CAP 5625: Computational Foundations for Artificial Intelligence

Linear regression, nonlinear regression, and model selection

Models that preserve the additive assumption

When discussing linear regression, we introduced the concept of adding dummy variables to condition on qualitative inputs.

For example, we considered a model where we wished to predict credit balance Y from an individual's income X_1, conditioning on their student status X_2, with

$$X_2 = \begin{cases} 1 & \text{student} \\ 0 & \text{non-student} \end{cases}$$

and the resulting regression model as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 = \begin{cases} (\beta_0 + \beta_2) + \beta_1 X_1 & \text{student} \\ \beta_0 + \beta_1 X_1 & \text{non-student} \end{cases}$$

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

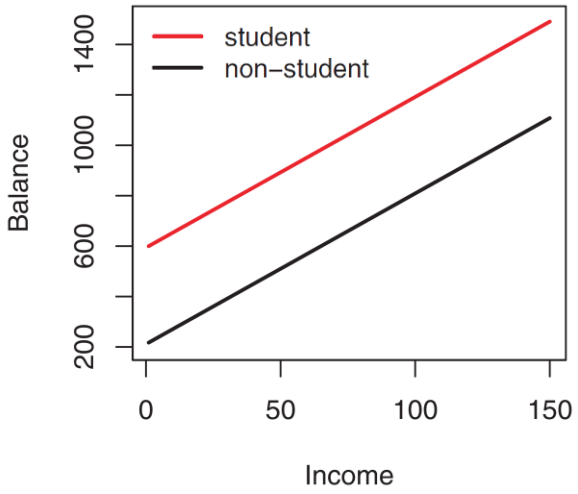
Models that preserve the additive assumption

The regression model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 = \begin{cases} (\beta_0 + \beta_2) + \beta_1 X_1 & \text{student} \\ \beta_0 + \beta_1 X_1 & \text{non-student} \end{cases}$$

assumes **additivity** (or linearity) between the inputs (features) X_1 and X_2 and the output (response) Y.

Specifically, the effect of feature X_j on response Y is independent of other features.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

The model is still our simple linear regression model

If we define the input vector

$$X = \begin{bmatrix} 1 \\ X_1 \\ X_2 \end{bmatrix}$$

and the coefficient (parameter) vector as

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

then the regression model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$ is our original linear regression model

$$Y = X^T \beta$$

Models that remove the additive assumption

When discussing linear regression, we considered an example model in which additivity was violated.

Recall that we considered a model where we wished to predict credit balance Y from an individual's income X_1, conditioning on their student status X_2, with

$$X_2 = \begin{cases} 1 & \text{student} \\ 0 & \text{non-student} \end{cases}$$

and the resulting regression model as

$$Y = \beta_0 + \beta_1 X_1 + (\beta_2 + \beta_3 X_1) X_2 = \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_1 & \text{student} \\ \beta_0 + \beta_1 X_1 & \text{non-student} \end{cases}$$

which allows the effect of income X_1 on Y to change depending on whether an individual was a student.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

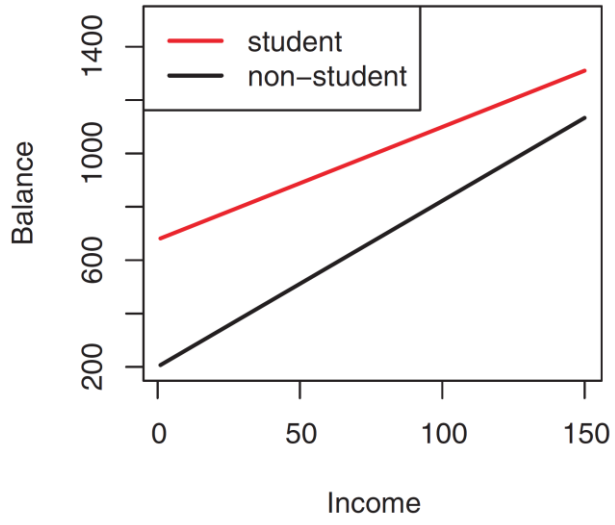
Models that remove the additive assumption

Expanding, we can write

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$$

where we have now included a third parameter β_3, which models how income X_1 and student status interact to affect credit balance Y.

This model removes the additivity assumption, because the effect of feature X_1 on Y is not independent of feature X_2 (and vice versa).



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

The model is still our simple linear regression model

If we define the input vector

$$X = \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ X_1 X_2 \end{bmatrix}$$

and the coefficient (parameter) vector as

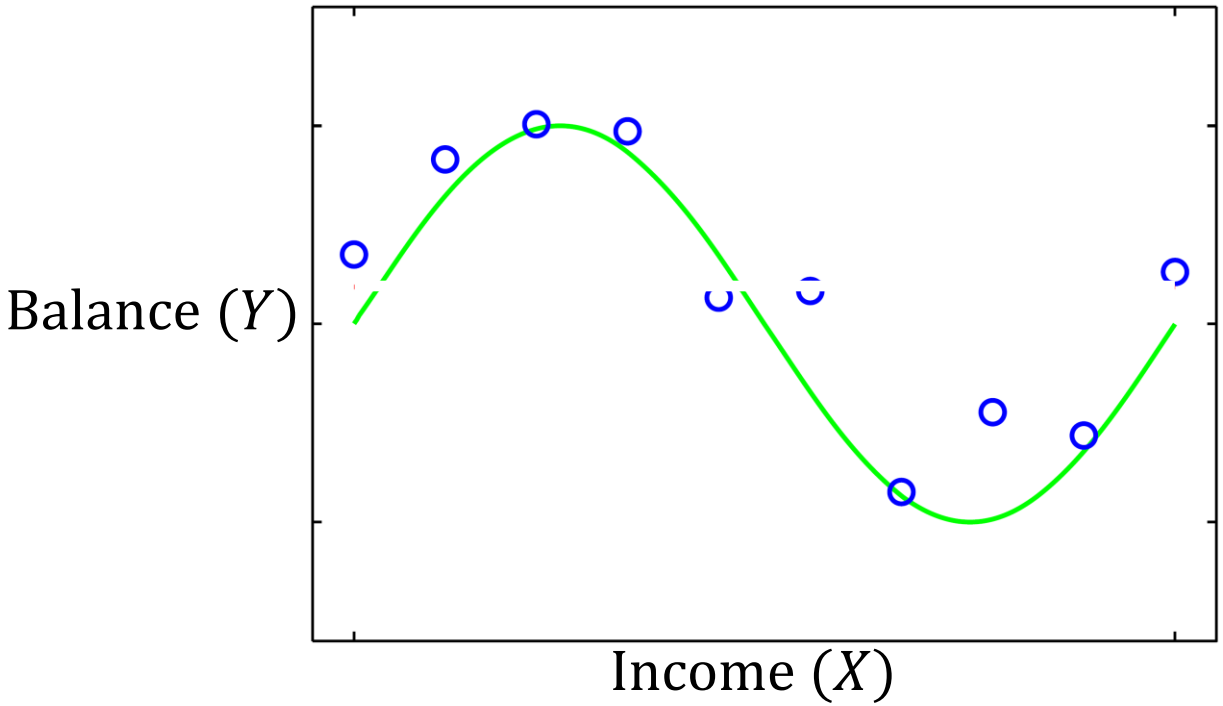
$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

then the regression model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$ is our original linear regression model

$$Y = X^T \beta$$

Modeling non-linear relationships

Suppose the relationship between credit balance Y and income X is nonlinear, as illustrated below as a green curve $Y = \sin(2\pi X)$.



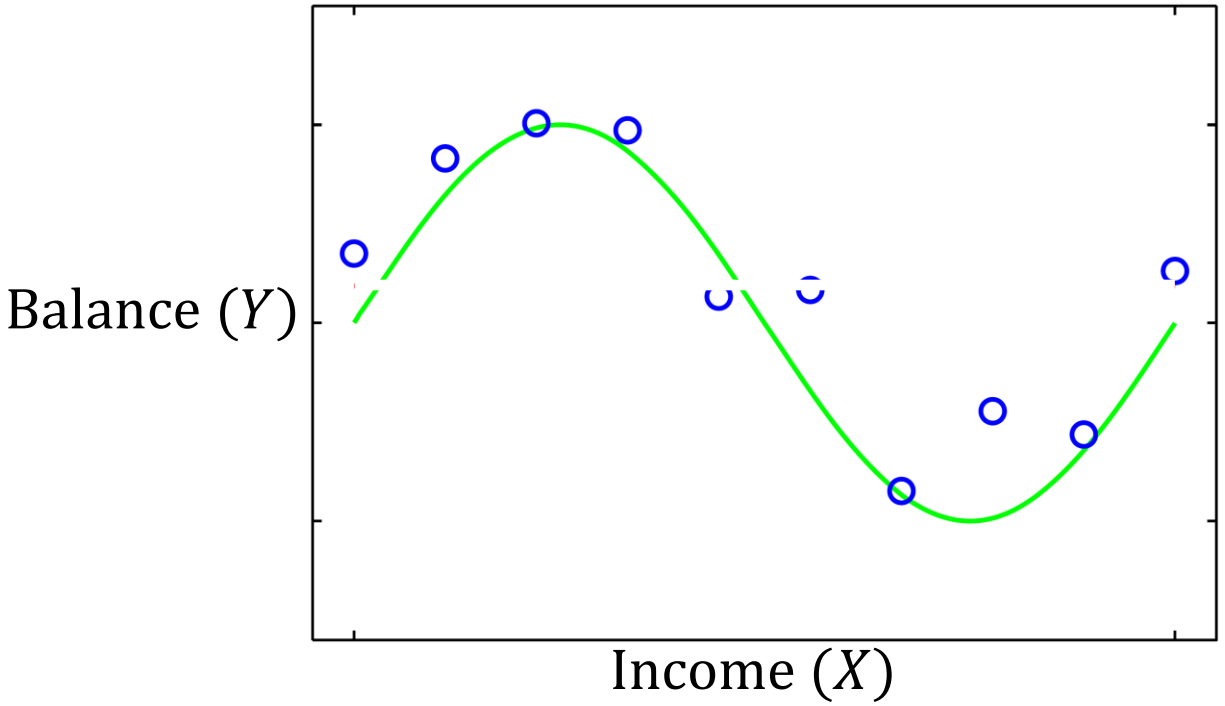
How might we model such a relationship? We could try letting balance Y be a polynomial function of income X.

Modified from Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Modeling output as a polynomial function of input

Specifically, we could let Y be written as a polynomial function of X of degree M, $M = 0,1, \ldots$, which is represented as

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_M X^M = \sum_{m=0}^{M} \beta_m X^m$$



Modified from Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

The model is still our simple linear regression model

If we define the input vector

$$X = \begin{bmatrix} 1 \\ X \\ X^2 \\ \vdots \\ X^M \end{bmatrix}$$

and the coefficient (parameter) vector as

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}$$

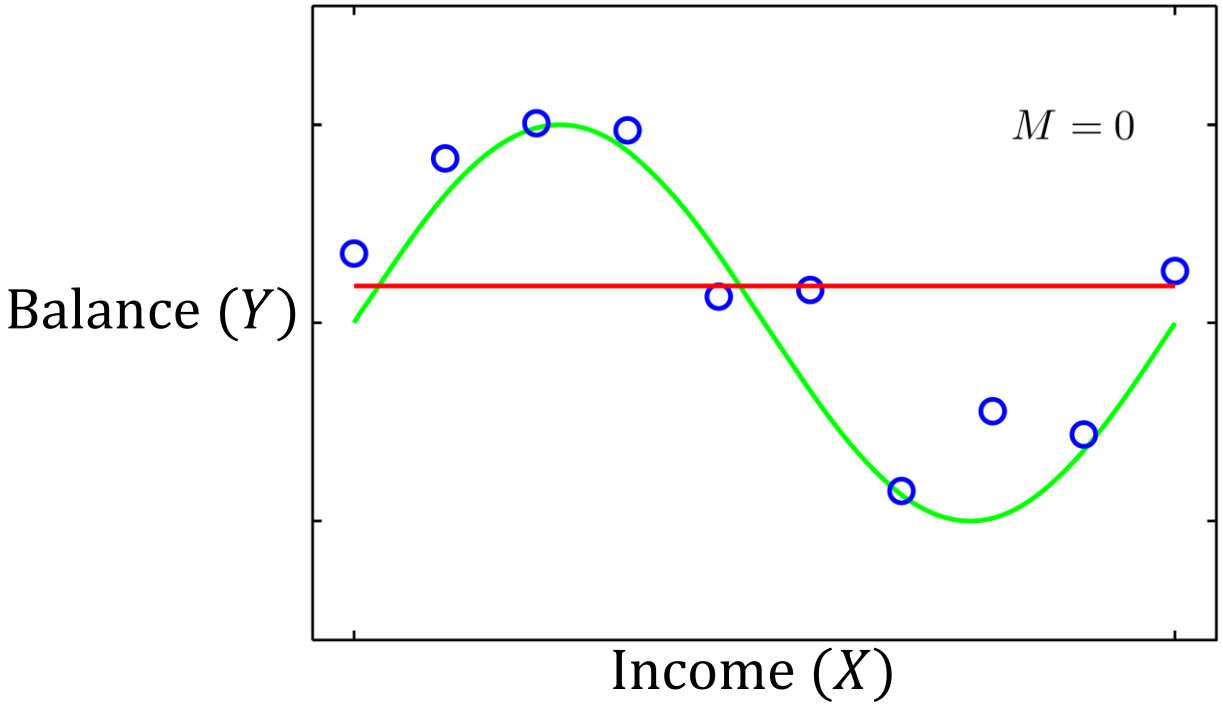
then the regression model $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_M X^M$ is our original linear regression model

$$Y = X^T \beta$$

Poor fit with a constant function (zero degree)

Letting the degree be $M = 0$, we fit a horizontal line (red), which demonstrates a poor fit to the data (blue circles)

$$Y = \beta_0$$

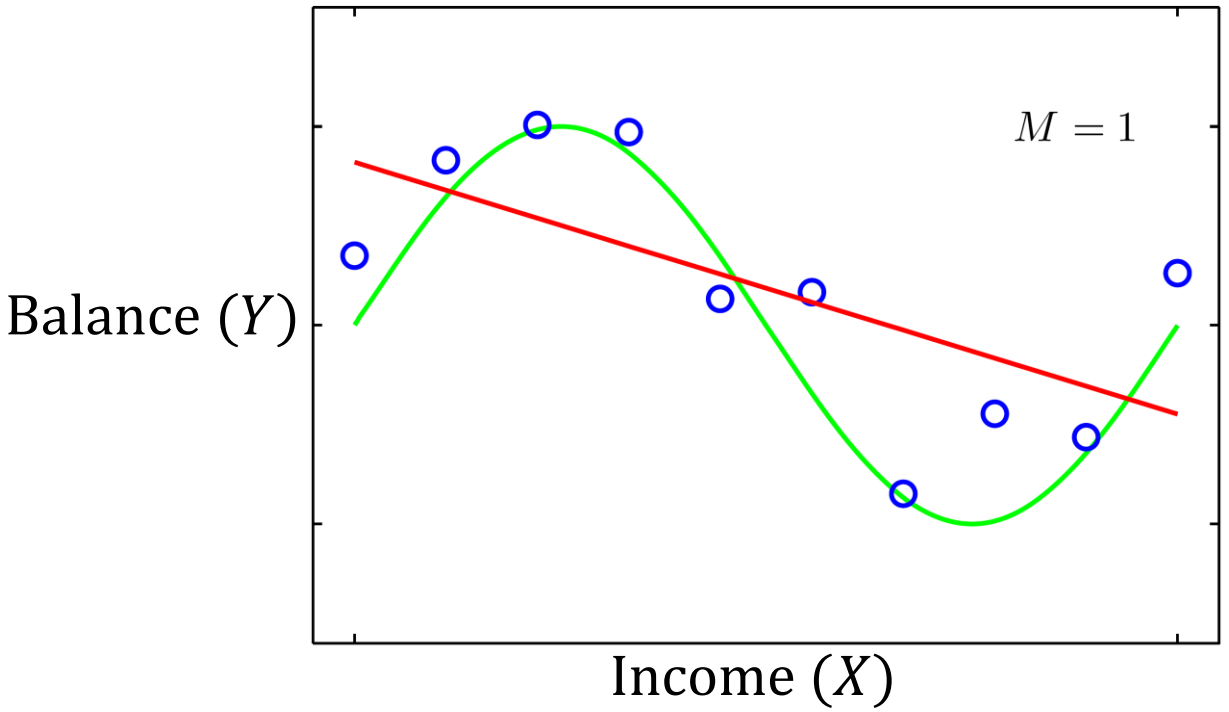


Modified from Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Poor fit with a linear function (first degree)

Letting the degree be $M = 1$, we fit a line with a non-zero slope (red), which also demonstrates a poor fit to the data, as the Y is nonlinear with respect to X

$$Y = \beta_0 + \beta_1 X$$

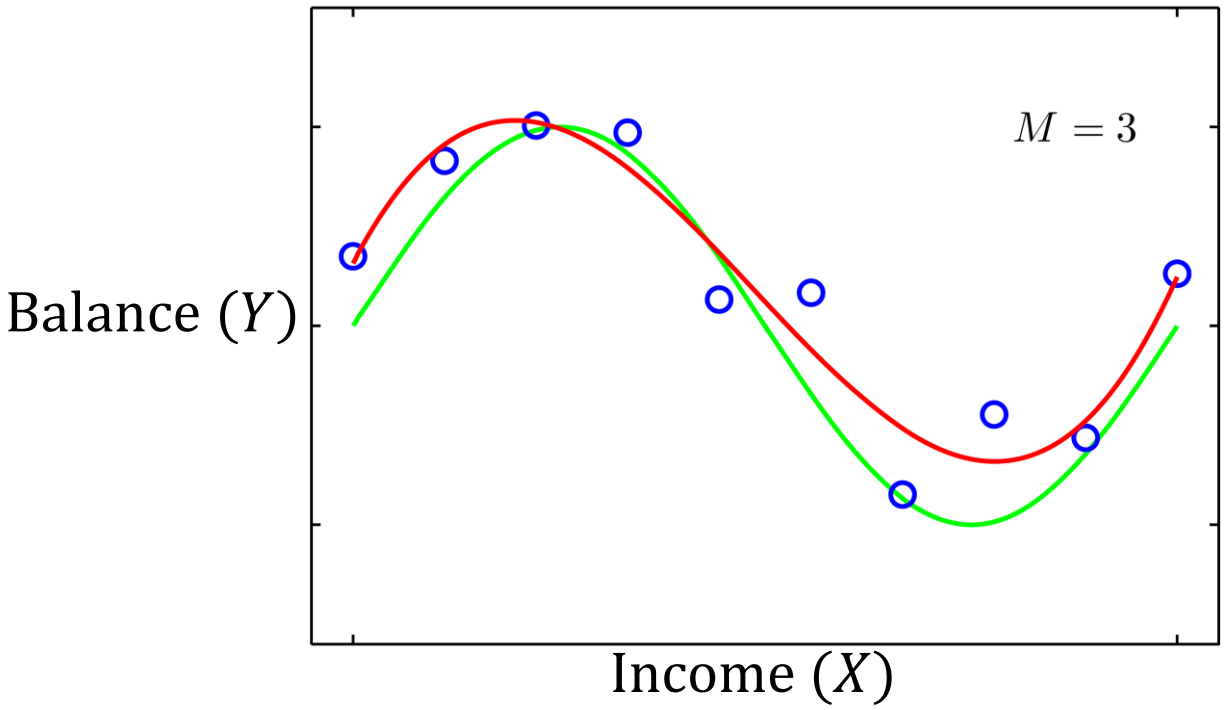


Modified from Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Good fit with a third-degree polynomial

Letting the degree be $M = 3$, we fit a curve (red), which demonstrates a good fit to the data, and models the oscillations of the function that generated the data

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$$

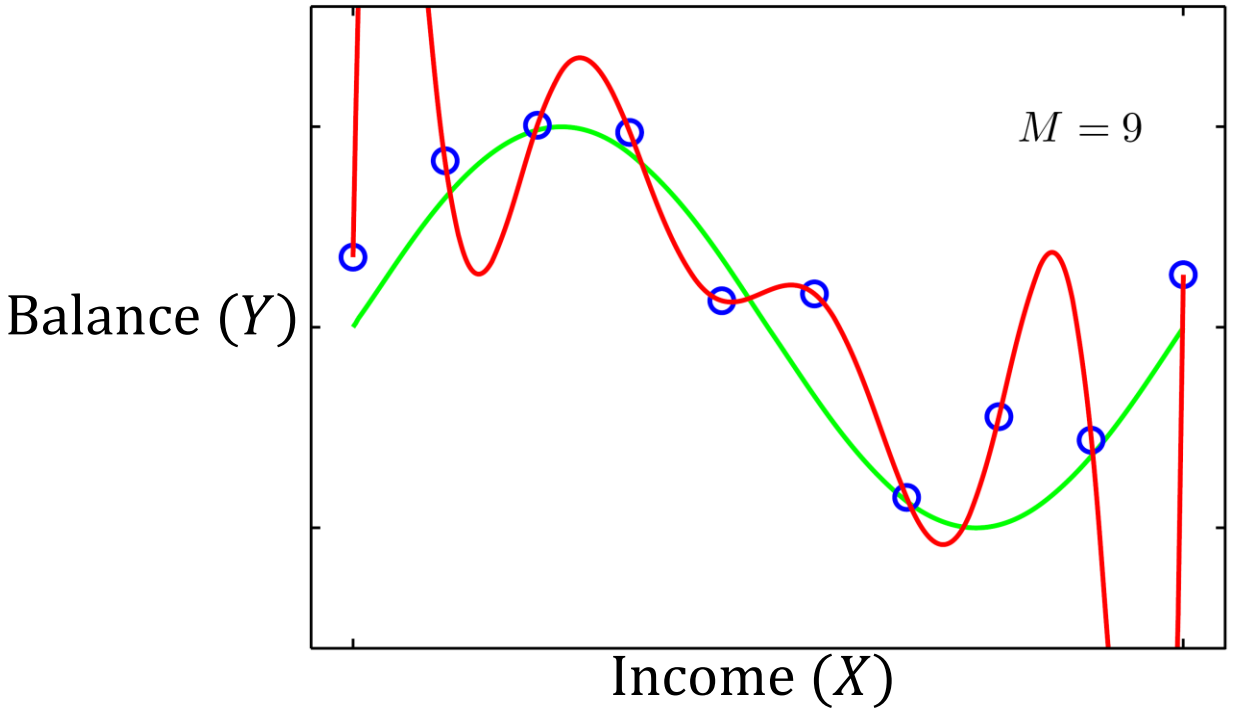


Modified from Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Poor fit with a ninth-degree polynomial

Letting the degree be $M = 9$, we fit a curve (red), which demonstrates a poor fit to the data, as we are now **overfitting** by modeling every training example exactly

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \cdots + \beta_9 X^9$$



Modified from Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Training and test error

We can evaluate the prediction error of our trained model either on the training dataset (known as **training error**) or on a dataset not used for training the model (known as **test error**).

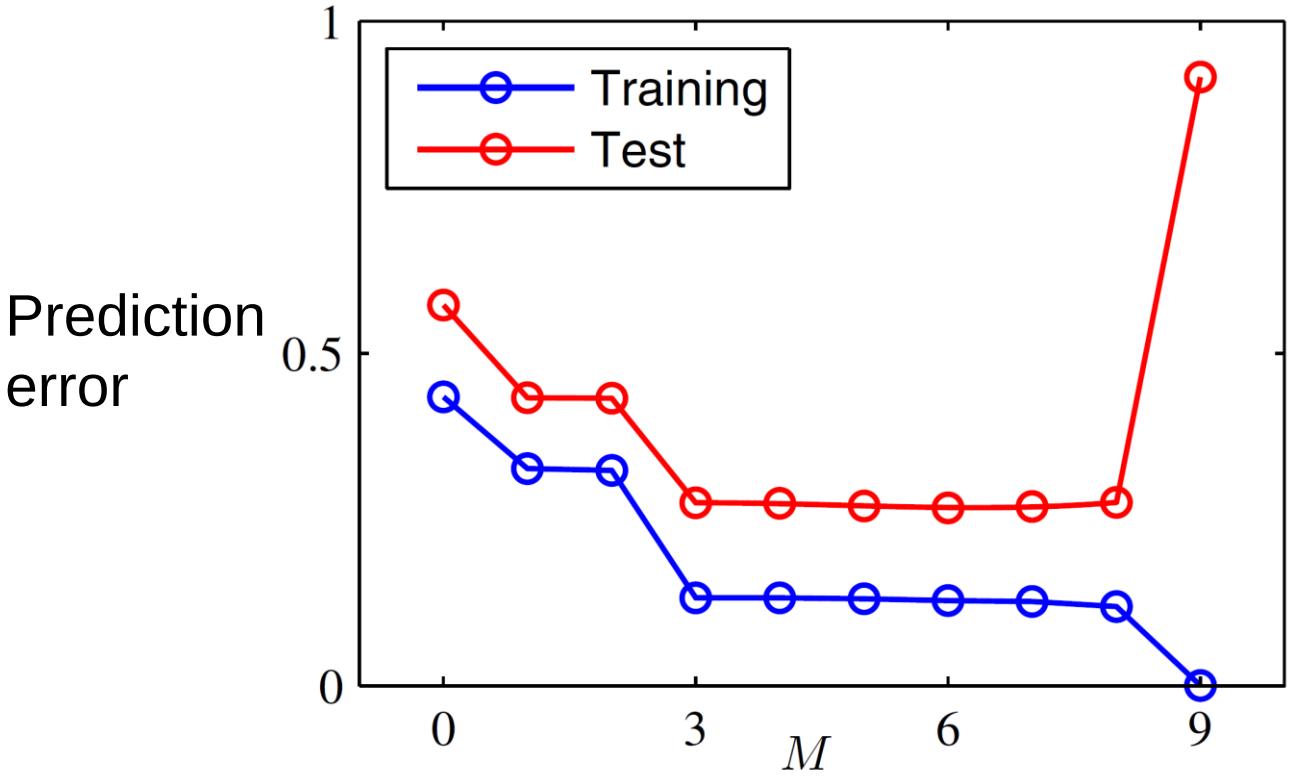
When training a regression model, the algorithm's goal is to minimize the training error.

However, what we really want is not to minimize the training error, but to minimize the test error, or the prediction error on samples we have not observed yet.

Overfitting is an issue, because we can begin modeling noise in the training data rather than only the underlying function.

Model fit as a function of polynomial degree M

Below is prediction error as a function of polynomial degree M for both training and non-training (test) datasets.

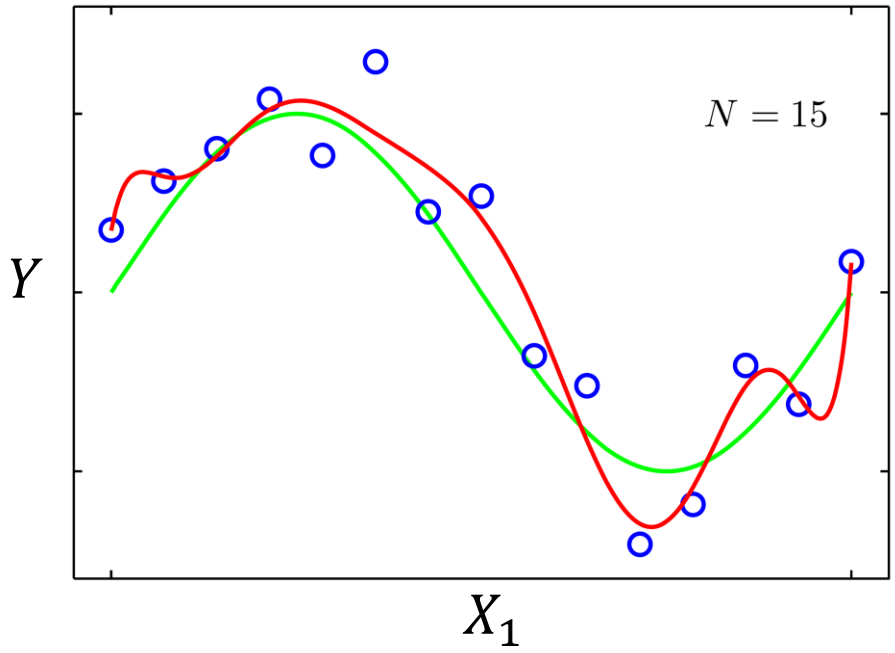


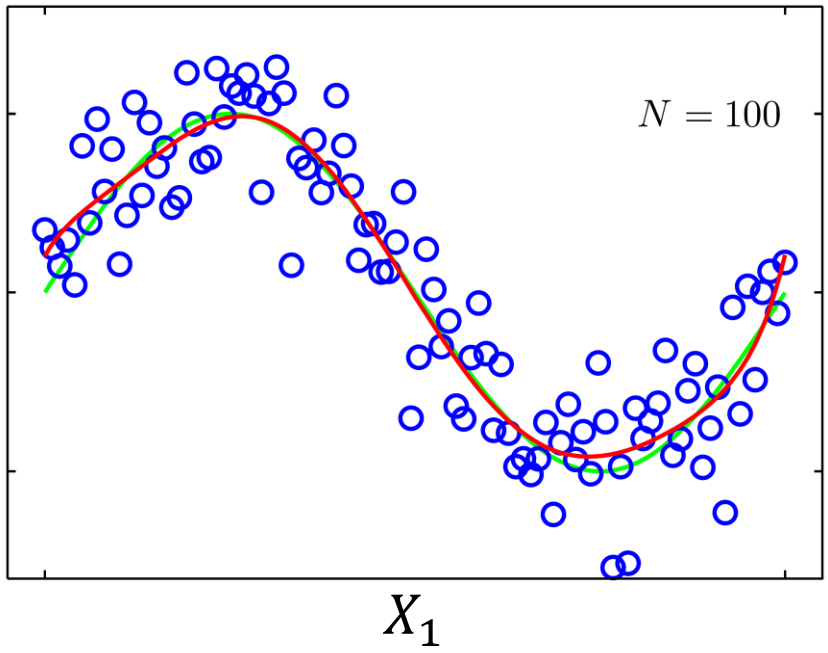
Training error always decreases with increasing model complexity, whereas test error has a U-shaped curve.

Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Size of the training dataset affects overfitting

For the same polynomial function of degree $M = 9$, the fit with $N = 100$ training observations is more robust compared to $N = 15$.



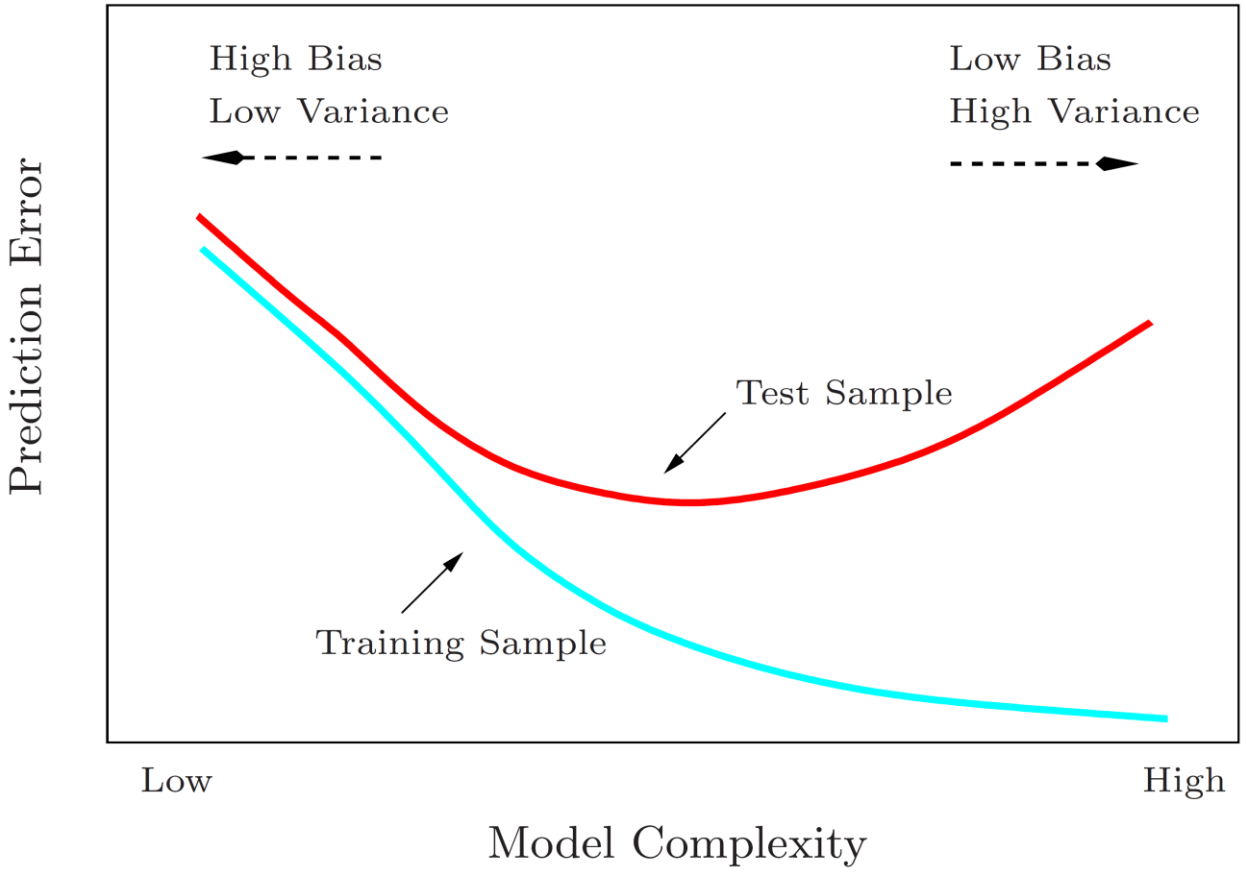


Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Bias-variance tradeoff for model complexity

Low complexity models: systematic prediction errors (high bias), but not overly sensitive to training data (low variance).

High complexity models: fit training data well (low bias), but overly sensitive to training data (high variance).



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

How are bias and variance related to model fit?

Consider the collection of p input features $X^T = [X_1, X_2, \ldots, X_p]$.

We have assumed that $Y = f(X) + \epsilon$, where

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

and predict $\hat{Y} = \hat{f}(X)$ from the model fit to training data as

$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$$

We can measure how well the model fits the data using the mean squared error (MSE), defined as

$$\mathrm{MSE}\left(\hat{f}\right) = \left(\mathrm{Bias}(\hat{f})\right)^2 + Var\left(\hat{f}\right) + Var(\epsilon)$$

What are the meanings of these components?

$$\mathrm{MSE}(\hat{f}) = \left(\mathrm{Bias}(\hat{f})\right)^2 + Var(\hat{f}) + Var(\epsilon)$$

$Var(\hat{f})$ is mean of the squared difference between the estimator $\hat{f}$ and its average.

$\mathrm{Bias}(\hat{f})$ is the difference between the true value f and the mean of $\hat{f}$.

$\left(\mathrm{Bias}(\hat{f})\right)^2 + Var(\hat{f})$ is known as the reducible error.

$Var(\epsilon)$ is known as the irreducible error.

First two terms are under our control, but third is not.

Therefore, we seek a model that minimizes the reducible error.

What if the linear model is true?

If the linear model is true, then we have that

$$E(Y|X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j = X^T \beta$$

The least squares estimate of β is unbiased. That is, for $j = 0,1, \ldots, p$,

$$E(\hat{\beta}_j) = \beta_j$$

Moreover, this implies that $\text{Bias}\left(\hat{f}(X)\right) = 0$, and therefore

$$\text{MSE}(\hat{f}) = Var(\hat{f}) + Var(\epsilon)$$

We therefore only need to minimize the variance of the estimator.

Gauss-Markov theorem

Assume linear model is true, and $\hat{\beta}$ was estimated by least squares.

Therefore, $\mathrm{Bias}(\hat{f}) = 0$ for linear model $\hat{f}(X) = X^T\hat{\beta}$.

Assume $\epsilon \sim N(0, \sigma^2)$, and that they are independent across samples.

Then the linear estimator $\hat{f}(X) = X^T\hat{\beta}$ has minimum variance of all other unbiased linear estimators.

Therefore, because

$$\mathrm{MSE}(\hat{f}) = Var(\hat{f}) + Var(\epsilon)$$

the least squares estimate has minimum MSE of all other linear unbiased estimators.

Is this the best we can do?

The least squares linear estimator provided unbiased predications.

But do we always want unbiased estimators with minimum variance?

Recall that $\mathrm{MSE}(\hat{f}) = \text{Reducible Error} + \text{Irreducible Error}$, where

$$\text{Reducible Error} = Var(\hat{f}) + \left[\mathrm{Bias}(\hat{f})\right]^2$$

Therefore, we may be able to obtain better prediction accuracy with a biased estimator, provided that the reduction of the variance is larger than the squared bias.

Many modern machine learning techniques make this tradeoff.

Computing training MSE (training error)

Suppose we have training dataset (x_i, y_i), $i = 1,2, \dots, N$. The training MSE (or training error) is given by

$$\text{Training MSE}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{f}(x_i)\right)^2$$

which for a linear model is

$$\text{Training MSE}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - x_i^T\hat{\beta}\right)^2$$

Here we are interested in how well the estimated function can predict the training observations, *i*.e., for $i = 1,2, \dots, N$,

$$y_i \approx \hat{f}(x_i)$$

Computing test MSE (test error)

We are generally more interested in whether the estimated function can predict a previously-unseen test observation (x_0, y_0), *i.e.*,

$$y_0 \approx \hat{f}(x_0)$$

Instead, we can compute the test MSE (test error) as

$$\text{Test MSE}\left(\hat{f}\right) = \frac{1}{\#\text{ test obs.}} \sum_{\text{test obs.}\,(x_0,y_0)} \left(y_0 - \hat{f}(x_0)\right)^2$$

which for a linear model is

$$\text{Test MSE}\left(\hat{f}\right) = \frac{1}{\#\text{ test obs.}} \sum_{\text{test obs.}\,(x_0,y_0)} \left(y_0 - x_0^T\hat{\beta}\right)^2$$

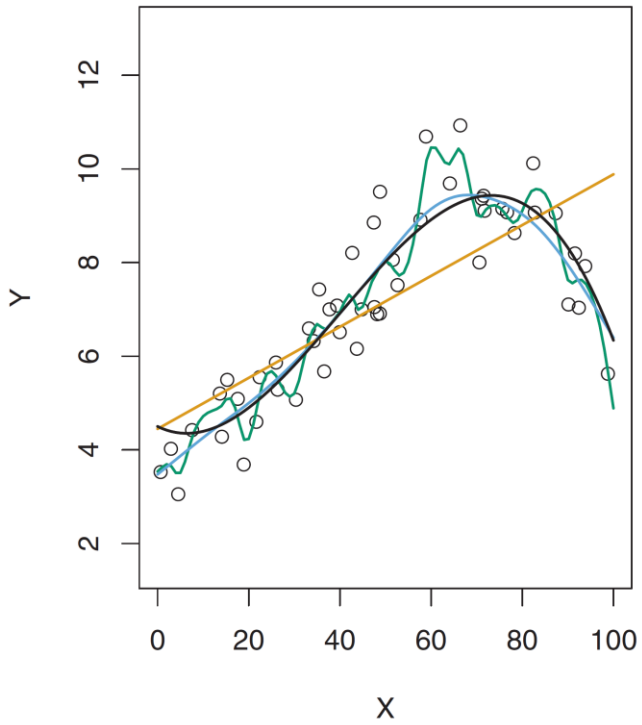
Minimum Test MSE cannot fall below the irreducible error.

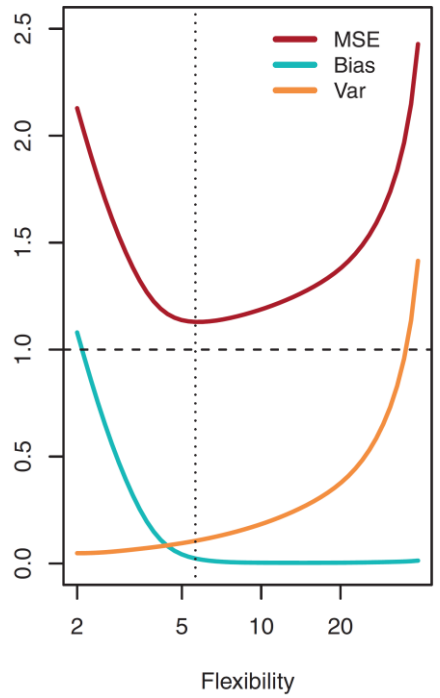
Model fit as a function of model complexity

When true function (black) is non-linear, a linear fit (orange) will have high bias and low variance, compared to those of increasing complexity (blue and green).

Test MSE never falls below irreducible error (horizontal dotted line).

High complexity models can shrink training MSE to 0.





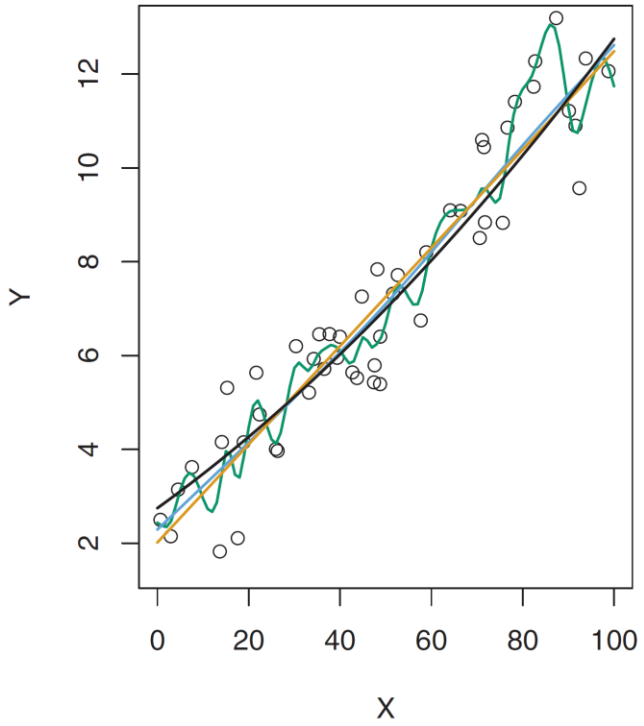
James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

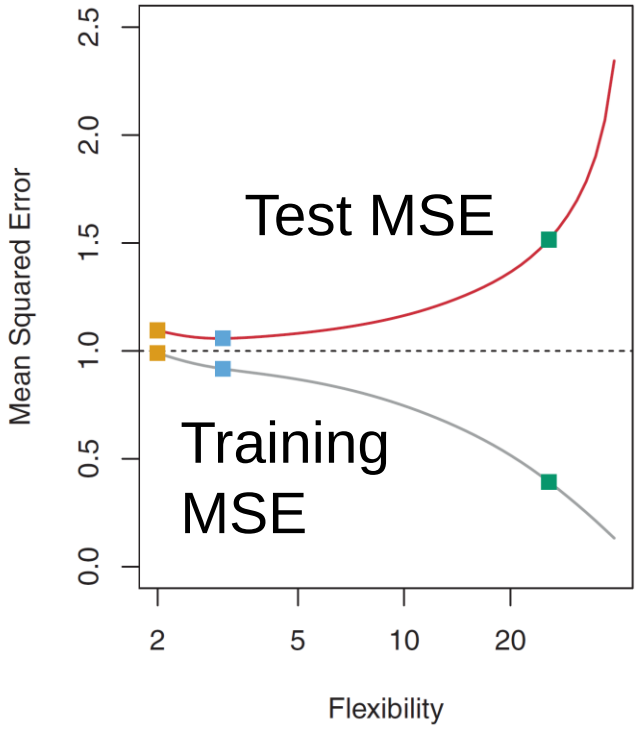
Model fit as a function of model complexity

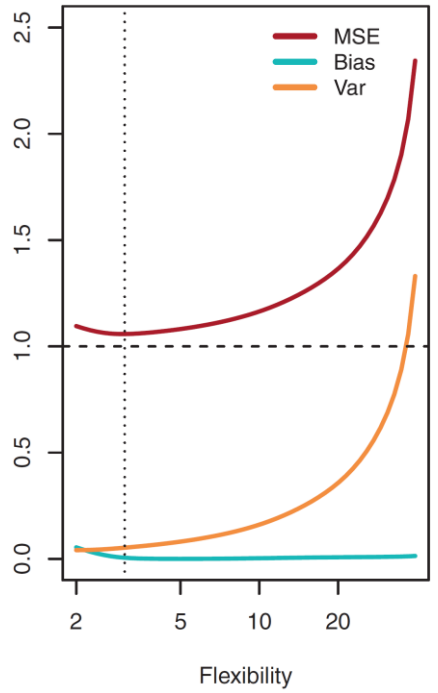
When true function (black) is close to linear, a linear fit (orange) will have both low bias and low variance.

Test MSE never falls below irreducible error (horizontal dotted line).

Bias of high complexity models remains low, but variance (and therefore MSE) is inflated compared to linear fit.







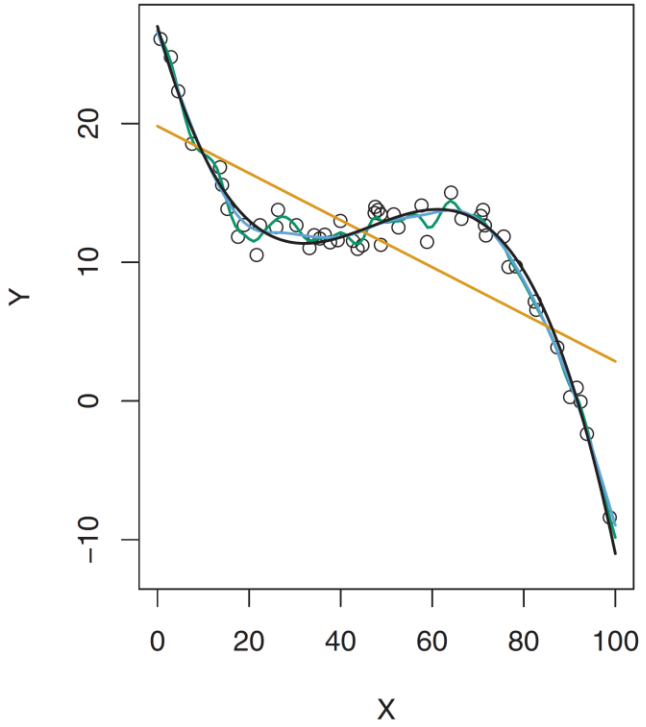
James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

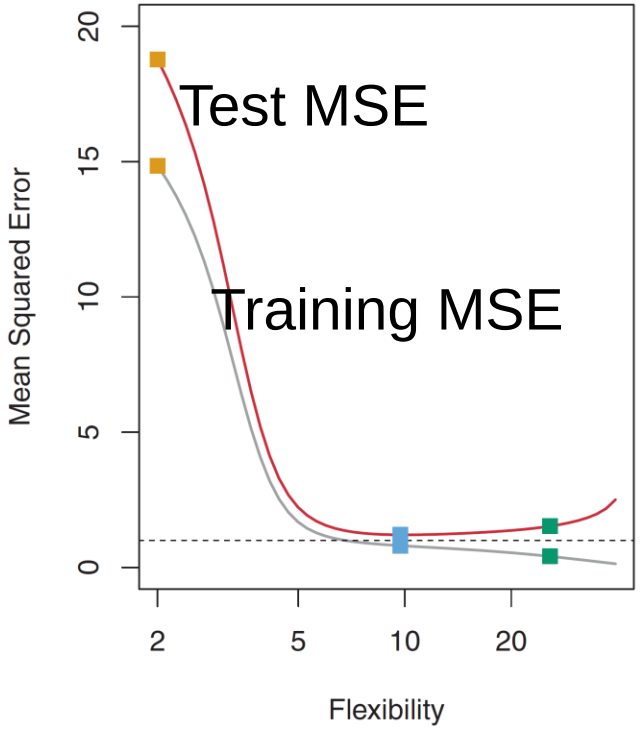
Model fit as a function of model complexity

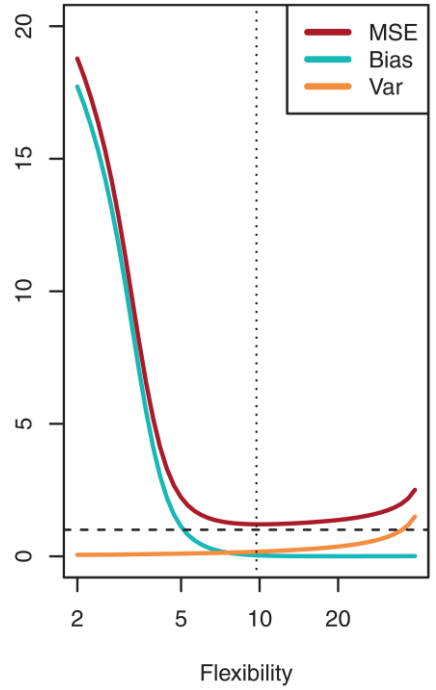
When true function (black) is highly non-linear, the test MSE of a linear fit (orange) may be driven almost completely by bias.

Test MSE never falls below irreducible error (horizontal dotted line).

Higher complexity models are clearly a better choice.







James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Issues that may cause bias or inflate variance

Non-linearity between the input and output variables

If the true relationship is very far from linear, then the prediction accuracy of the linear model can be substantially reduced due to inflated bias.

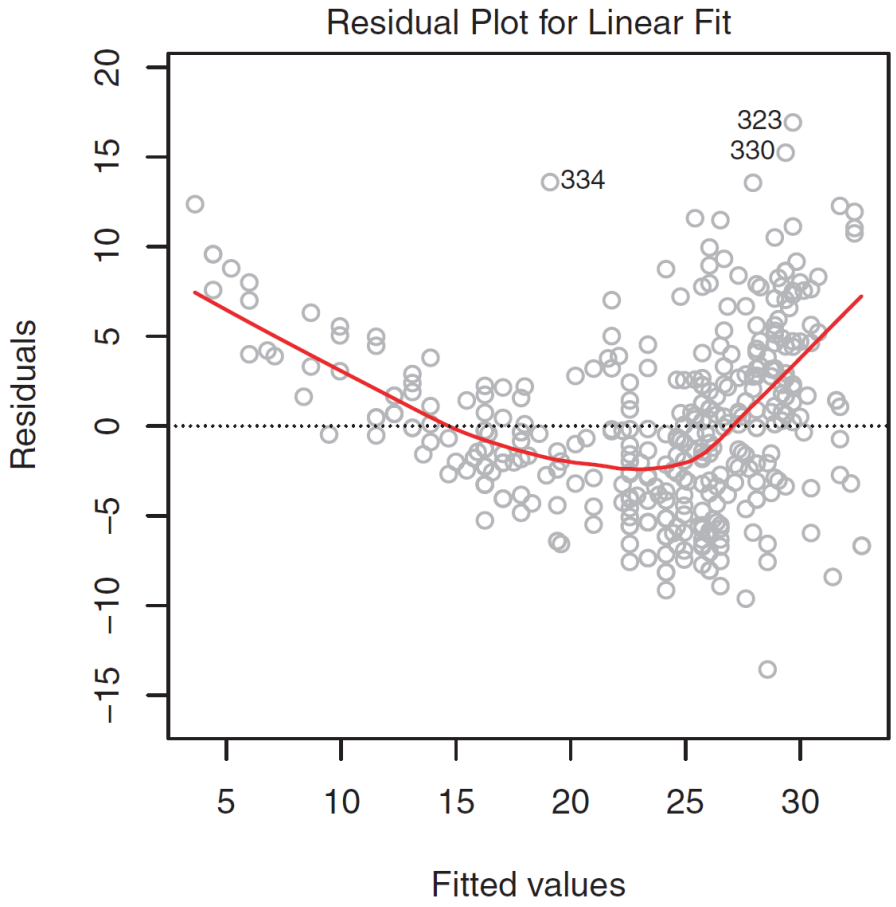
Residual plots can be useful for identifying non-linearity.

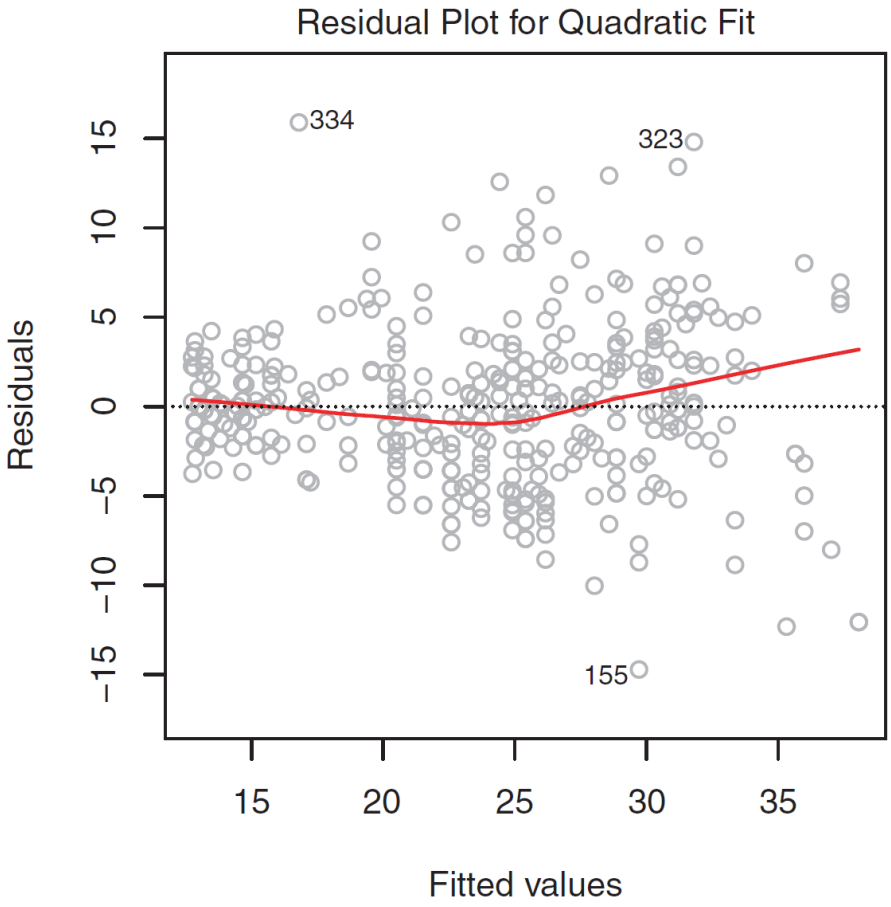
Residual plots display the residual e_i of each training observation i, $i = 1,2, \ldots, N$, as a function of the predicted value $\hat{y}_i$ for that training observation, where

$$e_i = y_i - \hat{y}_i$$

The hope is that there is no pattern in the residual plot.

Non-linearity identified by residual plot





If the residual plot indicates non-linearity, then a common approach is to perform a non-linear transformation of the features of the regression model (*e.g.*, $\log(X)$, $\sqrt{X}$, X^2).

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Issues that may cause bias or inflate variance

Correlation (non-independence) of errors

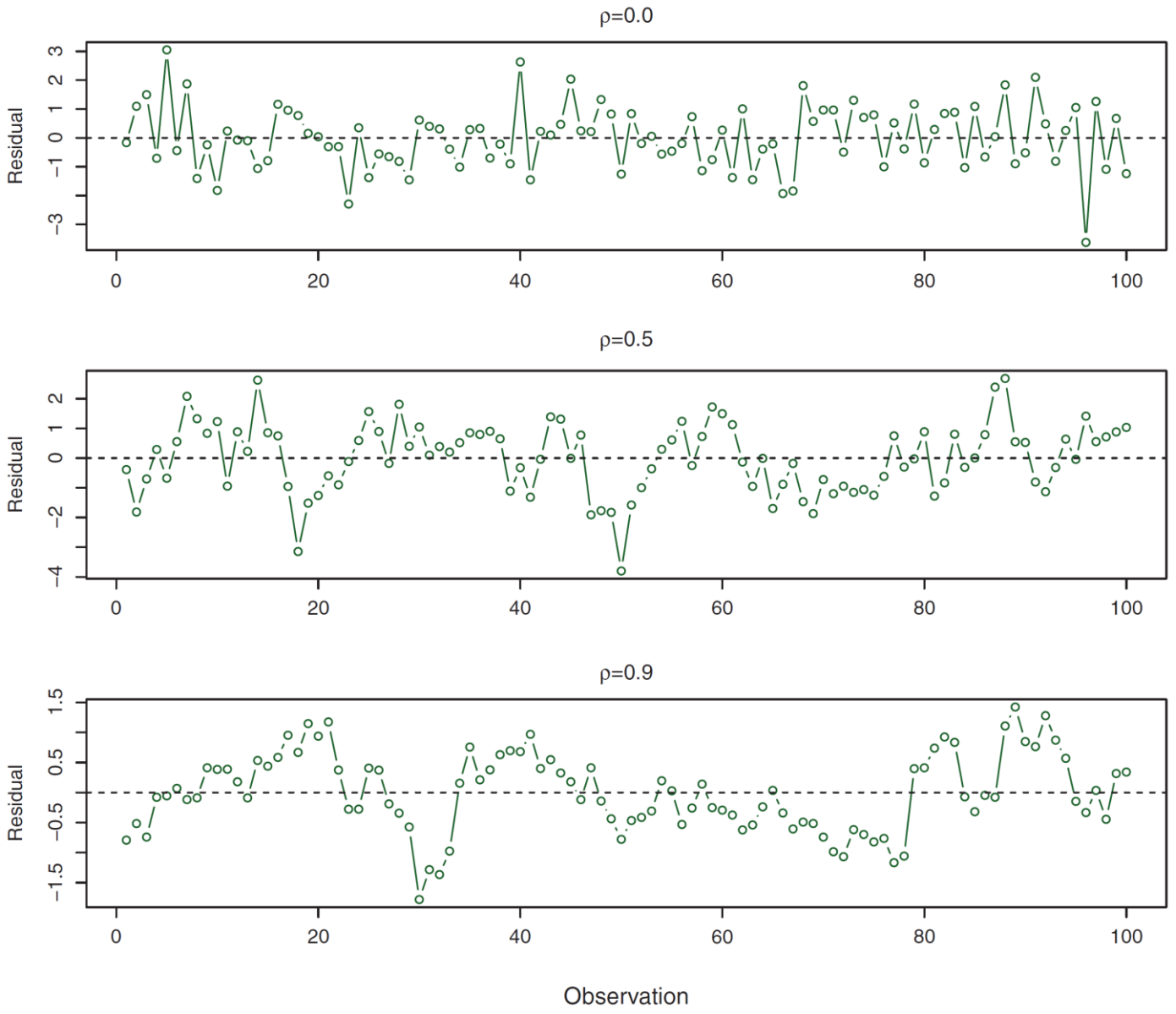
Under the Gauss-Markov theorem, the least squares estimate is the one with minimum variance for all unbiased linear estimators.

However, this assumes that the errors ϵ_i, $i = 1,2, \ldots, N$, are independent.

Inclusion of data with correlated errors may lead to fewer effective training examples, and therefore a potential higher variance of the linear model.

Such correlations are prevalent in time-series data, in which samples taken at neighboring time-points are correlated.

Non-independence of errors



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

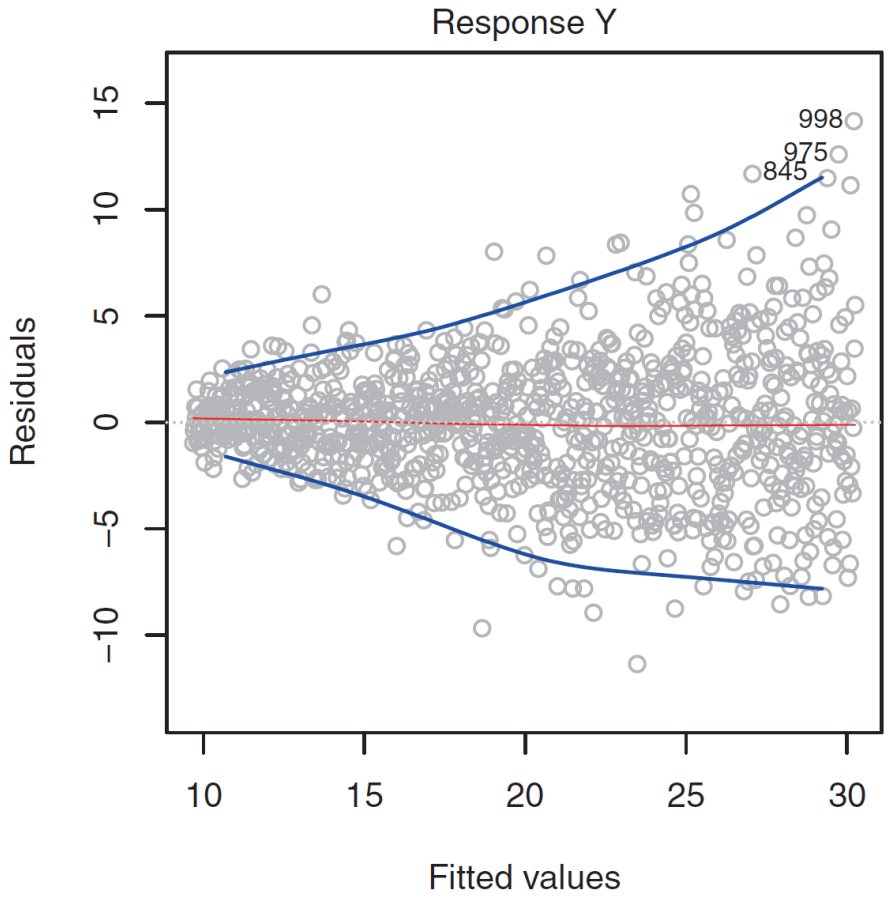
Issues that may cause bias or inflate variance

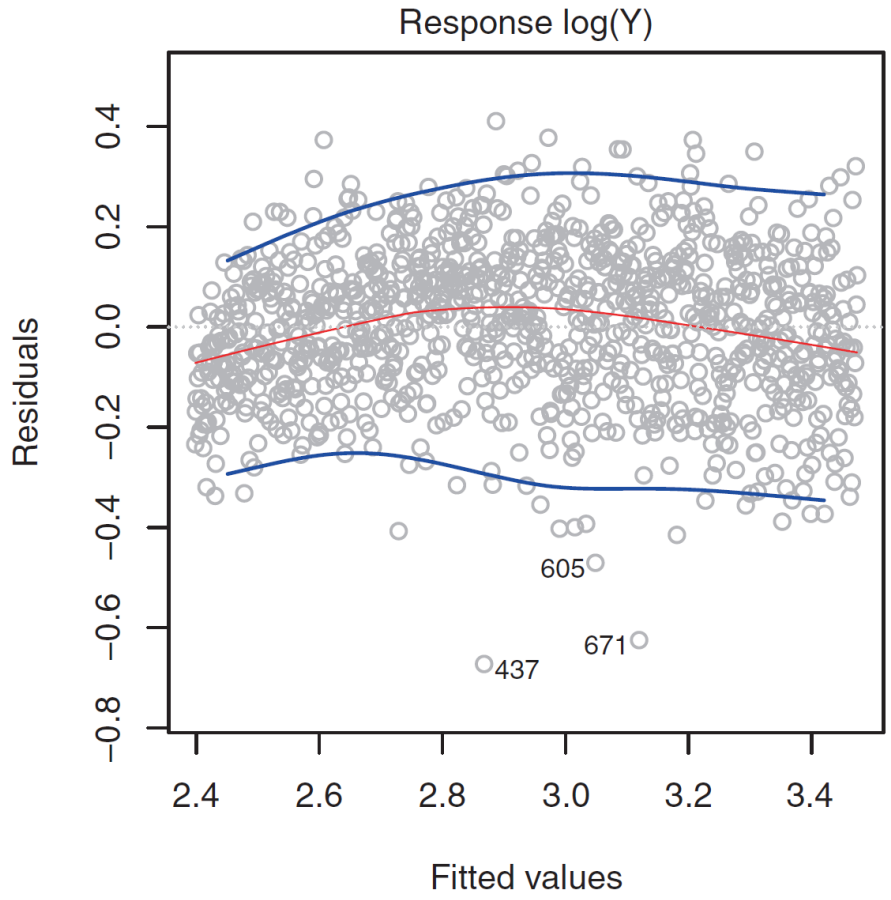
Non-constant variance of errors

Recall again that under the Gauss-Markov theorem, it was required that $\epsilon_i \sim N(0, \sigma^2)$, $i = 1,2, \ldots, N$. That is, error terms had the same variance.

It is possible to uncover non-constant variance of errors (heteroscedasticity) by observing a funnel shape within residual plots.

Non-constant variance of errors





Heteroscedasticity (left) can be potentially alleviated by transforming the output Y using a concave function (right), such as $\log(Y)$ or $\sqrt{Y}$.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Issues that may cause bias or inflate variance

Outliers

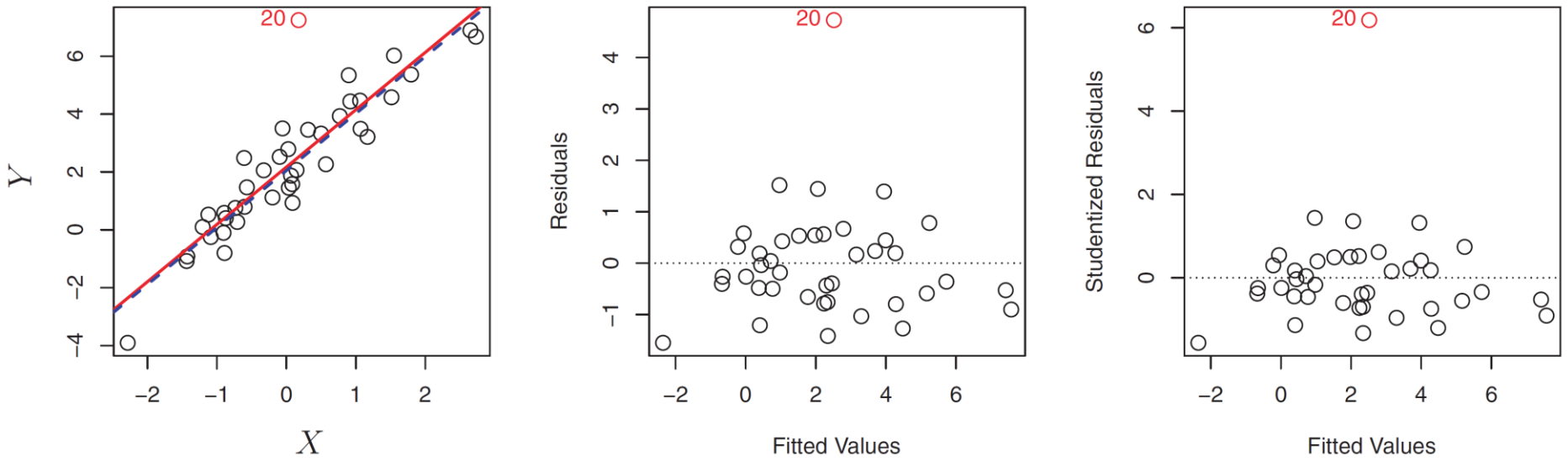
Points with output y_i that are distant from the predicted value $\hat{y}_i$. That is, large magnitude residuals $e_i = y_i - \hat{y}_i$.

To visualize outliers, make a studentized residual plot, in which the residuals are replaced with studentized residuals defined as

$$t_i = \frac{e_i}{\sqrt{\widehat{Var}(e_i)}}$$

Outliers are training datapoints in which $|t_i| > 3$.

Outliers



Training observation 20 (red) is a clear outlier based on the studentized residuals plot.

The least squares fit (red line) is robust to the outlier, as the regression line is similar once the outlier is removed (dotted line). However, outliers will reduce the RSS, thereby affecting measures of model fit.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Issues that may cause bias or inflate variance

High leverage observations

Points with unusual input x_i and measured by $\mathbf{H}_{ii}$.

Recall that $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$, and so

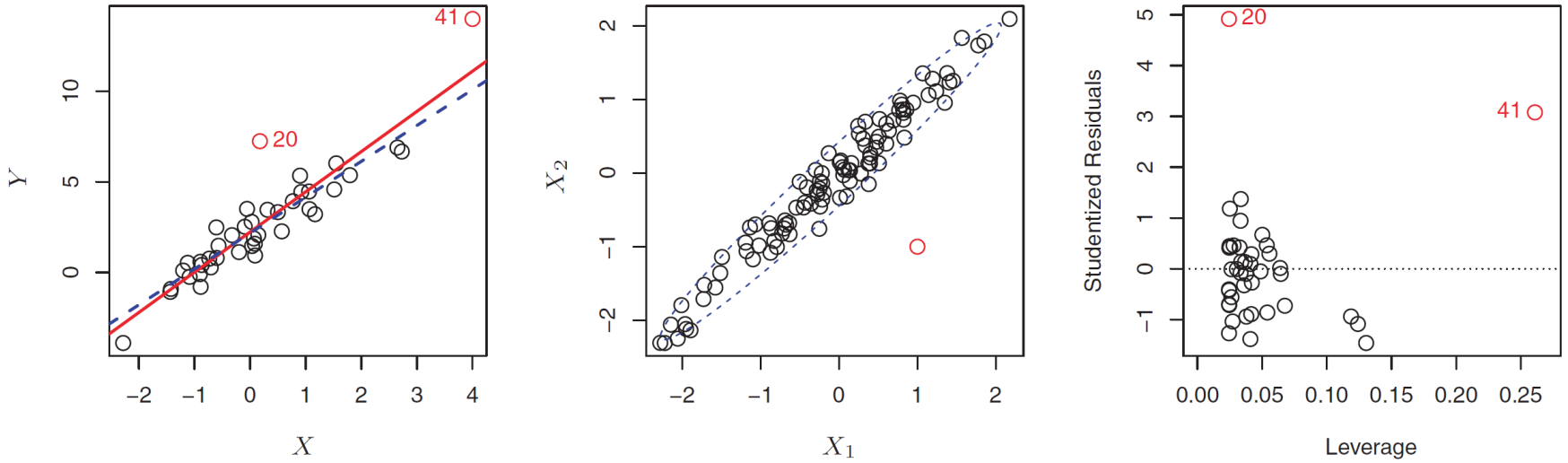
$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1N} \\ h_{21} & h_{22} & \cdots & h_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{NN} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

which gives

$$\hat{y}_i = \sum_{j=1}^{N} h_{ij} y_j = h_{ii} y_i + \sum_{j=1, j \neq i}^{N} h_{ij} y_j = \mathbf{H}_{ii} y_i + \sum_{j=1, j \neq i}^{N} \mathbf{H}_{ij} y_j$$

Therefore, $\mathbf{H}_{ii}$ relates the impact of y_i on $\hat{y}_i$.

High leverage observations



For high leverage observations, the model fit in such extreme regions of the parameter space relies heavily on one or a few datapoints

Outlier observations with high leverage can be particularly dangerous, as they can significantly alter the fitted models

With multidimensional input, high leverage observations need not take extreme values for any one dimension.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Correlated features can impact final model

Collinearity

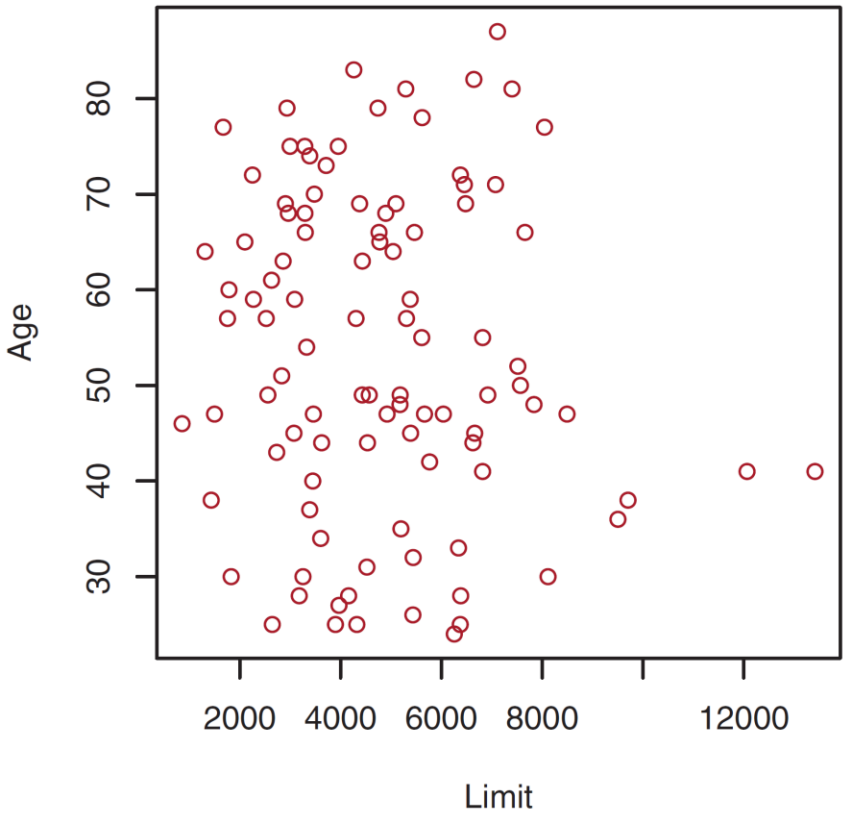
When two or more of the inputs (features) are correlated.

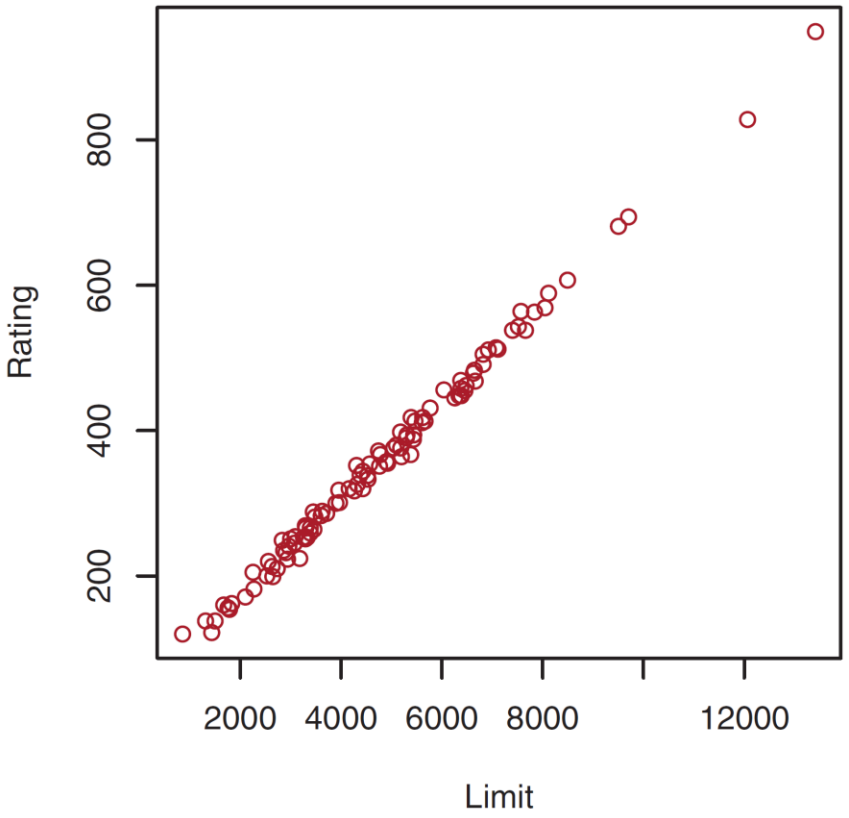
Can lead to parameter (coefficient) estimates to erratically change with slight alterations of the model or the training data.

An issue with using the normal equations with training data that has collinearity (or multicollinearity), is that the design matrix is not of full column rank, and therefore $\mathbf{X}^T\mathbf{X}$ cannot be inverted in the estimate

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Collinearity





Collinearity can be detected by computing the pairwise correlation matrix between features and identifying pairs with high correlation.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Non-parametric regression

Linear regression was our first example of a supervised learning method, and represented a parametric approach, in which we specify the type of function that relates output Y to input X.

However, as we have observed, if we specify the functional form of our parametric model such that it is too simple, then we may have high bias leading to poor prediction accuracy.

Similarly, if our specified functional form is too complex, then we may have high variance leading to poor prediction accuracy due to overfitting the training data.

We may therefore wish to not explicitly assume the relationship between Y and X.

K-nearest neighbors regression

We will consider the simplest of non-parametric regression methods, known as ***K*-nearest neighbors regression** (**KNN regression**).

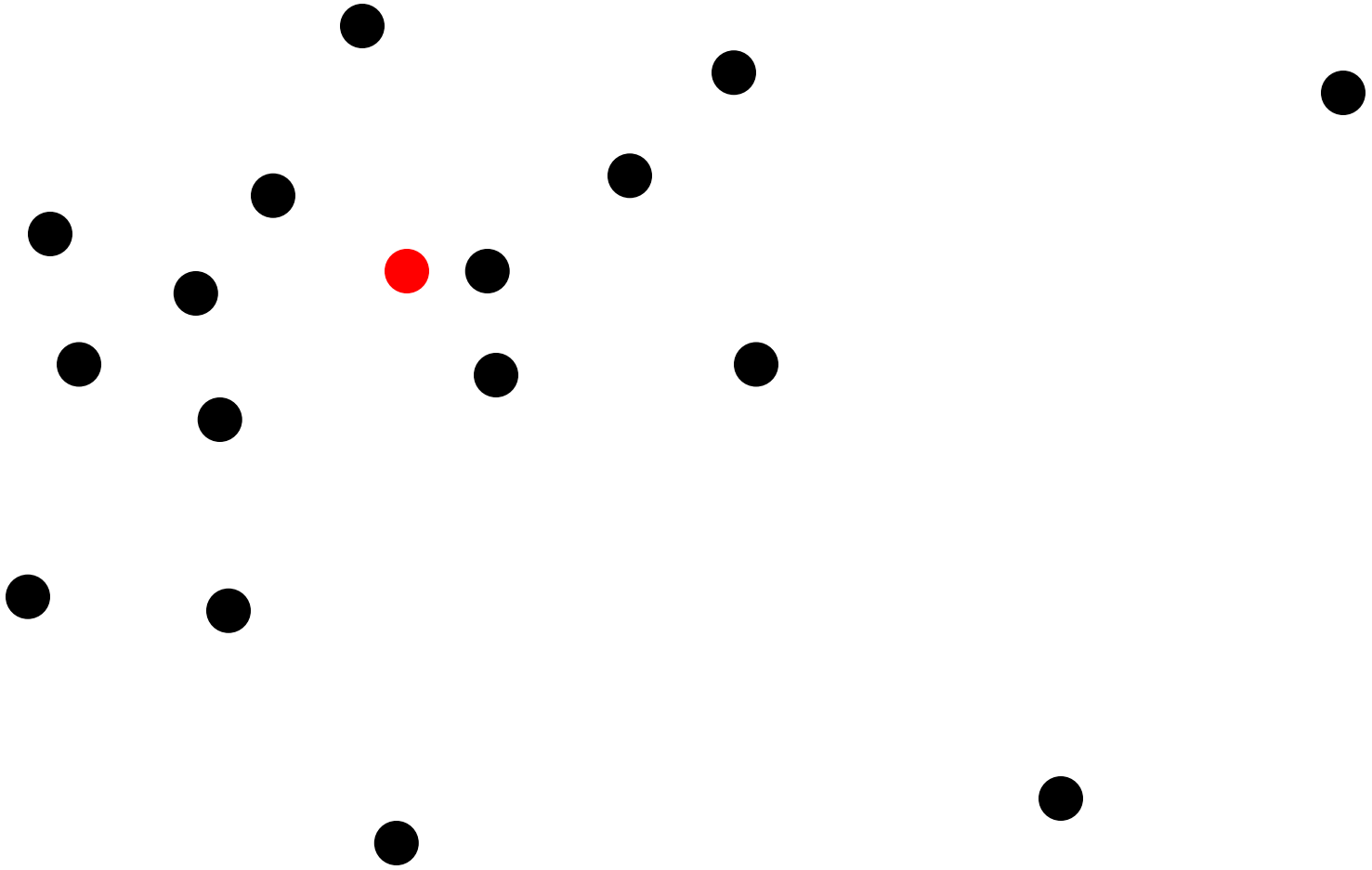
We start by choosing a positive integer K, $K = 1,2, ...$, which is a parameter of KNN regression, even though we say that KNN regression is non-parametric.

The parameter K represents the number of training observations that will be used to predict the output Y for the input X of a test observation.

How do we choose these K training observations used to make this prediction?

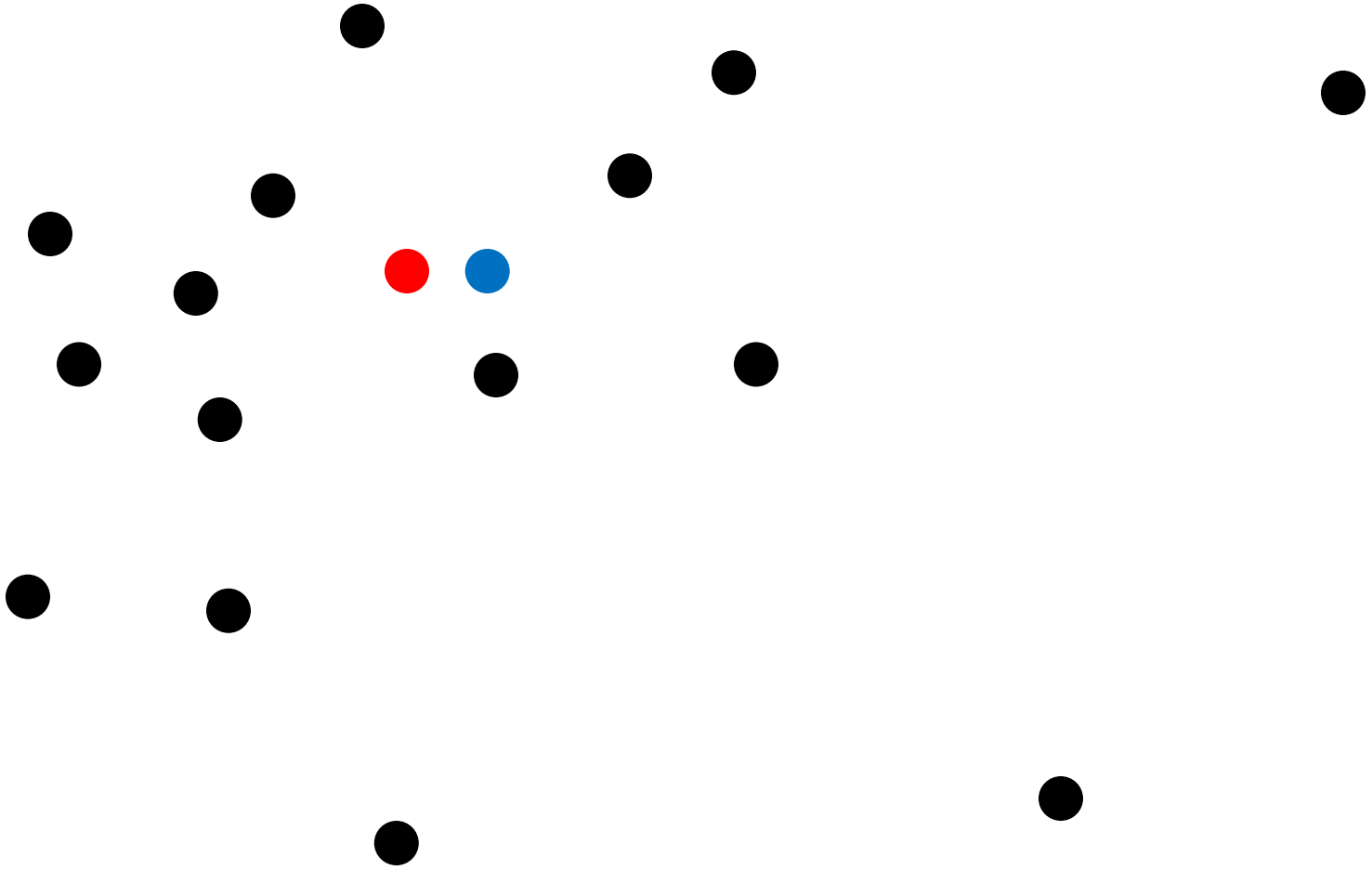
Neighborhood of the test observation

Let $\mathcal{N}(X)$ denote the set of K training observations (blue) that are the closest to test input X (red).



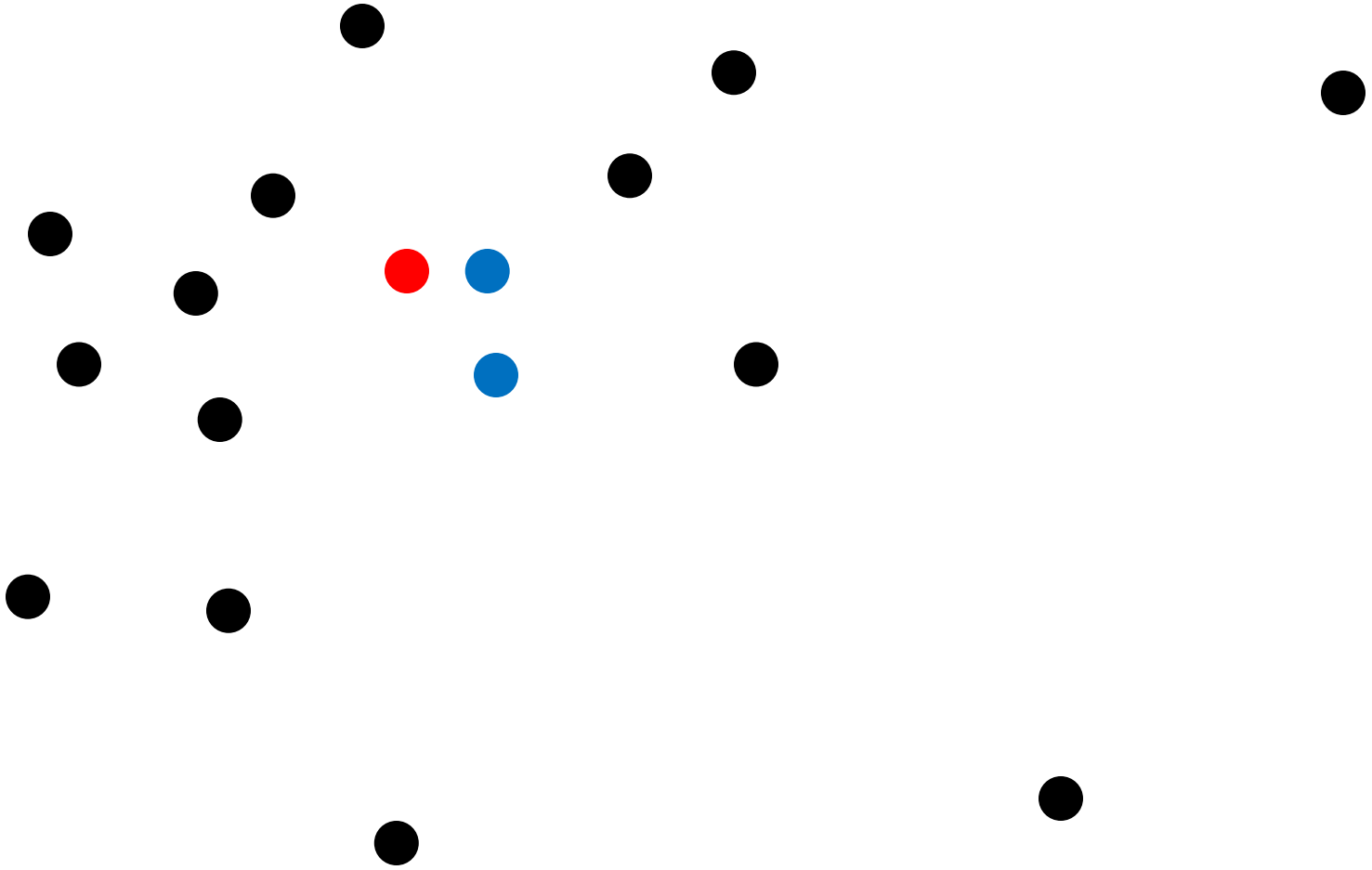
Neighborhood of the test observation ($K = 1$)

Let $\mathcal{N}(X)$ denote the set of K training observations (blue) that are the closest to test input X (red).



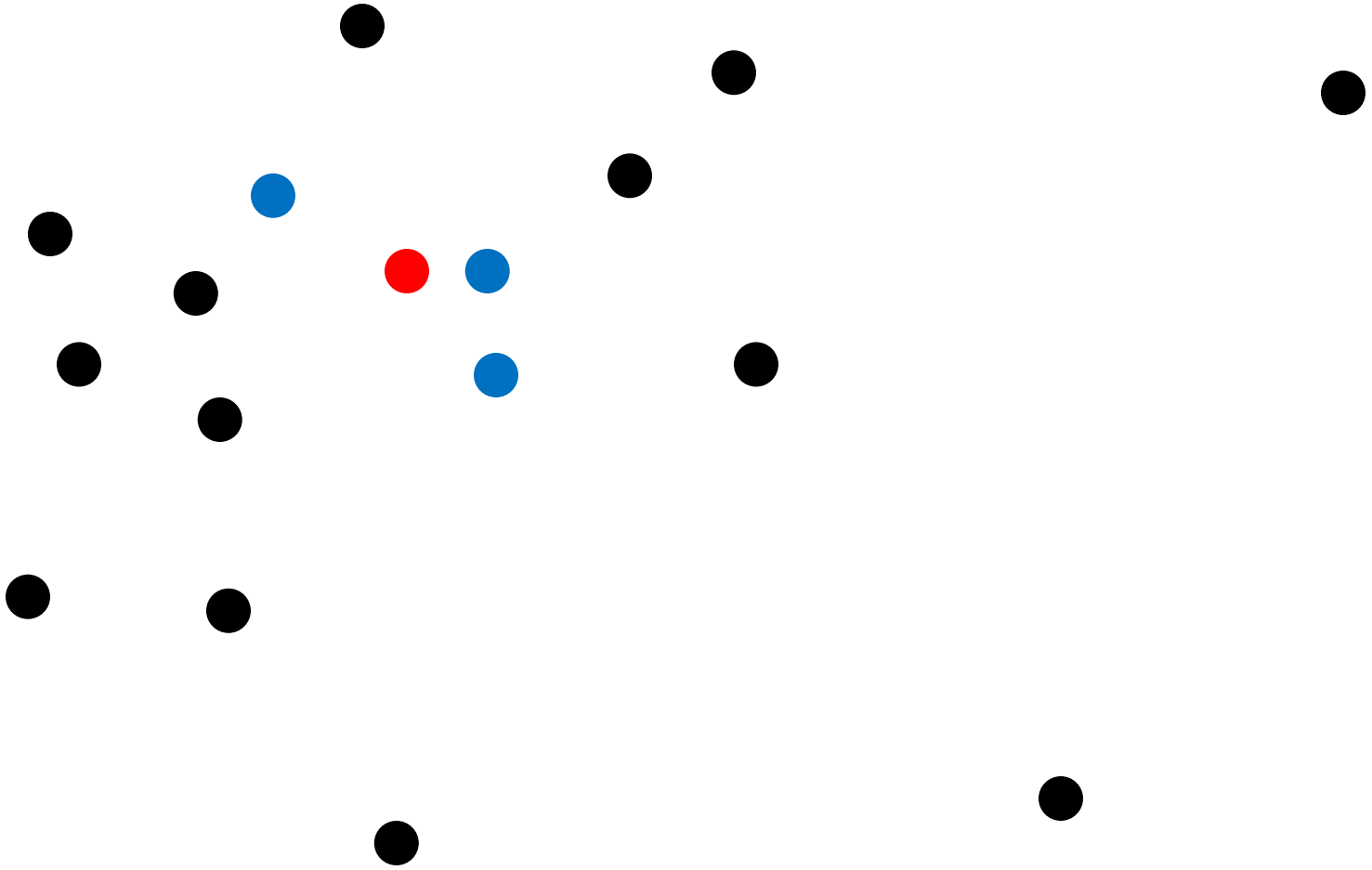
Neighborhood of the test observation ($K = 2$)

Let $\mathcal{N}(X)$ denote the set of K training observations (blue) that are the closest to test input X (red).



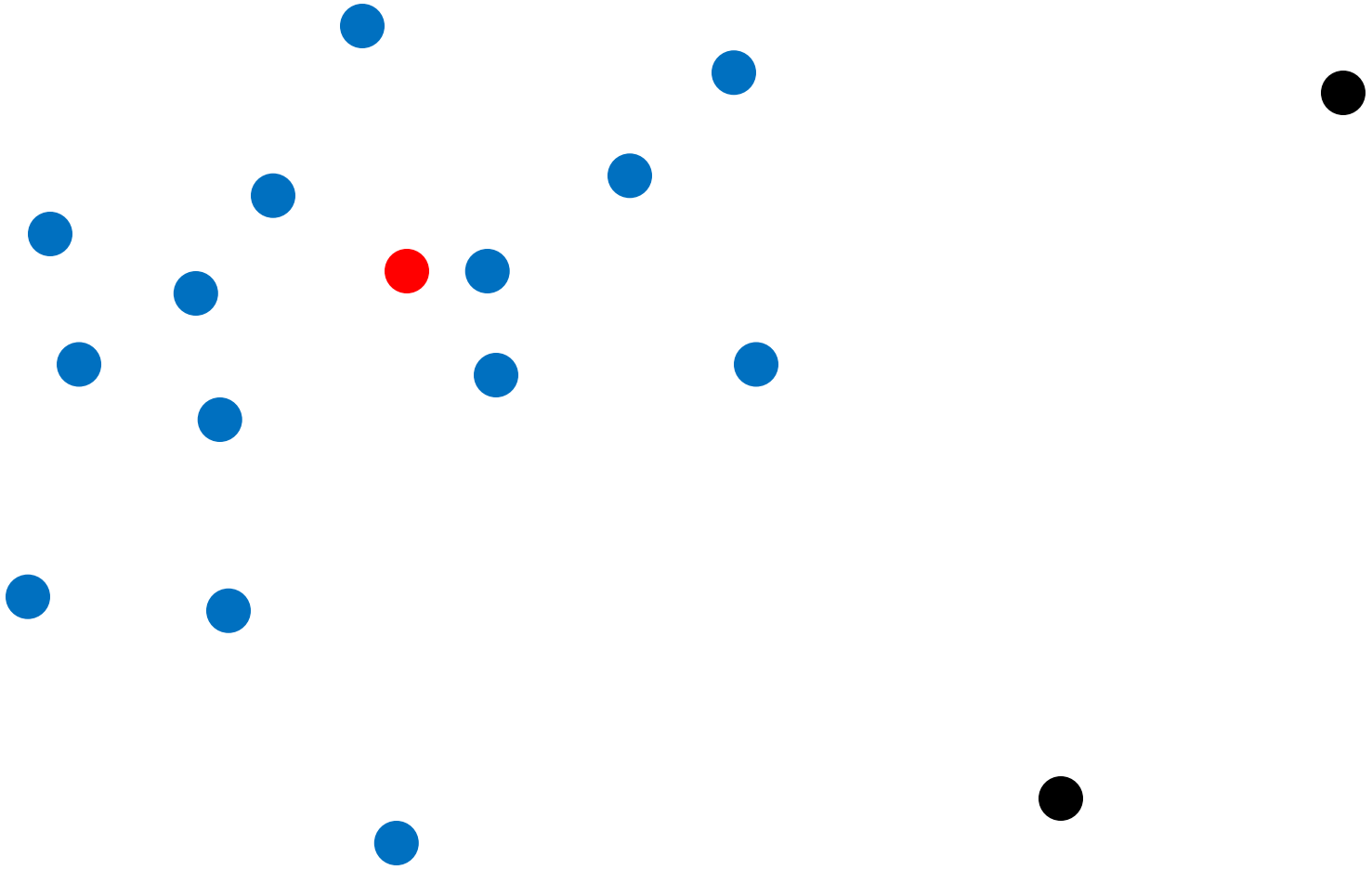
Neighborhood of the test observation ($K = 3$)

Let $\mathcal{N}(X)$ denote the set of K training observations (blue) that are the closest to test input X (red).



Neighborhood of the test observation ($K = 14$)

Let $\mathcal{N}(X)$ denote the set of K training observations (blue) that are the closest to test input X (red).



Predicting the output Y of test input X

Assuming parameter K, we will estimate $\hat{Y} = \hat{f}(X)$, where

$$\hat{f}(X) = \frac{1}{K} \sum_{x_i \in \mathcal{N}(X)} y_i$$

Therefore, we predict the output Y of test input X, by averaging the outputs of the K closest training observations.

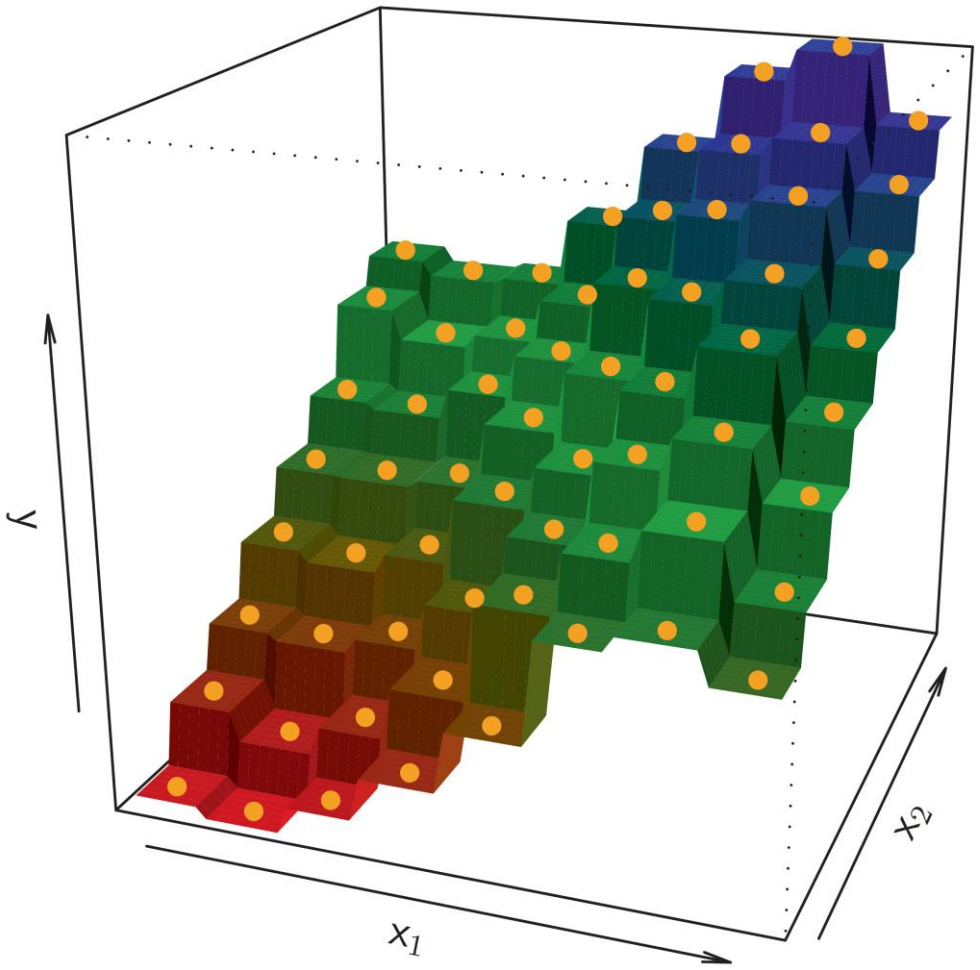
Small values of K lead to a small number of observations used to predict the output of the new observation

Large values of K will average over many training observations, some of which may be distant and not representative of the test observation.

Example prediction with $p = 2$ features ($K = 1$)

Training dataset contains $N = 64$ observations.

KNN prediction is rough step function passing through training data (orange).

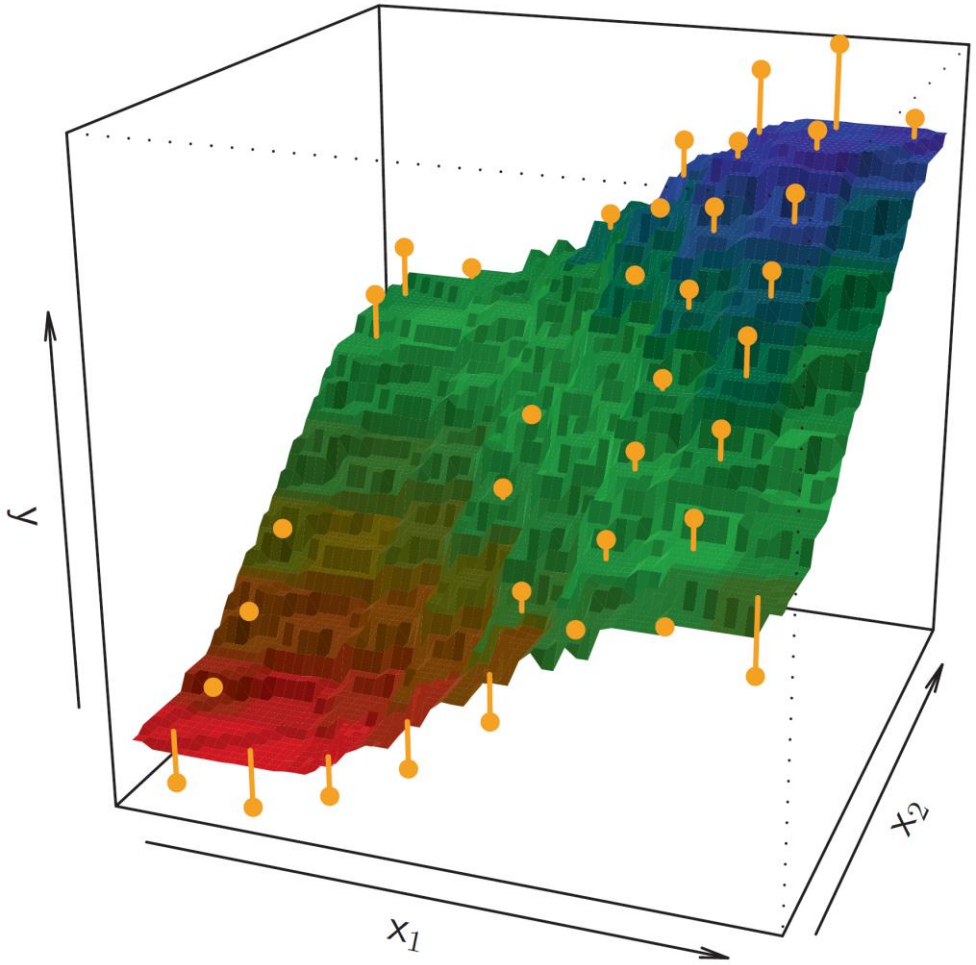


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Example prediction with $p = 2$ features ($K = 9$)

Training dataset contains $N = 64$ observations.

KNN prediction is smoother, and not passing through training data (orange).

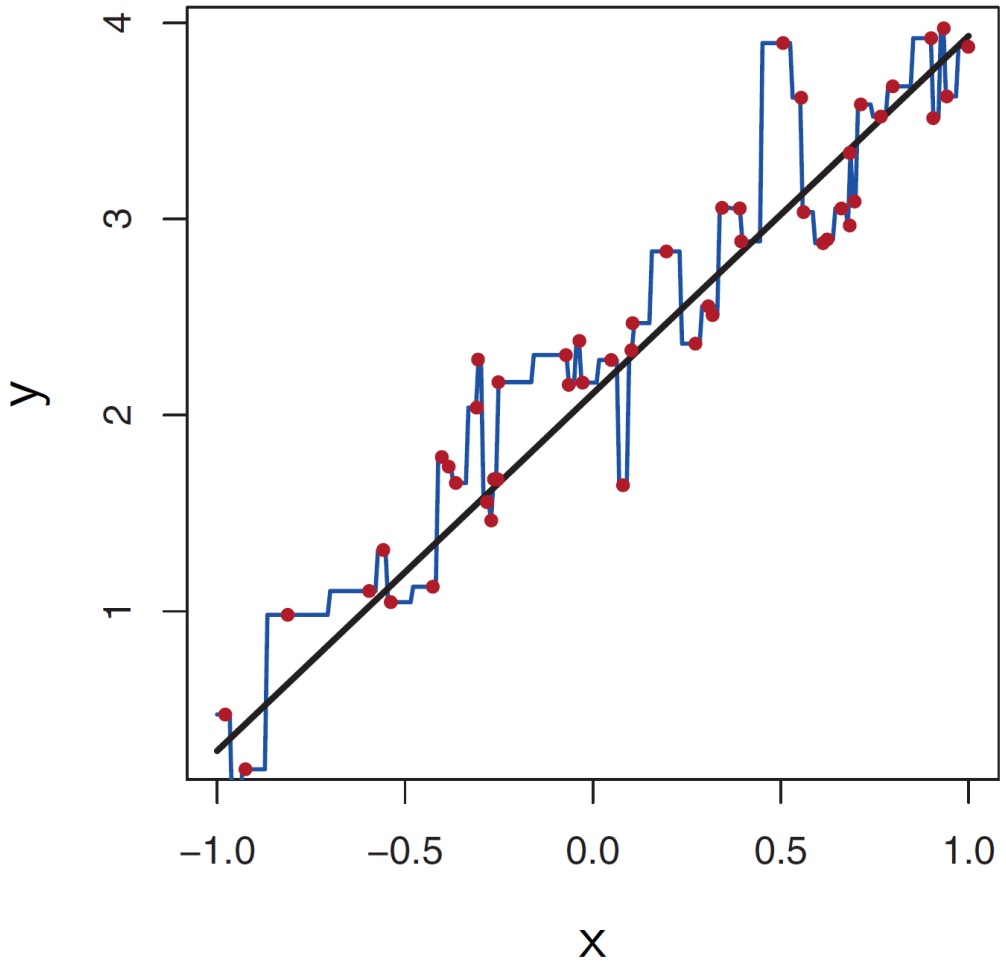


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Example prediction with $p = 1$ feature ($K = 1$)

True function is linear (black line).

KNN prediction (blue) directly models training data.

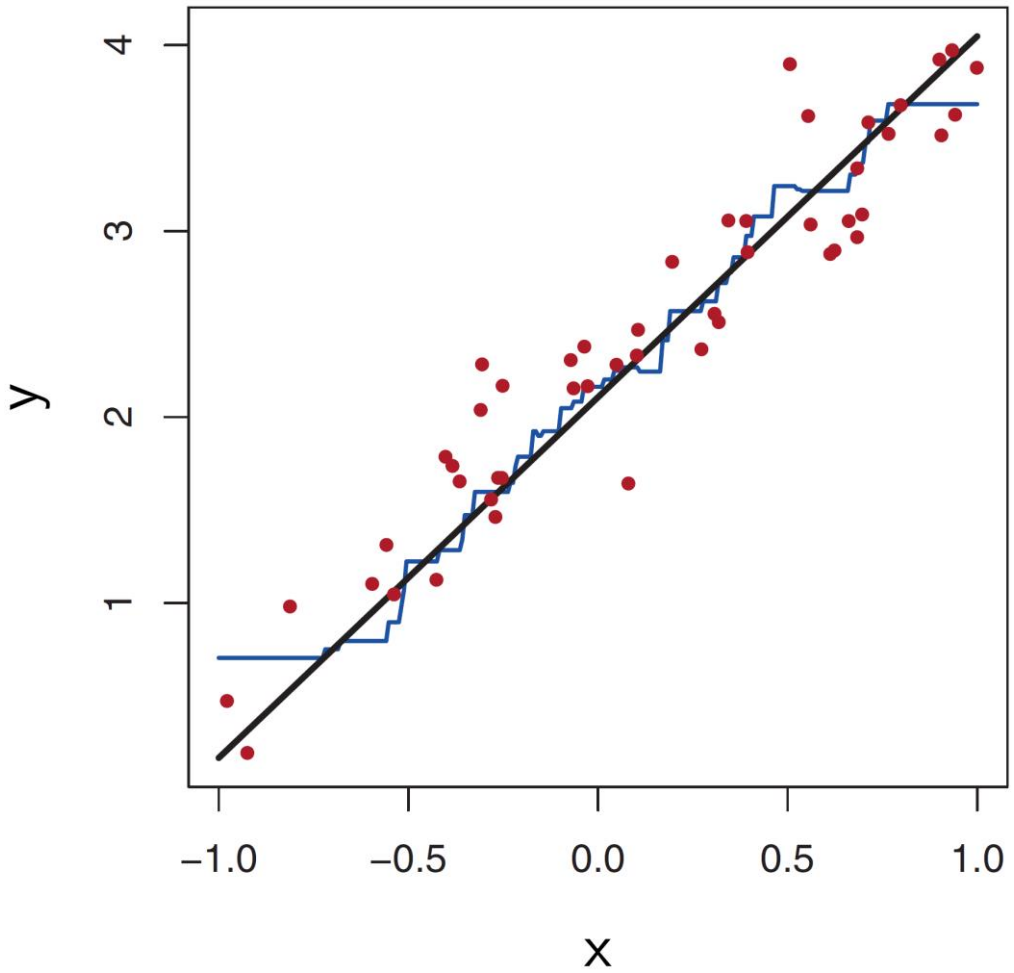


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Example prediction with $p = 1$ feature ($K = 9$)

True function is linear (black line).

KNN prediction (blue) behaves more linear.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

When is a parametric approach better?

A parametric approach, such as least squares linear regression will outperform a non-parametric approach like KNN if the true relationship between Y and X is close to the functional form defined by the parametric model.

That is, if the true relationship between Y and X is a polynomial of degree 1, then a linear model of the form

$$Y = \beta_0 + \beta_1 X$$

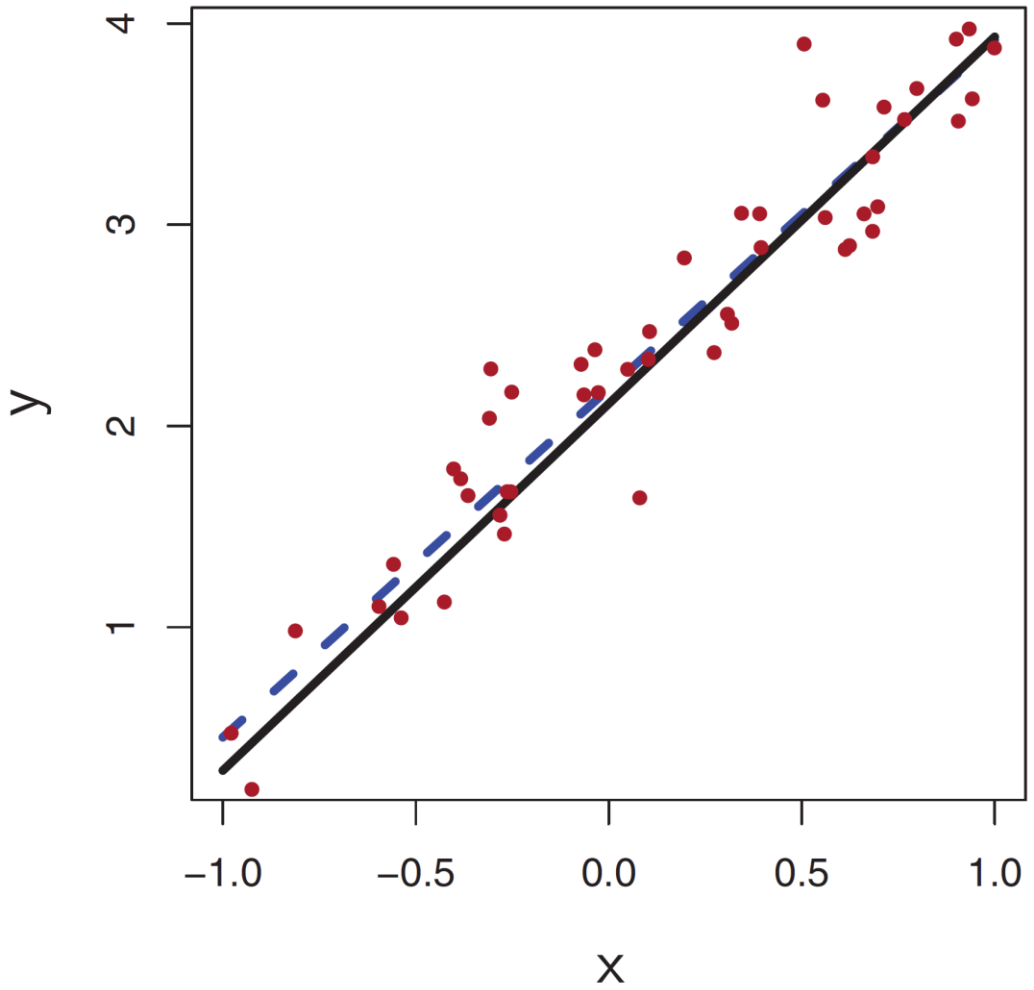
will outperform KNN.

However, if the form of the parametric model is mis-specified, then a non-parametric approach may perform better.

Linear regression outperforms KNN if truth is linear

True function is linear (black line).

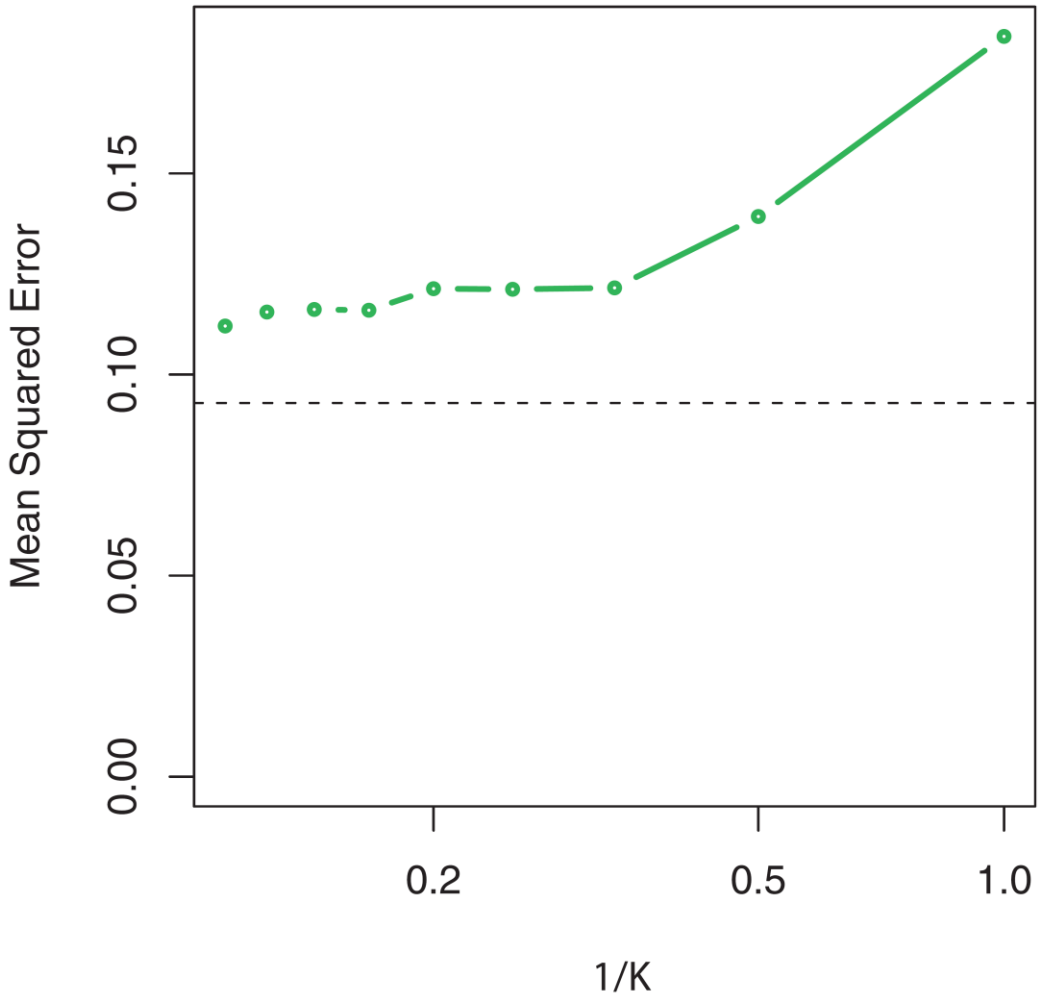
Linear regression (blue) provides good fit to function.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Linear regression outperforms KNN if truth is linear

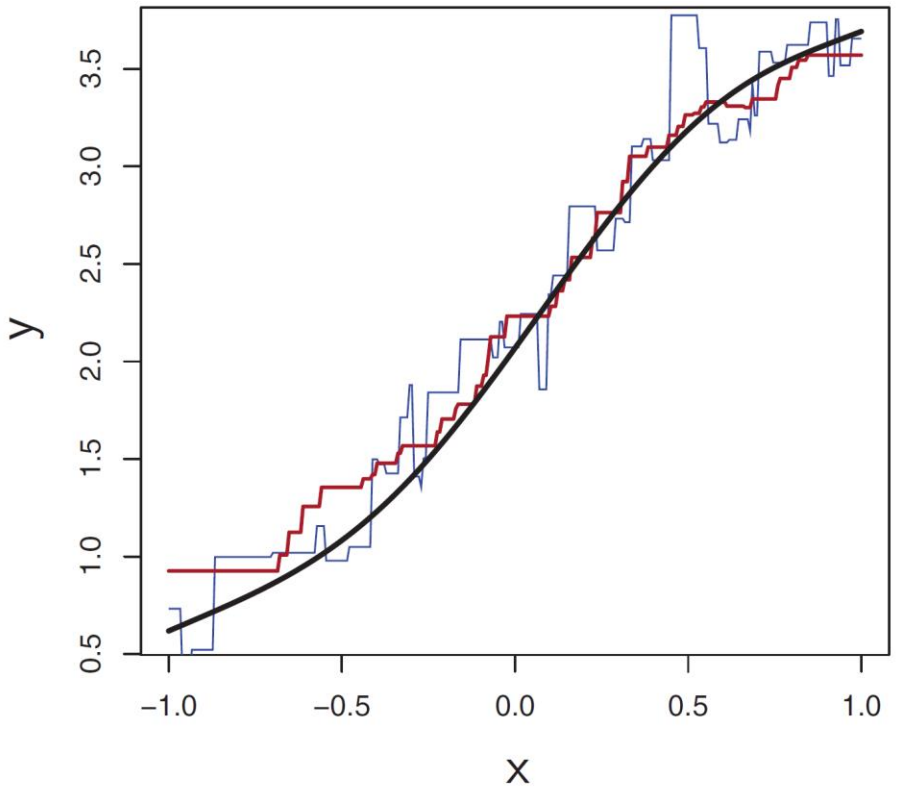
KNN regression test MSE (green) is always higher than linear regression test MSE (horizontal dashed line).

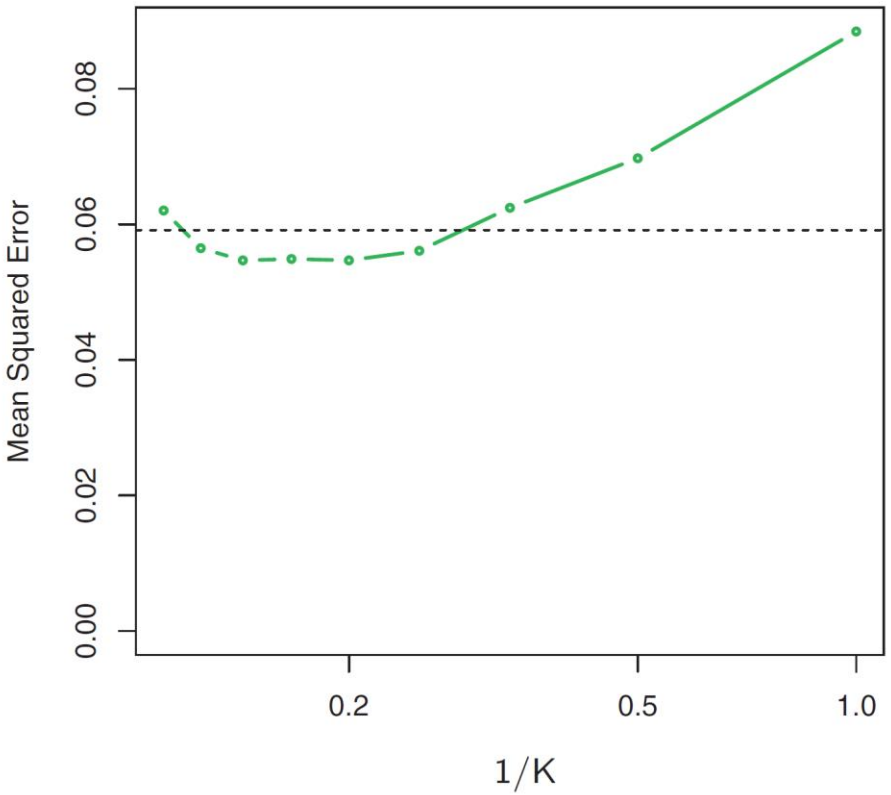


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

KNN can perform better when model is mis-specified

If the model is slightly non-linear, then KNN regression may outperform linear regression for intermediate values of K.

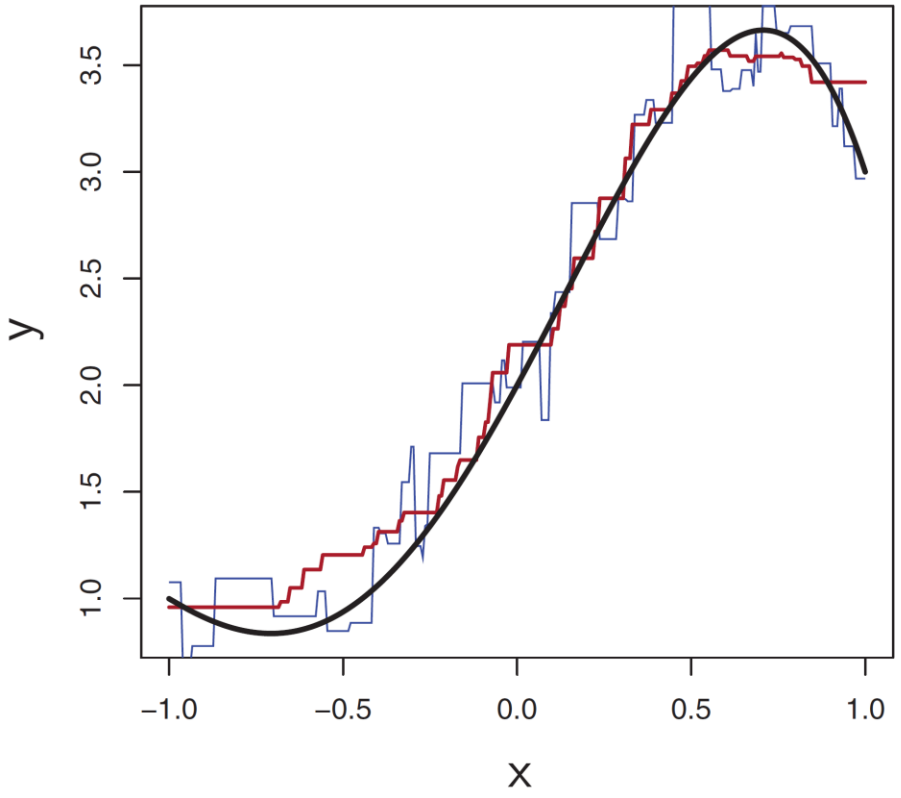


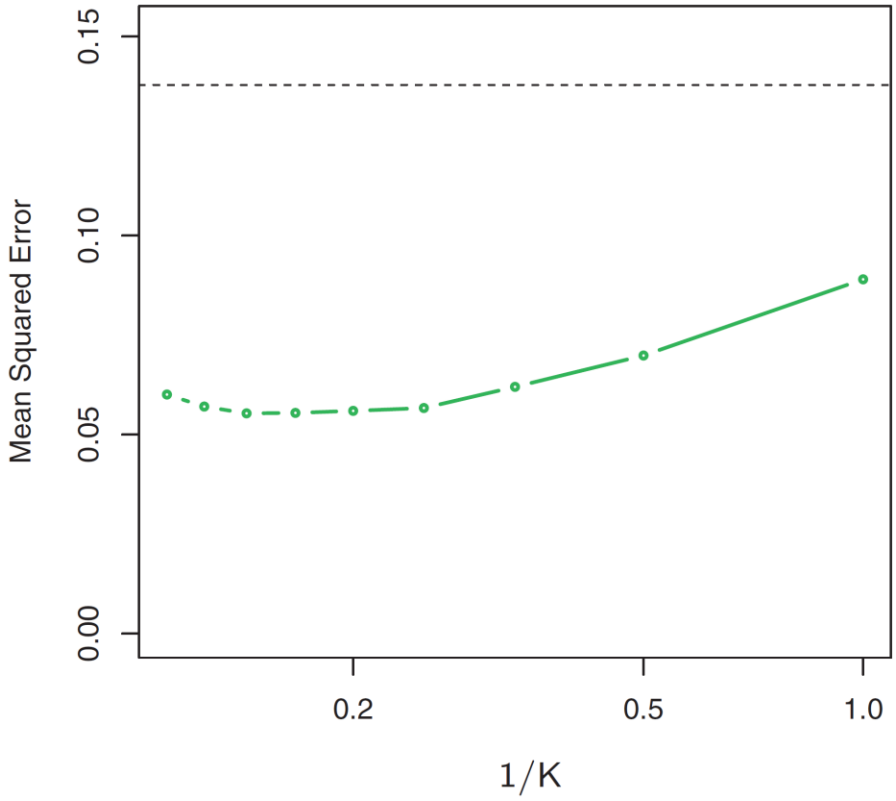


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

KNN can perform better when model is mis-specified

If the model is highly non-linear, then KNN regression may outperform linear regression for all or most values of K.





James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

The curse of dimensionality

A key assumption of KNN regression is that for a given test observation, there will be multiple (K) training observations nearby that are representative examples of the function nearby the test observation.

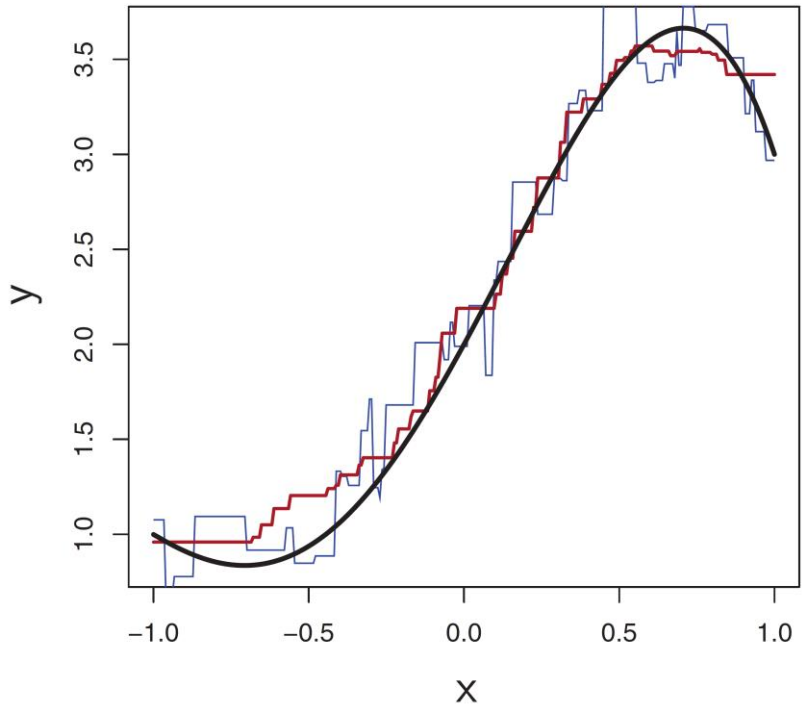
With many training observations, this is likely to be true with a small number of features p.

However, as the number of features p increases, the closest training examples to a given test observation may be very far away within the p-dimensional space, and therefore may not be good representatives.

Illustration of the curse of dimensionality

Consider a function of p features, where the relationship is nonlinear in the first dimension or feature and does not depend on the additional $p - 1$ features (*i.e.*, these features are additional noise).

The first dimension has non-linear relationship as below.



The training data is based on $N = 100$ observations.

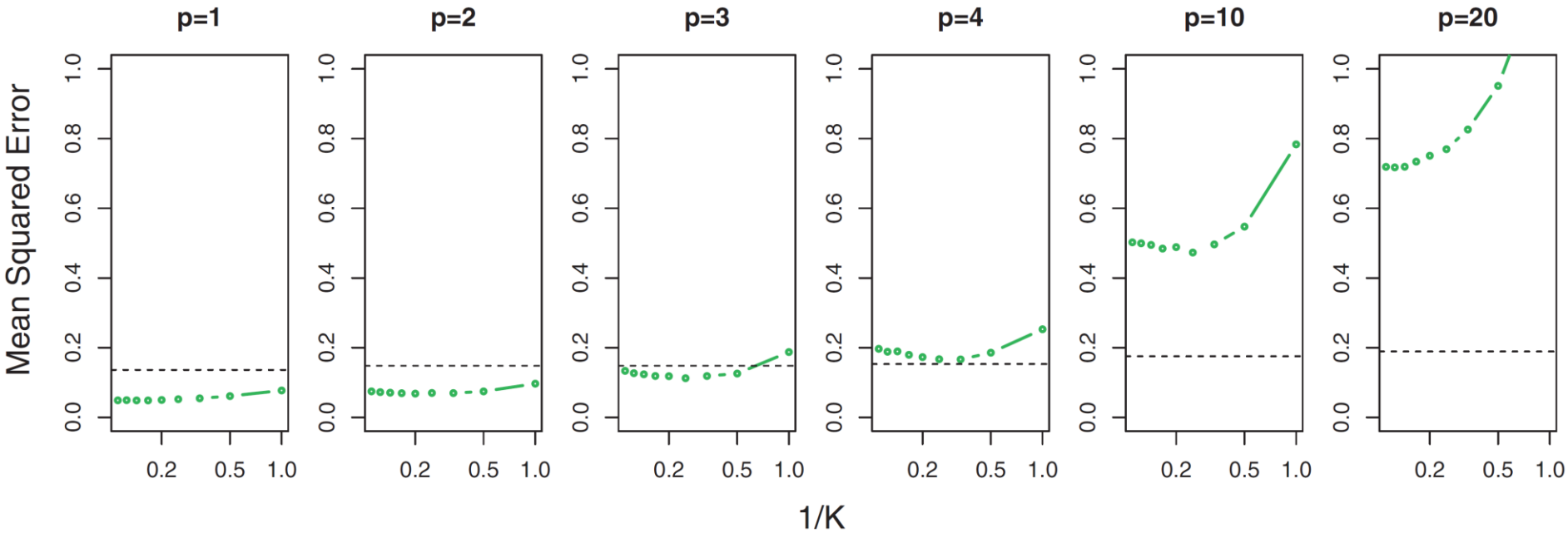
James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Illustration of the curse of dimensionality

For small values of p, test MSE of KNN (green) is lower than linear regression (horizontal dashed line).

For moderate values of p, test MSE is highly inflated and performs far worse than linear regression for all values.

In contrast, as p increases, test MSE for linear regression only slightly increases.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

How do we train a model to have low test error?

We have seen that the best model is not necessarily the simplest or most complex.

Moreover, though training error is used to measure the fit of a model and to estimate its parameters, this training error decreases with increasing model complexity.

However, we are interested in how well a trained model can predict unobserved datapoints not originally in the training set.

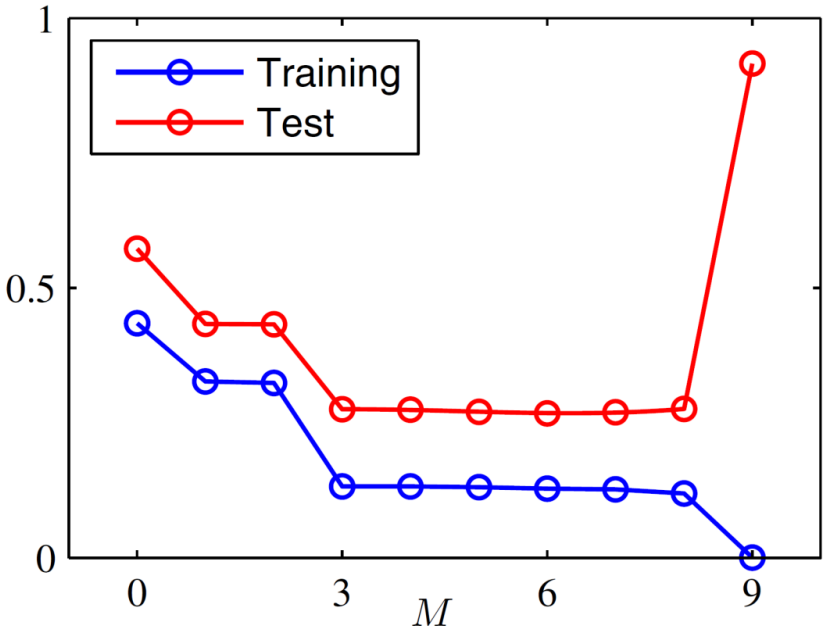
In general, the test error will display a U-shaped curve, with minimum test error at some intermediate level of complexity.

Illustration of training vs. test error

Recall that we can model an output Y by a polynomial function of input X of degree M, $M = 0,1, ...,$ as

$$Y = \sum_{m=0}^{M} \beta_m X^m$$

Prediction error



How do we identify M such that the inferred model has good test error?

Bishop (2006) *Pattern Recognition and Machine Learning*. Springer.

Goals when fitting models

We have two separate goals: model selection and model assessment.

Model selection: estimate the performance of different models to identify the best one.

Model assessment: given a final model choice, estimate its prediction error on a new dataset.

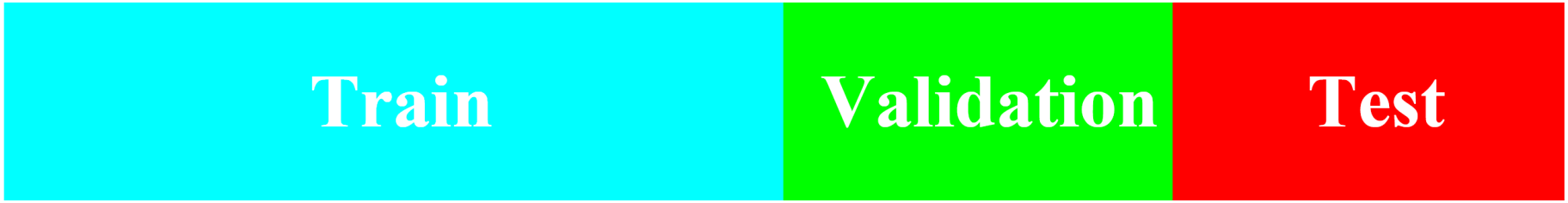
If we have a lot of data, then the optimal approach for both of these is to divide the dataset uniformly at random into three parts:

Training set

Validation set

Test set

Training, validation, and test sets



Training set: used for fitting the models.

Validation set: used to estimate prediction error for model selection.

Test set: used for assessing the prediction (test) error on new data.

The test set should only be used at the end of the analysis for model assessment, and not for choosing the best model.

There is not general rule for how to split the dataset, but typical split may look like 50%, 25%, and 25% for training, validation, and test.

Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Do we need to split into three parts?

For many datasets it may not be possible to split the data into three parts.

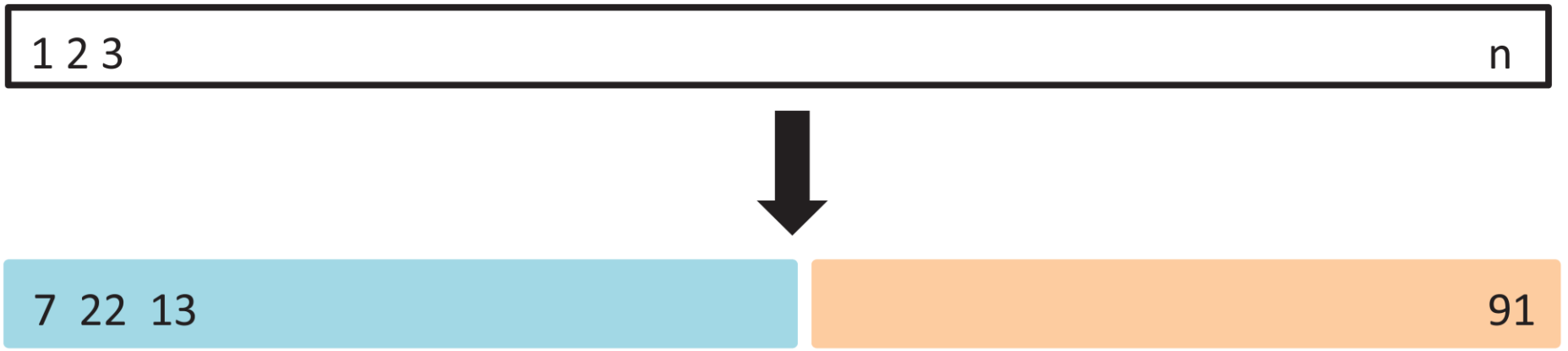
That is, there may not be enough data to create a separate test dataset, without lowering the sample size such that it is deleterious to model selection.

However, model selection is highly important, and often our goal will be to train a machine learning that should perform as well as possible on unseen data.

We will therefore be considered with methods that split the dataset into only training and validation sets.

The validation (hold-out) set approach

The validation (or hold-out) set approach involves dividing the training dataset uniformly at random into roughly two equally-sized parts: the training set and the validation set (known as the hold-out set).



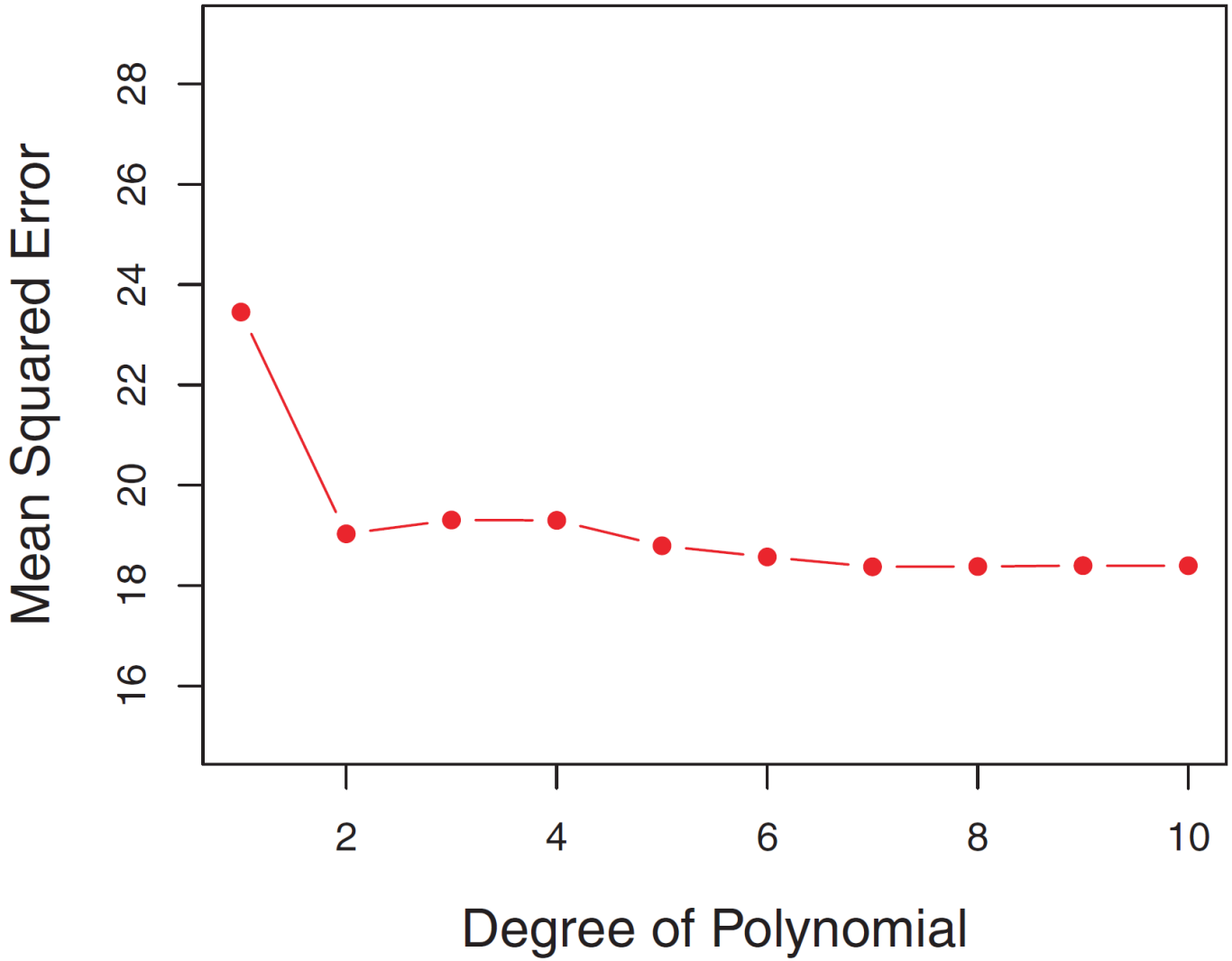
The model is trained on the training set (blue), and the test error of the trained model is estimated from the validation set (orange).

Models of different complexity can be trained and their test error evaluated, and the best-performing model can be selected.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Validation set approach on polynomial regression

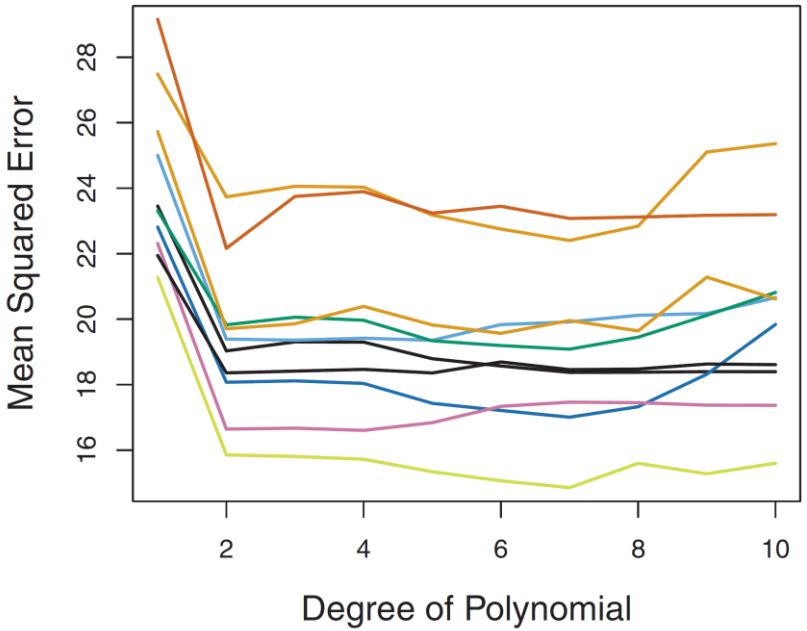
Validation (test) error for a single split into training and validation sets.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Validation set approach on polynomial regression

Validation (test) error for a different splits.



Test error is highly variable, and depends on which observations are included in the training vs. validation sets.

Validation set approach trained the model on a much smaller amount of data than the entire dataset.

Methods tend to perform worse when trained on fewer observations, suggesting validation error overestimates test error on entire dataset.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Overcoming pitfalls of the validation set approach

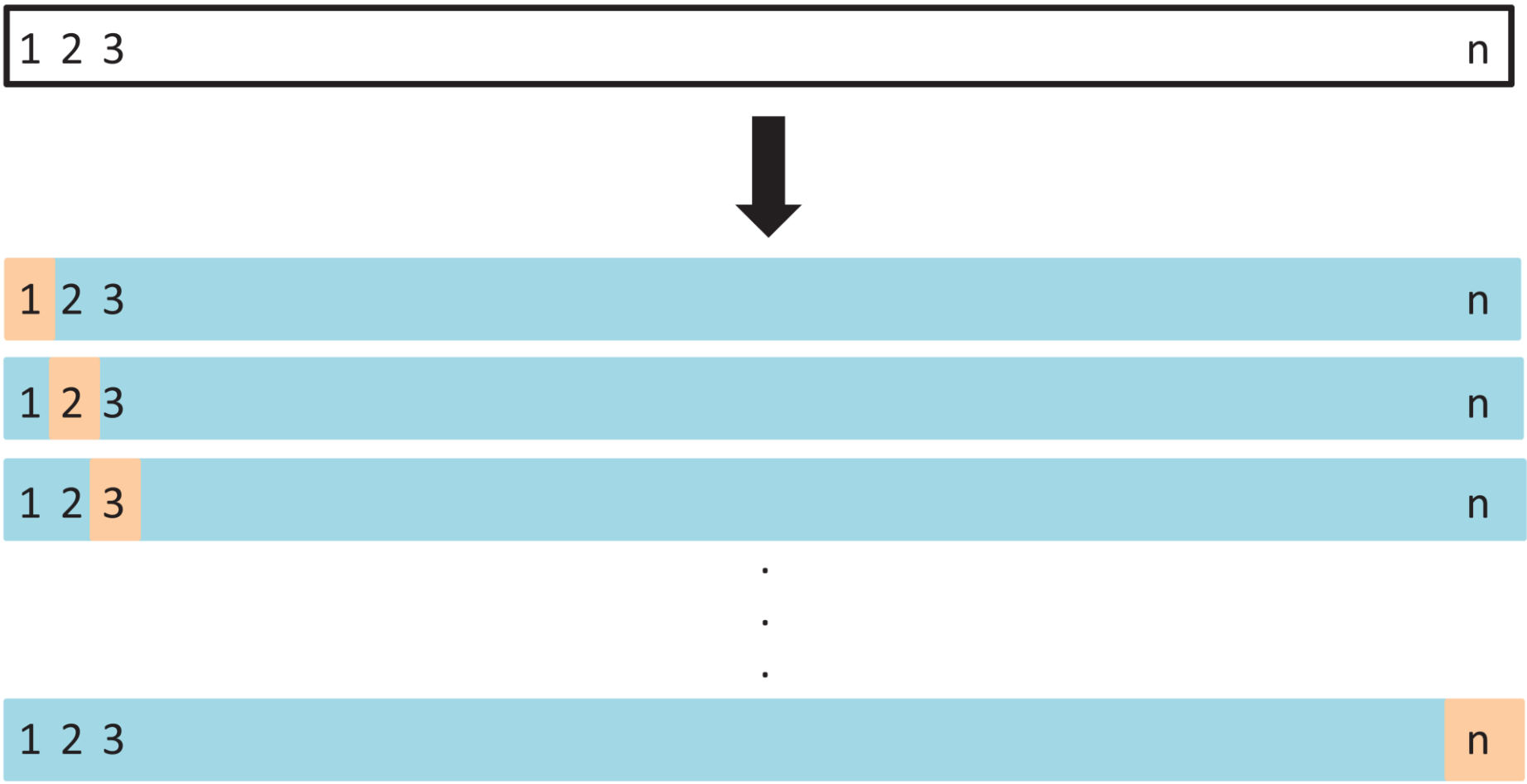
We need the learned model to be more robust to variability in the choice of training and validation set.

We also want the trained model to still be based on a large sample size.

We consider an approach termed **cross-validation** (**CV**), and specifically investigate two forms:

- Leave-one-out cross-validation (LOOCV)
  - Also termed N-fold cross validation
- k-fold cross validation

Schematic of LOOCV



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Leave-one-out cross-validation (LOOCV)

LOOCV also involves dividing the training dataset into two sets, one as training, and the other as validation.

However, only a single training observation is removed at a time (leaving one out).

This observation forms the validation set, and the remainder of the $N - 1$ observations form the training set.

Leave-one-out cross-validation (LOOCV)

We start by removing training observation (x_1, y_1), and using observations $\{(x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)\}$ for training.

The model is fit to the training data and a prediction is made for training observation (x_1, y_1).

This validation (test) error of training observation (x_1, y_1) is

$$\mathrm{MSE}_1 = (y_1 - \hat{y}_1)^2$$

This estimate of the MSE of the method will be highly variable because it is based on this single observation.

Leave-one-out cross-validation (LOOCV)

We repeat this process for each of the N training observations, such that we remove training observation (x_i, y_i), and using observations $\{(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \ldots, (x_N, y_N)\}$ for training.

The model is fit to the training data and a prediction is made for training observation (x_i, y_i).

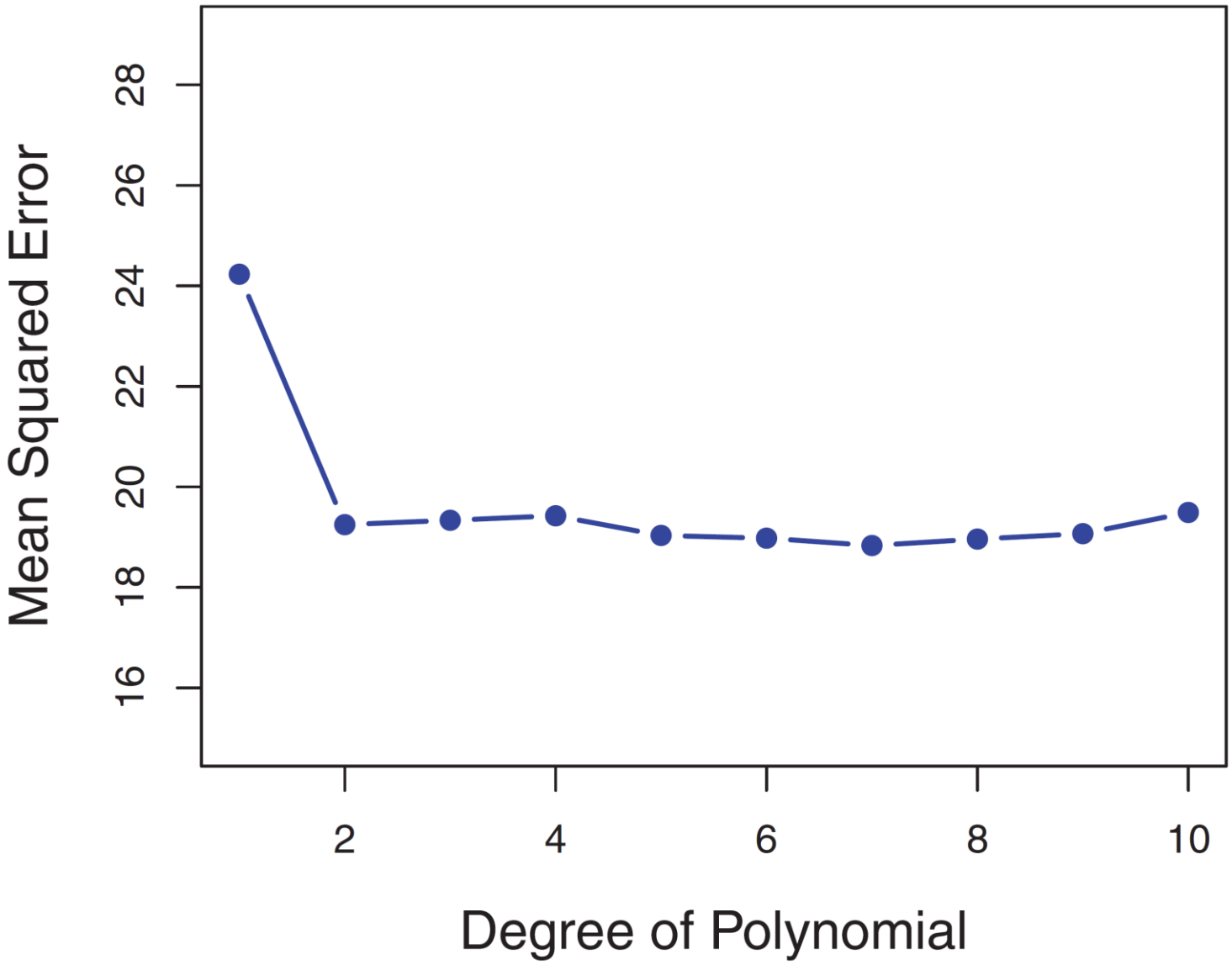
This validation (test) error of training observation (x_i, y_i) is

$$\mathrm{MSE}_i = (y_i - \hat{y}_i)^2$$

and the overall LOOCV (N-fold CV) validation (test) error is

$$\mathrm{CV}_{(N)} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{MSE}_i$$

LOOCV approach on polynomial regression



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Advantages of LOOCV over validation set

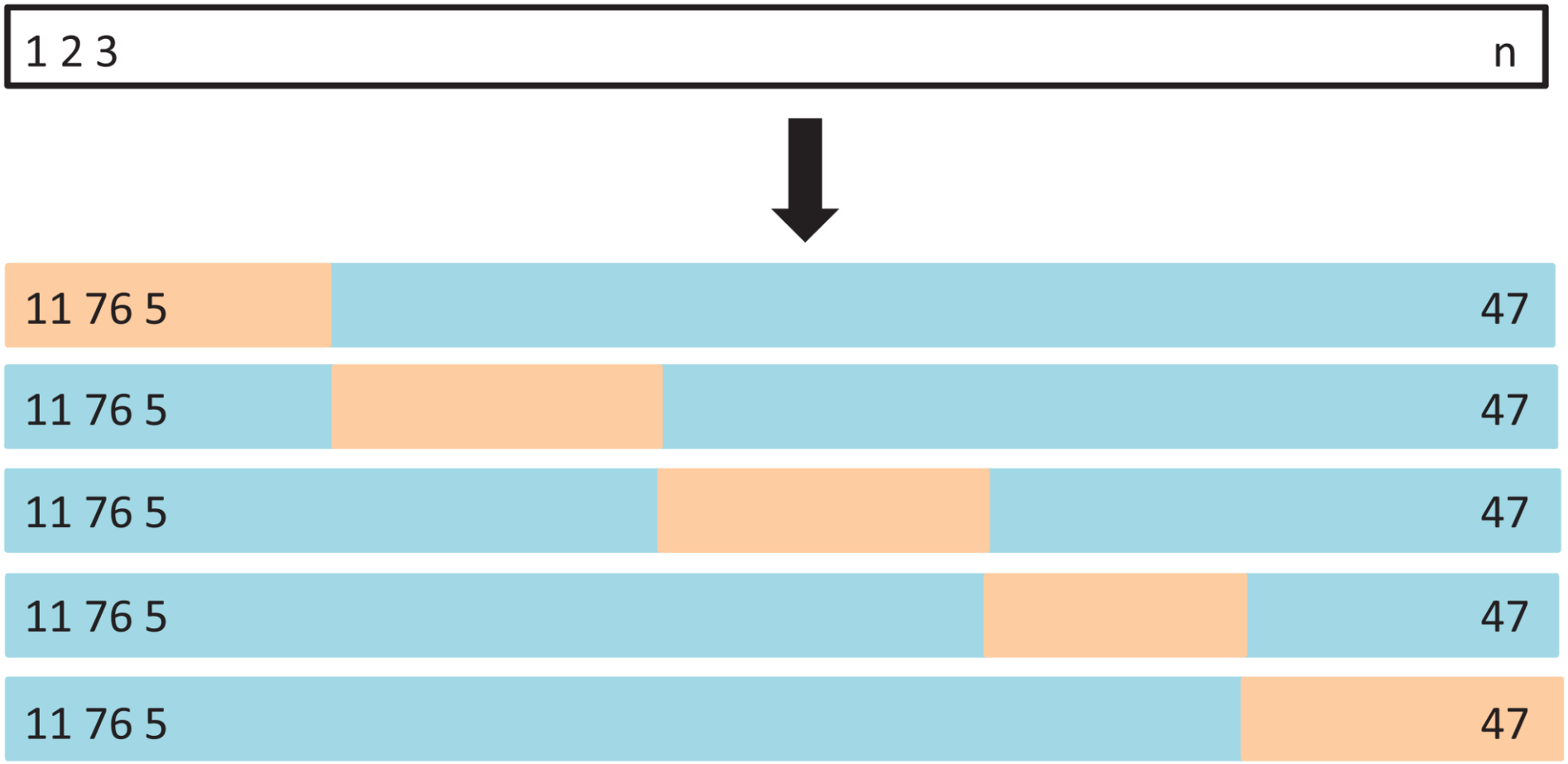
LOOCV has substantially less bias, since the model is repeatedly trained on a large sample set of size $N - 1$ training observations, whereas the validation set approach trains models with a sample size of approximately $N/2$ training observations.

Because of this, LOOCV tends to not overestimate the test error.

Moreover, running the LOOCV procedure will always yield the exact same inferred model, there will not be randomness in the results to particular splits of training and validation sets as in the validation set approach.

However, LOOCV is computationally expensive, because the model needs to be trained N times.

Schematic of k-fold CV



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

k-fold cross-validation (CV)

k-fold CV is a compromise between the validation set approach and the LOOCV approach.

The training dataset is divided uniformly at random into k sets (or folds) of roughly equal size (*i.e.*, size of each set is roughly N/k).

The first set is held out for validation, and the remaining $k-1$ sets (sets $2,3,\ldots,k$) are used for training.

The MSE across the observations in the first set is used to measure the validation (test) error, and is denoted by MSE_1

k-fold cross-validation (CV)

This process is repeated for each set.

When set i is held out for validation, the remaining $k-1$ sets (sets $1, \ldots, i-1, i+1, \ldots, k$) are used for training.

The MSE across the observations in the ith set is used to measure the validation (test) error, and is denoted by MSE_i and the overall k-fold CV test error is computed as

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{MSE}_i$$

Computational savings with k-fold CV

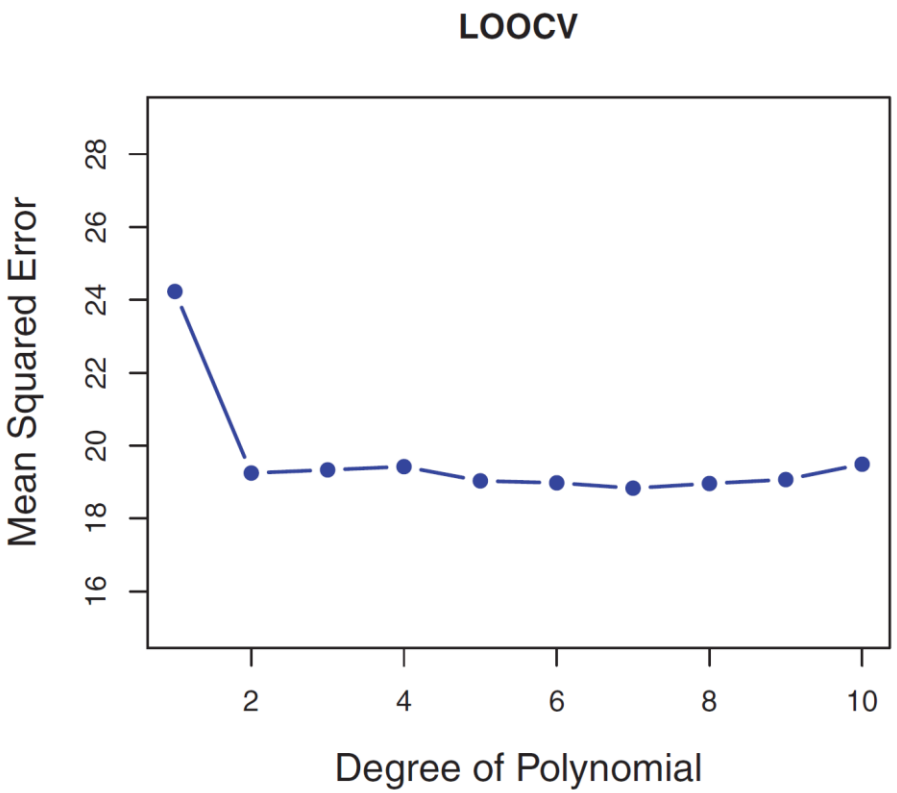
LOOCV is a special case of k-fold CV with $k = N$.

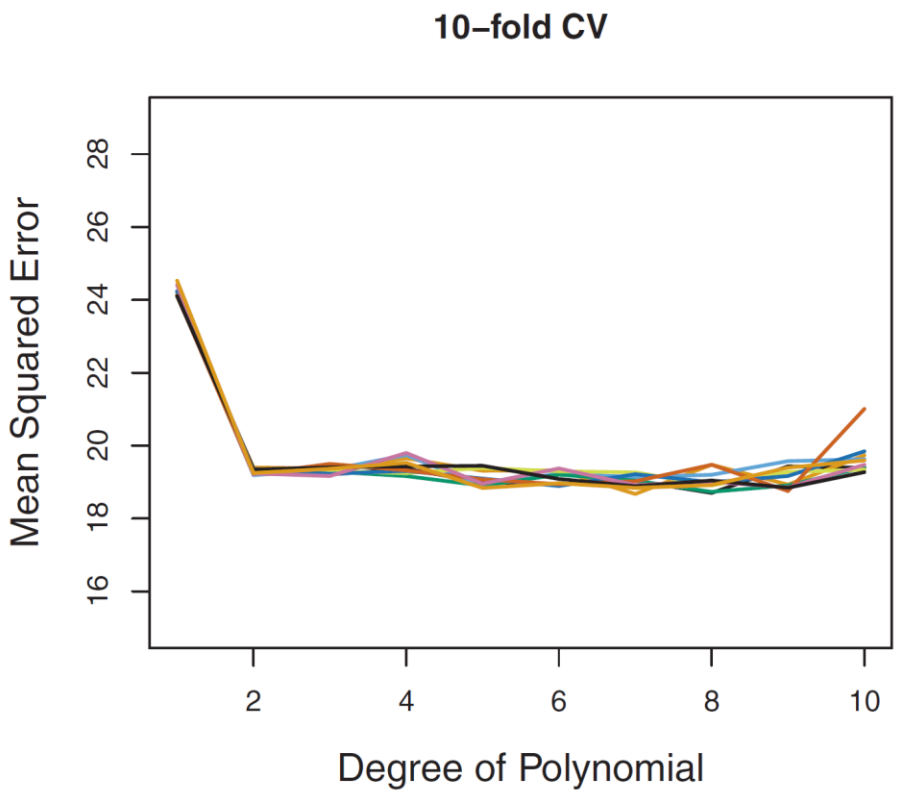
In practice, we usually use $k = 5$ or $k = 10$.

An advantage of 5- or 10-fold CV over N-fold CV (LOOCV) is the computational cost.

Here one is only training 5 or 10 models, rather than N, which could lead to enormous savings for those methods that are computationally intensive to fit.

N- vs 10-fold CV on polynomial regression





There is some variability across the nine 10-fold CV estimates.

But this variability is typically substantially lower than the variability in test sets from the validation set approach.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Non-computational advantage of k-fold CV ($k < N$)

Because LOOCV uses a greater number of observations in each training set, its model estimates will be based on a larger sample size, and should exhibit smaller bias relative to k-fold CV ($k < N$).

However, LOOCV has a higher variance than k-fold CV ($k < N$).

LOOCV is essentially averaging the output of N fitted models, each of which is highly positively correlated with the others (as almost all their training observations overlap).

k-fold CV is averaging the output of k fitted models, each of which is partially correlated with the others.

Non-computational advantage of k-fold CV ($k < N$)

The mean of many highly correlated quantities has higher variance than the mean of many quantities that are less correlated.

Because of this, the test error estimates resulting from k-fold CV ($k < N$) tend to be smaller than those from LOOCV.

Therefore, there is a bias-variance trade-off depending on the chosen value for k in k-fold CV.

For these reasons, 5- and 10-fold CV is often performed, as they have been demonstrated empirically to provided estimates of the test error that do not experience high bias or high variance.