QBUS6810: Statistical Learning and Data Mining

Lecture 3: Regression Modelling

Semester 1, 2019

Discipline of Business Analytics, The University of Sydney Business School

Lecture 3: Linear Regression and K-Nearest Neighbours

1. Derivation of the formula for the OLS estimator
2. Statistical properties of OLS
3. The Gaussian MLR model
4. K-Nearest Neighbours
5. Comparison with linear regression

Derivation of the formula for the OLS estimator

Least squares (reminder)

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

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

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.

MLR model and linear algebra

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

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

This can be neatly written in matrix form as:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \qquad \text{where}$$

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} \quad \boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix} \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \vdots \\ \beta_p \end{pmatrix} \quad \boldsymbol{\epsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_i \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Least squares and linear algebra

For any candidate coefficient vector $\tilde{\boldsymbol{\beta}}$, we have:

$$\boldsymbol{X}\tilde{\boldsymbol{\beta}} = \begin{pmatrix} \tilde{\beta}_0 + \tilde{\beta}_1 x_{11} + \tilde{\beta}_2 x_{12} + \ldots + \tilde{\beta}_p x_{1p} \\ \tilde{\beta}_0 + \tilde{\beta}_1 x_{21} + \tilde{\beta}_2 x_{22} + \ldots + \tilde{\beta}_p x_{2p} \\ \vdots \\ \tilde{\beta}_0 + \tilde{\beta}_1 x_{n1} + \tilde{\beta}_2 x_{n2} + \ldots + \tilde{\beta}_p x_{np} \end{pmatrix}$$

and hence the vector of residuals can be written as:

$$\begin{pmatrix} y_1 - [\tilde{\beta}_0 + \tilde{\beta}_1 x_{11} + \tilde{\beta}_2 x_{12} + \ldots + \tilde{\beta}_p x_{1p}] \\ y_2 - [\tilde{\beta}_0 + \tilde{\beta}_1 x_{21} + \tilde{\beta}_2 x_{22} + \ldots + \tilde{\beta}_p x_{2p}] \\ \vdots \\ y_n - [\tilde{\beta}_0 + \tilde{\beta}_1 x_{n1} + \tilde{\beta}_2 x_{n2} + \ldots + \tilde{\beta}_p x_{np}] \end{pmatrix} = \boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{\beta}}$$

The RSS is the sum of squared elements of this vector

Least squares and linear algebra

The RSS is the sum of squared elements of the residual vector, i.e., its squared norm:

$$\text{RSS}(\tilde{\boldsymbol{\beta}}) = \|\boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{\beta}}\|^2$$

The OLS estimator, $\widehat{\beta}$, achieves the minimum value of RSS.

We derive the formula for $\widehat{\beta}$ by minimising the RSS. More specifically, we: (a) take $p+1$ partial derivatives with respect to the $\tilde{\beta}$ coefficients, (b) set these derivatives equal to zero, and (c) solve the corresponding equations for $\widehat{\boldsymbol{\beta}}$.

The full derivation is posted on Canvas as optional material.

OLS estimator

As a result of the derivation, we get the following formula for the OLS estimator of the linear regression coefficient vector $\boldsymbol{\beta}$:

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

where

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

Statistical properties of OLS

Sampling distribution of an estimator

In classical statistics, the population parameter β is fixed and the data is a random sample from the population. We estimate β 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 $\widehat{\boldsymbol{\beta}}$ and obtain a set of estimates $\{\widehat{\boldsymbol{\beta}}(\mathcal{D}^{(s)})\}_{s=1}^{S}$. The sampling distribution can be thought of as the distribution of these estimates as we let $S \to \infty$.

Note that this concept refers to hypothetical datasets rather than the one dataset that have in practice.

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:

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

We will derive the first formula in tutorial 6 and use the second formula without proof

Variance

Consider (without proof) the following more interpretable expression for the variance of the individual coefficients:

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

For each j, we use R_j^2 to denote the R-squared from the linear regression of predictor j on all other predictors.

$s_{x_j}^2$ denotes 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, σ^2.
- The variance of $\widehat{\beta}_j$ decreases as n increases.
- The higher the correlation (measured by R_j^2) 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 (σ^2).

An unbiased estimator of σ^2 is:

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

When we replace σ 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 ith individual is:

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

Consider (without proof) the following formula for the average variance of predictions on the training set:

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

Think about how each component (n, σ^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 ε 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 ε has Normal distribution with mean 0 and variance σ^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:

estimate $\pm$ critical value $\times$ standard error

An approximate $100 \times (1-\alpha)\%$ confidence interval for β_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.

Note: in the most common case of the 95% CI we have $t_{n-p-1,\alpha/2} \approx 2$ 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 $\boldsymbol{x}$:

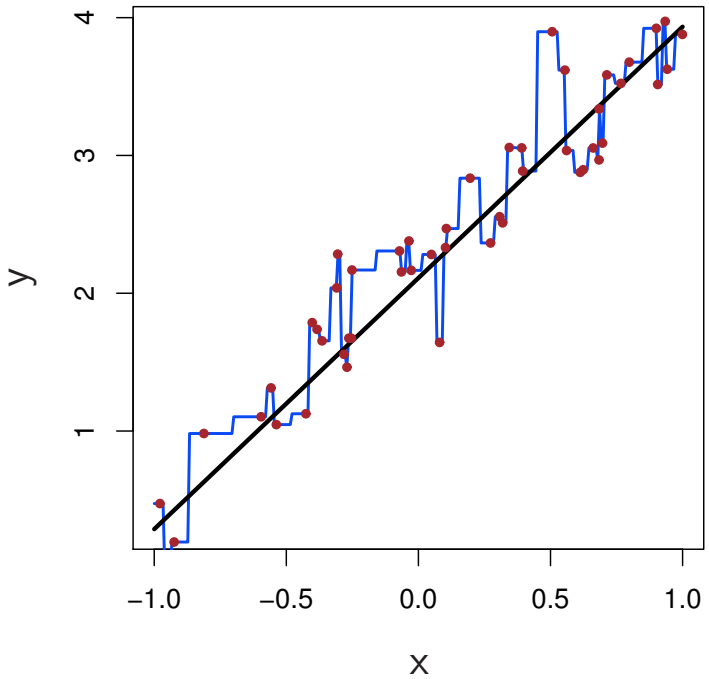
$$\widehat{f}(\boldsymbol{x}) = \text{Average}\Big[\, y_i \,\Big|\, \boldsymbol{x}_i \text{ is in } N_k(x) \,\Big]$$

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 $k = 9$

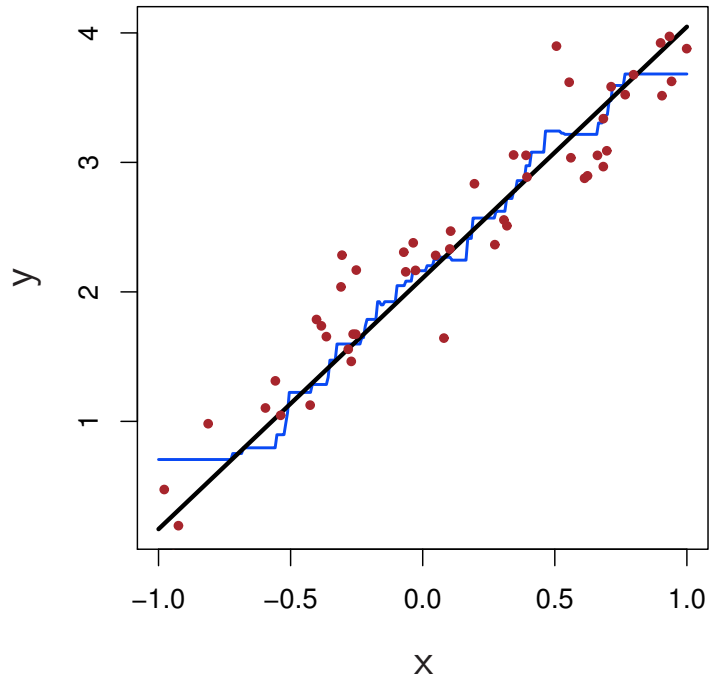


Figure from ISL

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 approximate any reasonable regression function f without making assumptions about the specific form (e.g. linearity) 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. The 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.
- If we simply chose the value of k based on the training error, we would always choose $k = 1$ and fit the data perfectly!
- Instead, we select k based on bias-variance trade-off considerations.
- Next week we will use model selection and 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

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

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

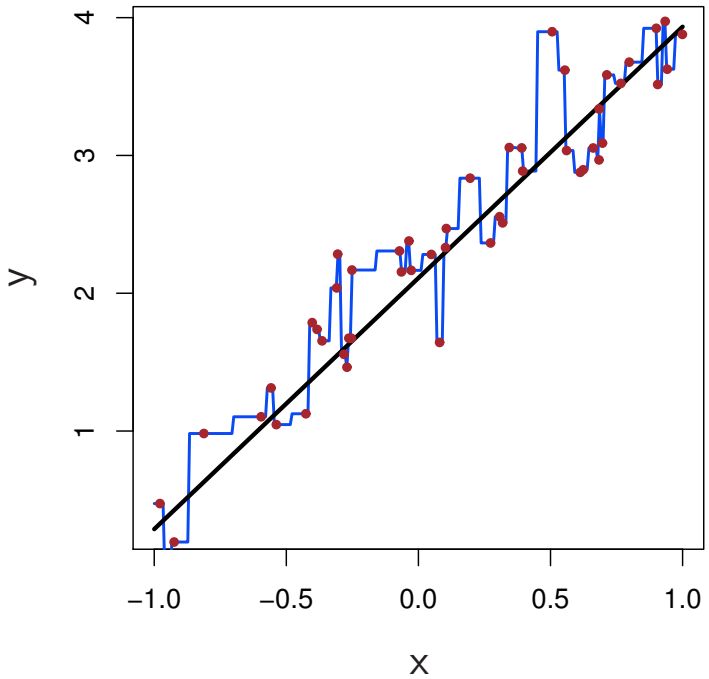
Comparison with linear regression

- 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 $k = 9$

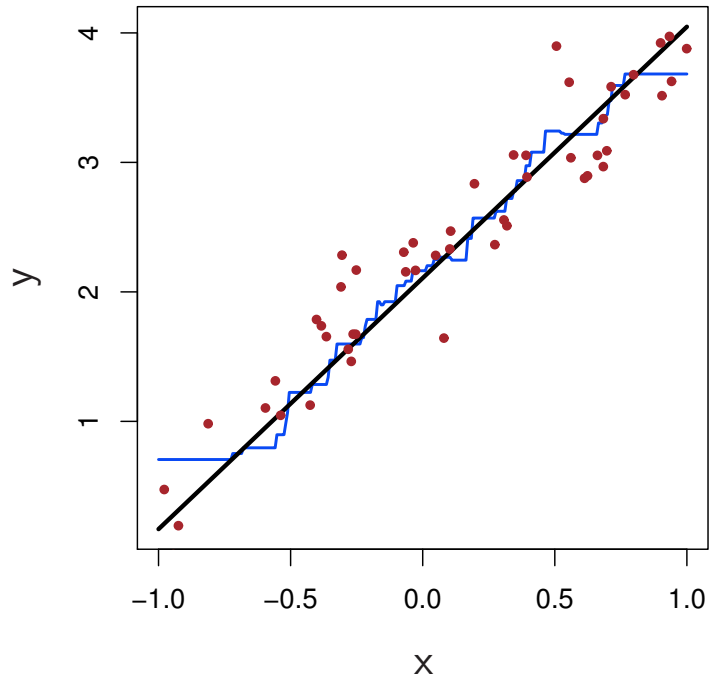


Figure from ISL

Linear regression and k-NN: Illustration

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

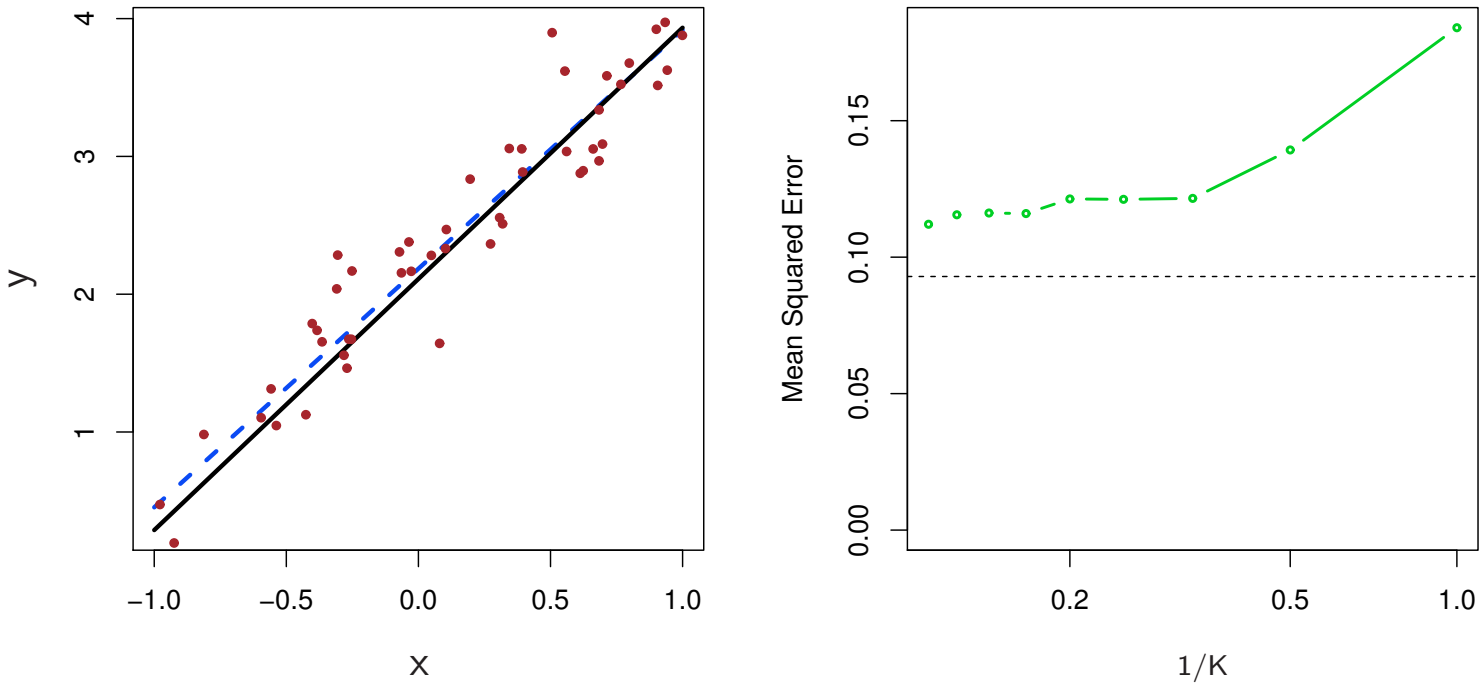


Figure from ISL

Same as before, but a nonlinear true regression function

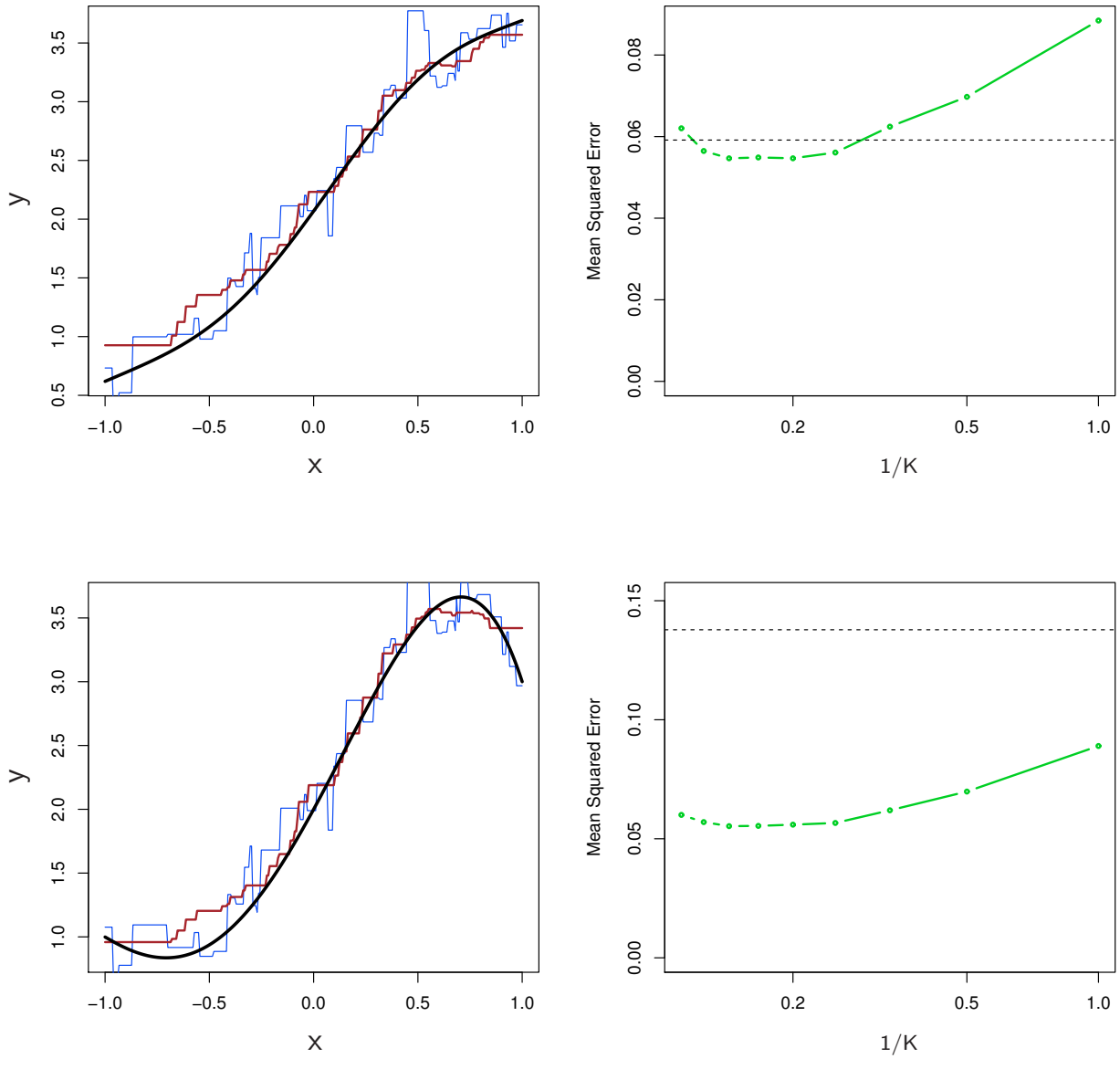


Figure from ISL

Many predictors

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

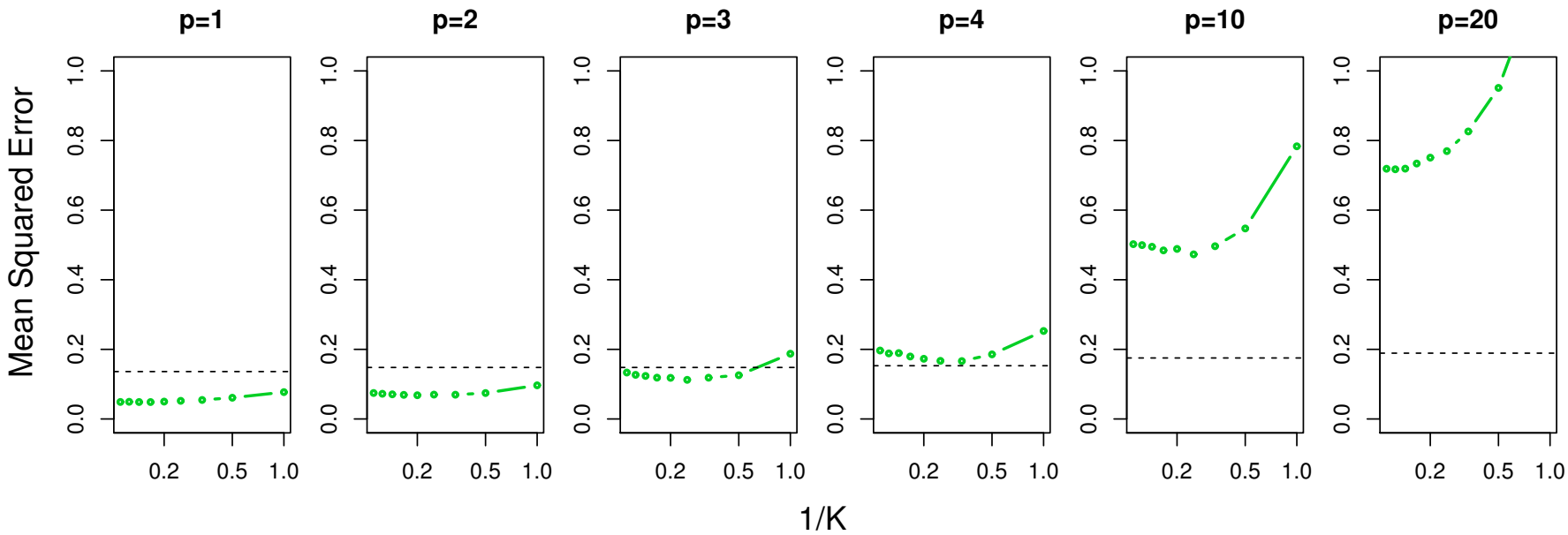


Figure from ISL

Summary

- KNN algorithm is a highly flexible method that does not explicitly assume a form for the regression function $f(X)$.
- A small value of k provides the most flexible model, with low bias but high variance.
- A larger value of k provides smoother estimates, at the cost of a less flexible approximation.

Summary

However, the KNN method has disadvantages that we need to keep in mind:

- The estimate of the regression function can be highly unstable as it is an average of only a few points. This is the price that we pay for flexibility.
- Curse of dimensionality.
- Generating predictions is computationally expensive.

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?