf{MSE}_{\mathsf{CV}_n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}^{-i}(\boldsymbol{x}_i))^2$$

For the OLS method we can use a *shortcut* to compute the estimate above, without having to refit the model $n$ times:

$$\mathsf{MSE}_{\mathsf{CV}_n} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \widehat{f}(\boldsymbol{x}_i)}{1 - h_i}\right]^2$$

where $\widehat{f}$ is the OLS estimate from the entire training set and $h_i$ is the $i$th diagonal element of the matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ (so all $h_i$ can be easily computed in one go).

# Analytical criteria

# Analytical criteria

Analytical criteria estimate the expected test error based on theoretical arguments. They have the form:

**criterion = training error + penalty for number of parameters**

We select a model that **minimizes** the value of the criterion.

Note that if we did't include a penalty and simply minimized the training error - the selected model would overfit.

# Example: linear regression

It can be shown that in the case of linear regression the expected value of the training MSE underestimates the corresponding expected test MSE by

$$\frac{\sigma^2}{n} 2(p+1)$$

(we will use this fact without going into the proof)

Thus, we could make a correction to the training MSE by adding the above quantity to it. This would allow us to make a fairer comparison of different models.

# Mallow's $C_p$ statistic

The **Mallow's** $C_p$ statistic applies to linear regression. It directly implements the recipe suggested on the previous slide (note that $MSE = RSS/n$).

We select the model with the lowest $C_p$, where

$$C_p = \frac{\text{RSS}}{n} + \frac{\widehat{\sigma}^2}{n} 2(p+1)$$

Here, $\widehat{\sigma}^2$ is an estimate of the variance of the error term in the regression model. This estimate is typically computed using the full model containing all the predictors.

# Akaike Information Criterion

Mallow's $C_p$ can be viewed as a special case of the **Akaike information criterion** (**AIC**), which is a popular and versatile strategy for model selection and applies to models estimated by maximum likelihood (this topic is discussed in the next lecture).

In the case of linear regression with Gaussian errors:

$$\text{AIC} = \frac{1}{\widehat{\sigma}^2}\left(\frac{\text{RSS}}{n} + \frac{\widehat{\sigma}^2}{n} 2(p+1)\right)$$

where $\widehat{\sigma}^2$ is again an estimate of the variance of the error term, typically computed using the full model with all the predictors.

# Mallow's $C_p$ and AIC

Comparing

$$\text{AIC} = \frac{1}{\widehat{\sigma}^2}\left(\frac{\text{RSS}}{n} + \frac{\widehat{\sigma}^2}{n}2(p+1)\right),$$

and

$$C_p = \frac{\text{RSS}}{n} + \frac{\widehat{\sigma}^2}{n}2(p+1)$$

we see that AIC and $C_p$ lead to the same selected model.

For practical purposes, the AIC and $C_p$ are regarded as the same for linear regression.

# Bayesian information criterion

The **Bayesian information criterion** (**BIC**) also applies to models estimated by maximum likelihood, and is derived from a Bayesian point of view (also discussed in the next lecture).

In the special case of linear regression with Gaussian errors, BIC has a very similar form to AIC:

$$\text{BIC} = \frac{1}{\widehat{\sigma}^2}\left(\frac{\text{RSS}}{n} + \frac{\widehat{\sigma}^2}{n}\log(n)(p+1)\right)$$

Thus, BIC is proportional to AIC and $C_p$, but with a $\log(n)$ penalty factor instead of $2$ (thus, typically a heavier penalty on complexity).

# Model selection methods

- LOOCV and AIC generally pick similar models, especially when the sample size is large.
- The advantage of AIC over LOOCV is mainly computational.
- Cross-validation is universally applicable, while this is not the case for AIC.
- Cross-validation should be preferred to AIC when the assumptions of the model (e.g. constant error variance) are likely to be wrong.

# Limitations of model selection

- Standard statistical inference (e.g. confidence intervals, p-values) is no longer valid after model selection.
- This is because standard inference assumes a fixed model, whereas model selection will by definition pick a specific model that fits the data well.
- In our context the way around this difficulty would be data splitting: using one part of the sample for model selection, and another for inference.

# Review questions

- What is a sampling distribution?
- What are the benefits of assuming a Gaussian MLR model?
- How does the KNN method compute predictions?
- What is the curse of dimensionality?
- Write and interpret the bias-variance decomposition for a KNN prediction.
- What are the advantages and disadvantages of the KNN method?

# Review questions

- How does model selection relate to the bias-variance trade-off?
- What is a validation set? How is it different from a test set?
- What is K-Fold cross validation? Describe how it works.
- What is the motivation for the $C_p$ criterion and how does this criterion relate to AIC and BIC in the case of linear regression?