COMP9033
DATA ANALYTICS

6/12

LINEAR REGRESSION

DR. DONAGH HORGAN

DEPARTMENT OF COMPUTER SCIENCE
CORK INSTITUTE OF TECHNOLOGY

2018.03.06

Overview

1. Data modelling:
    - Rule-based models
    - Statistical models.
    - Machine learning.
2. Sources of model error:
    - Underfitting and overfitting.
    - Bias, variance and irreducible error.
    - The bias-variance trade off.
3. Cross validation:
    - Split size.
    - Exhaustive vs. non-exhaustive.
    - Cross validation techniques.
    - Stratification.
4. Model selection:
    - Choosing the optimal model.
    - Grid search.
    - Nested cross validation.

1. Linear regression:
   - What it is.
   - How it works.
   - Measuring model error.
2. The least squares technique:
   - The residual sum of squares.
   - The least squares solution.
   - Performance considerations.
3. Shrinkage methods:
   - Problems with least squares.
   - Ridge regression.
   - Hyperparameters.
4. Subset selection:
   - Forward stepwise selection.
   - Backward stepwise selection.
   - Hybrid methods.

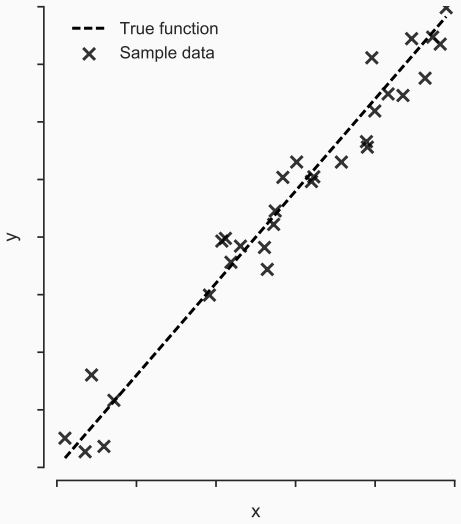
Linear regression

- The term *linear regression* describes a class of mathematical models that can be used to describe *quantitative* data.
- Generally, machine learning algorithms are used to build linear regression models, *e.g.*
  - The least squares technique.
  - Ridge regression.
  - The lasso technique.
- These are *supervised* machine learning algorithms, *i.e.* they learn from labelled data using both statistics and heuristics.

- One familiar example of linear regression is the fitting of a straight line, *i.e.* the estimation of m and c in the equation

$$y = mx + c. \tag{6.1}$$

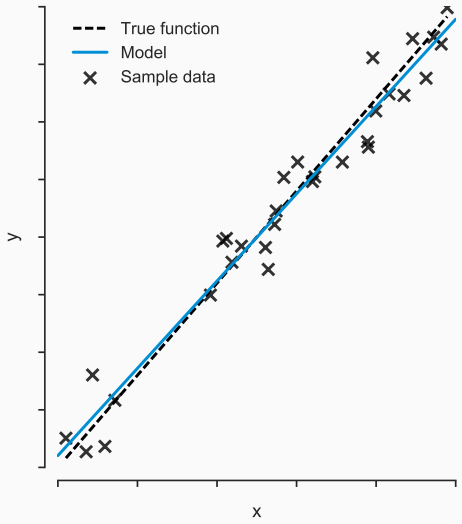
- For instance, the figure opposite shows some data that has been noisily sampled from the function $y = 2x + 1$.



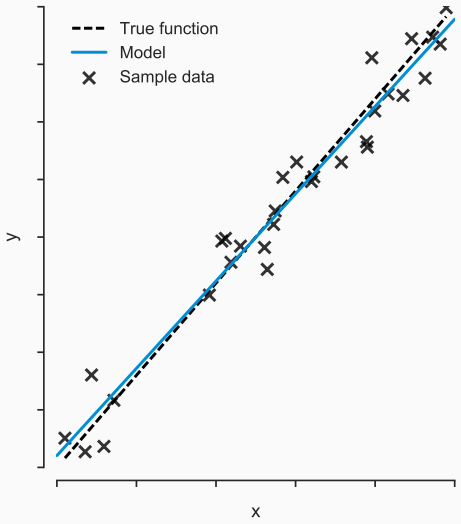
- Using linear regression, we can create a model of a line that fits this data, *e.g.*

$$\hat{y} = 1.9x + 1.05,$$

where $\hat{y}$ represents a *prediction* of the true value y, *i.e.* $\hat{y} \approx y$.



- More generally, we would like to fit coefficients to *many* features, *e.g.*
  - Predict temperature based on atmospheric pressure *and* wind speed (*two* features).
  - Predict server CPU load based on the number of active users, network throughput and disk I/O (*three* features).
  - Predict the price of Apple stocks based on the prices of other stocks (an *arbitrary* number of features).



- Linear regression makes predictions based on multiple features.
- Typically, we have k features and we want to estimate some *quantitative* output function y, also known as the target.
- Linear regression does this by fitting an intercept (β_0) and k coefficient values ($\beta_1, \beta_2, \ldots, \beta_k$) to the data[1] as

$$\hat{y} = \beta_0 + \sum_{j=1}^{k} \beta_j x_j, \tag{6.2}$$

where $\hat{y}$ is the predicted value of y.

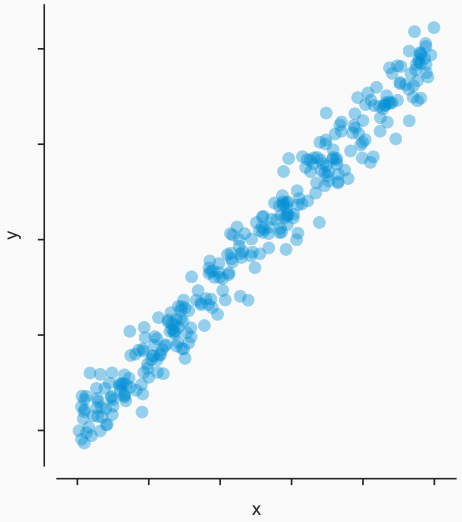
[1] In fact, Equation 6.1 is a just special case of Equation 6.2 with $k = 1$, $\beta_0 = c$ and $\beta_1 = m$.

- One advantage of linear regression is that the model can be expressed mathematically:
  - The contribution of each feature can be measured by its coefficient[2].
  - The model is relatively straightforward to explain to a non-technical audience, *e.g.* customers, management.
- Features can take a number of forms, but must be *quantitative*, *e.g.*
  - An arbitrary quantitative input variable, *e.g.* temperature.
  - A binary indicator variable encoding a categoric input, *e.g.* via one hot encoding.
  - A mathematical transformation of an input variable, *e.g.* $\sqrt{x_1}$ or $\log x_2$.
  - An interaction between input variables, *e.g.* $x_3 = x_1 \cdot x_2$.
- Transformations and interactions can be used to account for non-linear relationships between the features and the target.

[2] For this, features *must* be standardised before computing the regression coefficients.

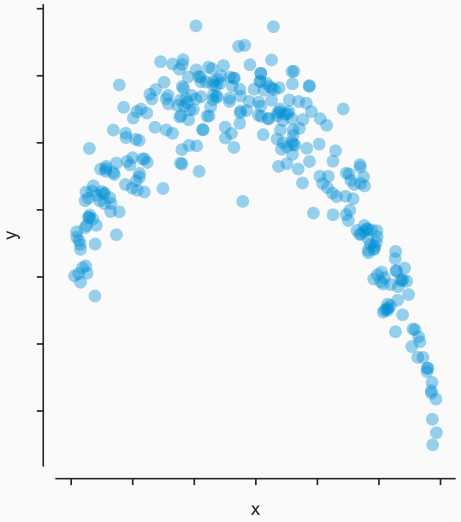
- Linear regression works well when we are dealing with *linear* systems, *i.e.* systems where there is a linear dependency between y and x, *e.g.*

$$y = 10x,$$
$$y = 2x_1 + 4x_2,$$
$$y = 3.5x_1 + 2x_2 + 4x_3 + 17.$$

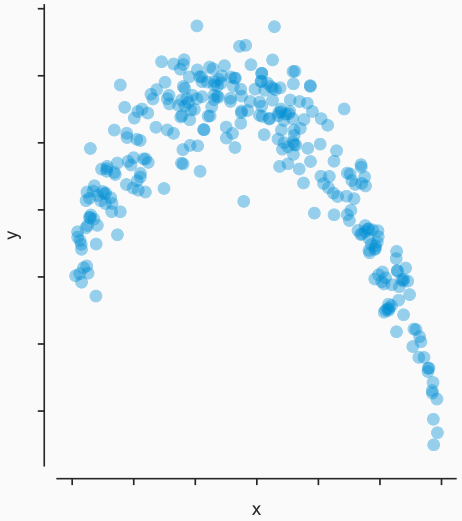


- However, things get a little trickier if we are dealing with *non-linear* systems, *i.e.* systems where y is a function of *powers* of x variables, *e.g.*

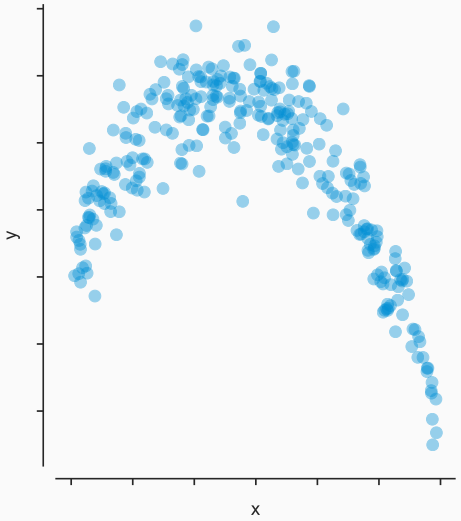
$$y = 10x^2,$$
$$y = 2x_1^3 + 4x_2,$$
$$y = 3.5x_1^5 + 2x_2^3 + 4\sqrt{x_3} + 17.$$



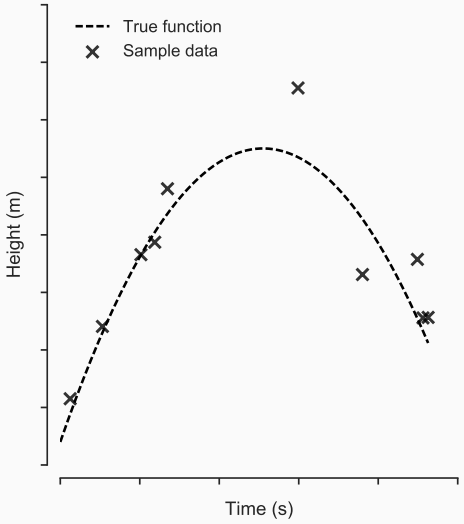
- This is because Equation 6.2 computes predictions based on *linear* combinations of the features.
- If this assumption is violated, *e.g.* the relationship between *y* and its features is non-linear, Equation 6.2 will not produce a reasonable prediction.



- If we know that there is a non-linear relationship between *y* and one of its features, then we can use *feature generation* to create a new feature that adequately describes the relationship.
- We can then supplement or replace the original feature with the new feature, using model selection to determine which choice is better when the answer is not obvious.



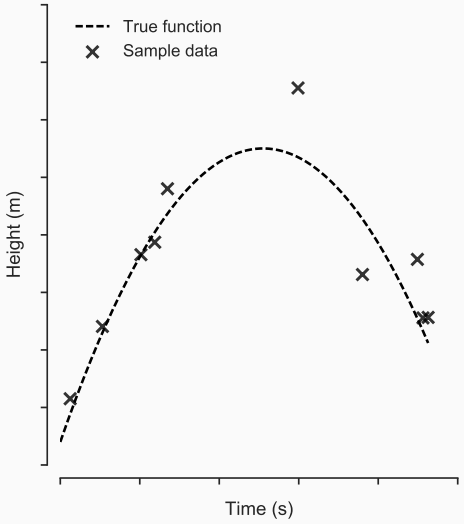
- In an experiment, a rocket is projected into the air at a speed of 50 m s^{-1} from a height of 10 m.
- As the rocket travels through the air, its height above ground level is measured.
- What height is the rocket at after an arbitrary period of time, *t*?



- The true relationship between the height (h) and time (t) is well-known from the laws of physics:

$$h = -4.9t^2 + 50t + 10.$$

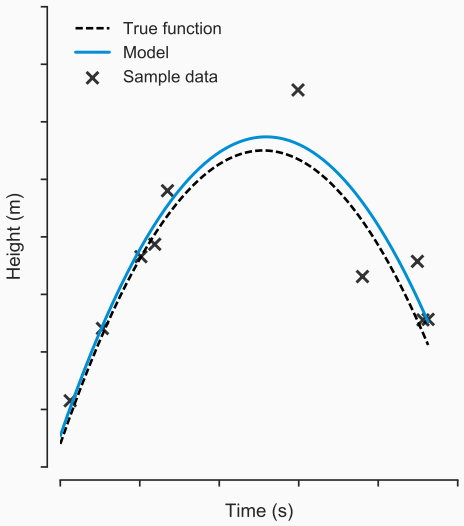
- However, we can approximate it using linear regression and feature selection.



- As we know that there is a non-linear (in this case, quadratic) relationship between h and t, we cannot use linear regression without feature generation.
- If we define the features $x_1 = t$ and $x_2 = t^2$ (*i.e.* generate it from x_1), then we can use linear regression to build a model:

$$\hat{h} = -4.8t^2 + 50.2t + 13.5,$$

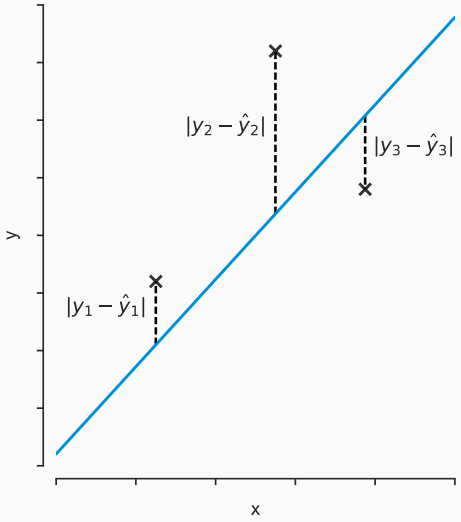
which, as can be seen, is a reasonably accurate approximation.



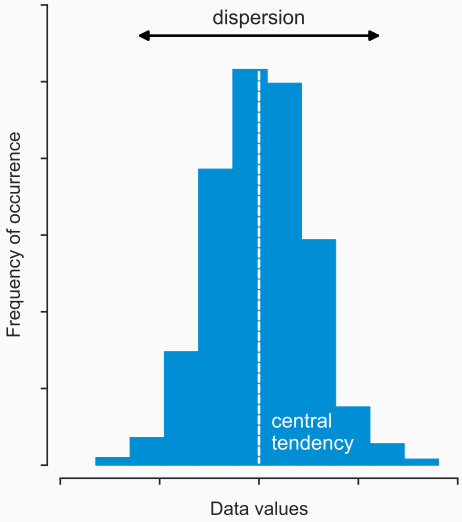
- If we use a linear regression model to predict a single value $\hat{y}_i \approx y_i$, then the prediction error, ϵ_i, is given by:

$$\epsilon_i = y_i - \hat{y}_i. \tag{6.3}$$

- Using Equation 6.3, we can infer that:
  - If $\epsilon_i < 0$, then we have *overestimated* the real value.
  - If $\epsilon_i > 0$, then we have *underestimated* the real value.
  - If $\epsilon_i = 0$, then our prediction was completely accurate.

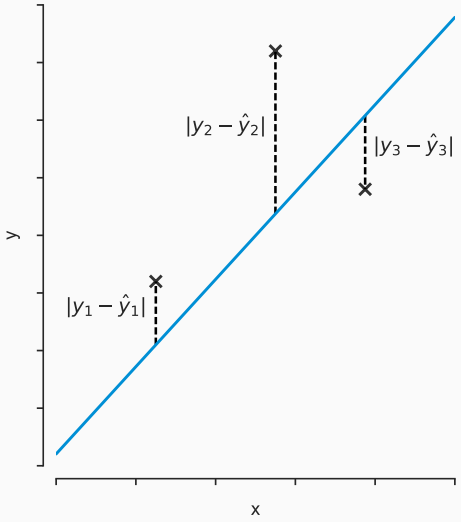


- Equation 6.3 is useful when dealing with single sample values, but we will usually generate many sample predictions in order to validate our model, *e.g.* if we use a test set in cross validation.
- We could simply average the errors, but this doesn't take their variation into account (*e.g.* low bias, high variance).
- Need some way to measure the *magnitude* of the errors.



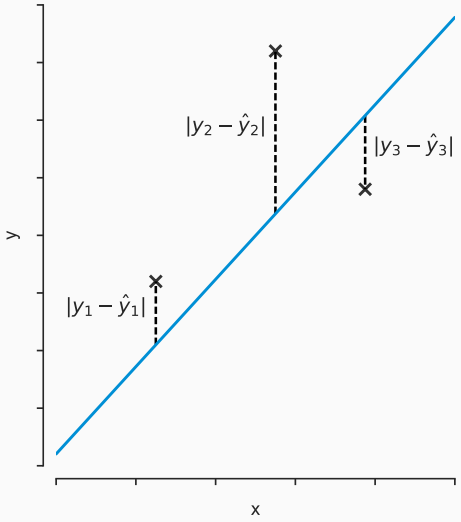
- The *mean absolute error* (MAE) is one way to do this: it simply averages the absolute values of the sample errors, *i.e.*

$$\begin{aligned} \text{MAE} &= \frac{1}{n}\sum_{i=1}^{n} |\epsilon_i| \\ &= \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|. \end{aligned} \tag{6.4}$$



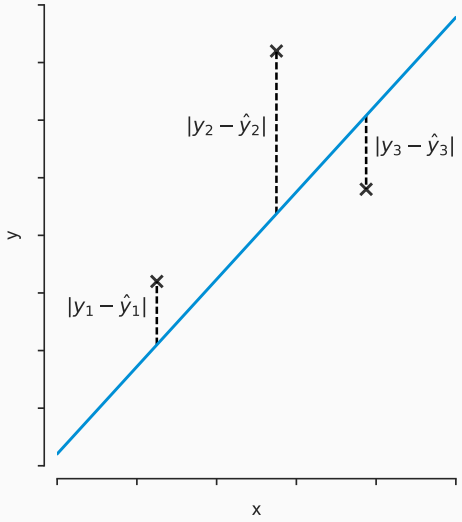
- The *root mean square error* (RMSE) is another way to do this: it is the square root of the average of the squared errors, *i.e.*

$$\begin{aligned} \text{RMSE} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\epsilon_i^2} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \end{aligned} \tag{6.5}$$

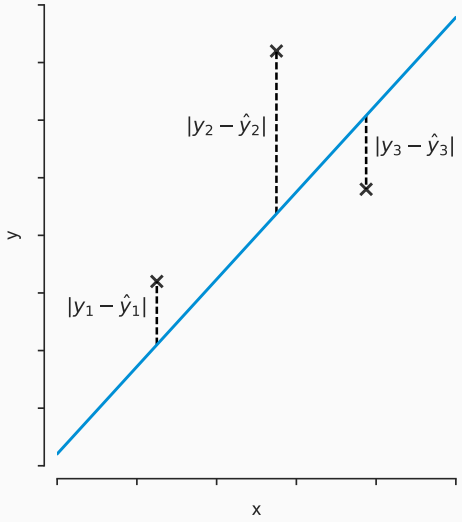


- MAE relies on the absolute value of the errors ($|\epsilon_i|$), whereas RMSE relies on the squares of the errors (ϵ_i^2).
- This is an important distinction as

$$\epsilon_i < 1 \implies \epsilon_i^2 << 1,$$
$$\epsilon_i = 1 \implies \epsilon_i^2 = 1,$$
$$\epsilon_i > 1 \implies \epsilon_i^2 >> 1.$$



- Consequently, if a sample contains a small number of large errors, then its RMSE tends to be larger than its MAE.
- This can be exploited to choose a model that produces fewer larger errors (*i.e.* with lower variance error), at the cost of producing more smaller ones (*i.e.* with higher bias error).



The least squares technique

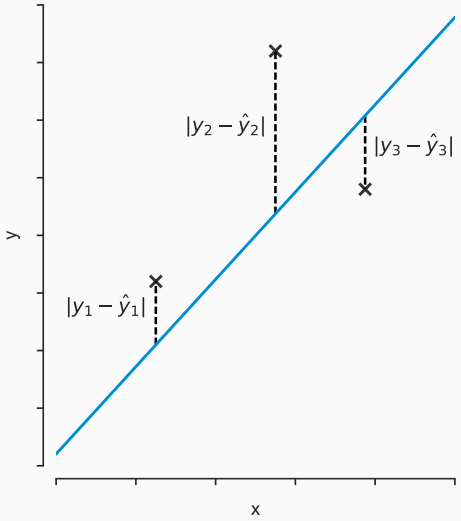
- The *least squares technique* is a commonly used method for estimating the intercept and model coefficients in Equation 6.2.
- It does this by attempting to minimise a quantity known as the residual sum of squares (RSS), *i.e.*

$$\text{RSS} = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} \epsilon_i^2. \tag{6.6}$$

- Noting the definition in Equation 6.5, the RSS is related to RMSE as

$$\text{RSS} = n\text{RMSE}^2. \tag{6.7}$$

- The RSS is simply the sum of the squared distances between a candidate fit line and the sample data.
- Consequently, if we can choose our intercept and model coefficients well, then we can minimise our RSS and have a good chance of finding a line that fits the data well.
- Effectively, least squares reverses this process: by minimising the RSS, we hope to determine the intercept and coefficients that give the best fit line.



- When we have a data set with k features and n samples, it's typical to arrange it as a matrix, $\boldsymbol{X}$, where each column corresponds to a feature and each row corresponds to a sample, *i.e.*

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,k} \\ x_{2,1} & x_{2,2} & \dots & x_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,k} \end{bmatrix}, \tag{6.8}$$

where $x_{i,j}$ represents the i^{th} sample value for the j^{th} feature.

- Under this matrix-vector formulation, Equation 6.2 can be written as

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}, \tag{6.9}$$

  where $\hat{\boldsymbol{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$ is a column vector of the predicted values for each input sample and $\boldsymbol{\beta} = \{\beta_1, \beta_2, \ldots, \beta_k\}$ is a column vector of the coefficients for each feature.
- However, Equation 6.9 does not account for the intercept term, β_0!

- In order to include the intercept term (β_0) in the calculation, an all-ones feature column is usually prepended to $\boldsymbol{X}$, *i.e.*

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,k} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \dots & x_{n,k} \end{bmatrix}. \tag{6.10}$$

- As the all-ones feature is constant across all samples, its corresponding coefficient will be the intercept term (by definition).
- Consequently, if Equation 6.10 is used with Equation 6.9, then the resulting coefficient vector will be $\boldsymbol{\beta} = \{\beta_0, \beta_1, \dots, \beta_k\}$, *i.e.* the intercept will be computed also.

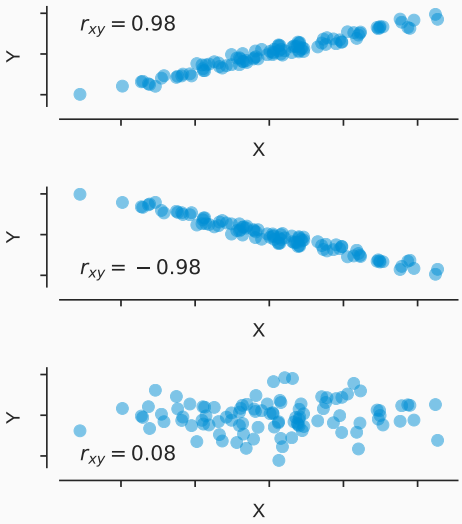
- Using Equation 6.9, it can be shown that the value of $\boldsymbol{\beta}$ that minimises the RSS is given by

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{6.11}$$

  where $\boldsymbol{X}^T$ is the transpose of the matrix $\boldsymbol{X}$ and $\boldsymbol{X}^{-1}$ represents the inverse of the matrix $\boldsymbol{X}$.
- Equation 6.11 is known as the *least squares* solution and is the *optimal* choice of $\boldsymbol{\beta}$ among all *unbiased* estimates.

- In practice, the number of samples we are dealing with is often large and the number of features may be large too.
- In such cases, computing Equation 6.11 by matrix multiplication can become very computationally expensive!
- In general, there are two solutions to this problem:
  1. Use an optimised matrix multiplication library, *e.g.* BLAS.
  2. Use numerical methods (*e.g.* stochastic gradient descent) to find an *approximate* solution.
- The trade off is between exactness and speed: numerical methods generally arrive at a solution faster than direct solvers, but can sometimes arrive at a local optimum rather than a global optimum — which will never happen with a direct solution.

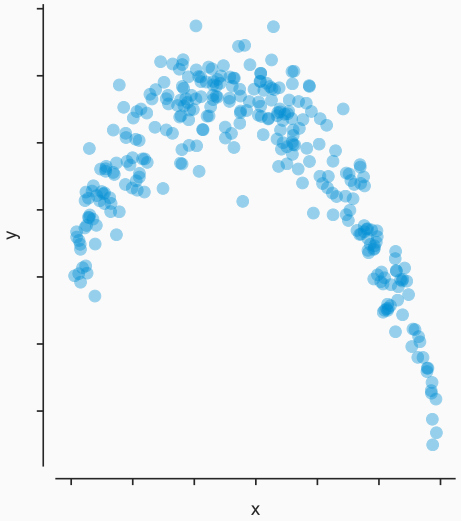
- The term *multicollinearity* describes situations when two or more features are highly correlated.
- This can have negative consequences when training linear regression models:
  - In general, there is greater a risk of overfitting.
  - In extreme cases, the algorithm may find a very unstable solution.
  - In cases with perfect multicollinearity, the algorithm may not be able to find *any* solution.



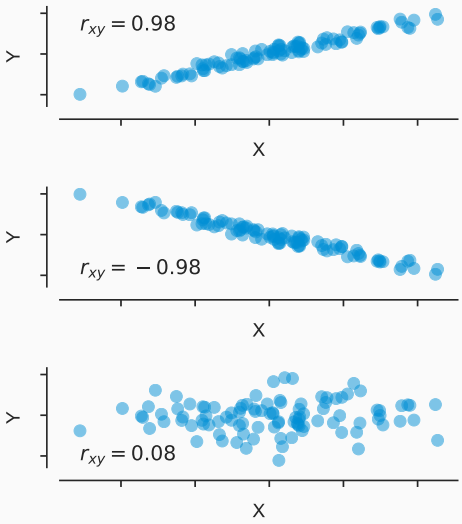
- One common cause of multicollinearity is one hot encoding.
- For instance, in the table to the right, the correlation between the IS MALE and IS FEMALE columns is -1, *i.e.* they are perfectly negatively correlated.
- Generally, this can be mitigated by dropping *any* one of the encoded features from the analysis.

	AGE	IS MALE	IS FEMALE
Alice	36	0	1
Bob	58	1	0
Carol	22	0	1

- A second common cause of multicollinearity is feature generation.
- This can happen when new features are generated from existing features (*e.g.* transformations, interactions) and *both* the new and existing features are included in the model.
- Generally, this can be mitigated by *standardizing* the input features before fitting the model.

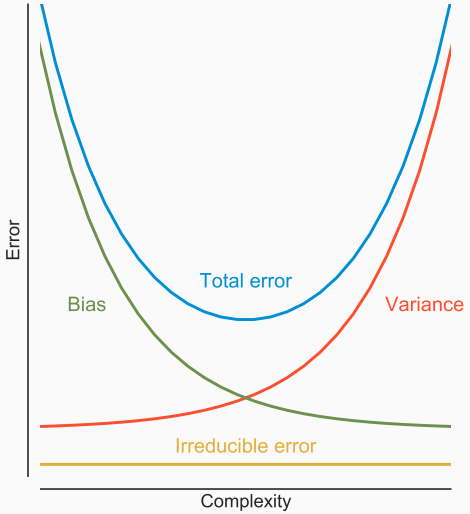


- More generally, the effects of multicollinearity can be mitigated by the use of:
  1. Shrinkage methods, *e.g.* ridge/lasso regression.
  2. Feature selection techniques, *e.g.* forward/backward stepwise selection.
  3. Other dimensionality reduction techniques, *e.g.* principal component analysis.

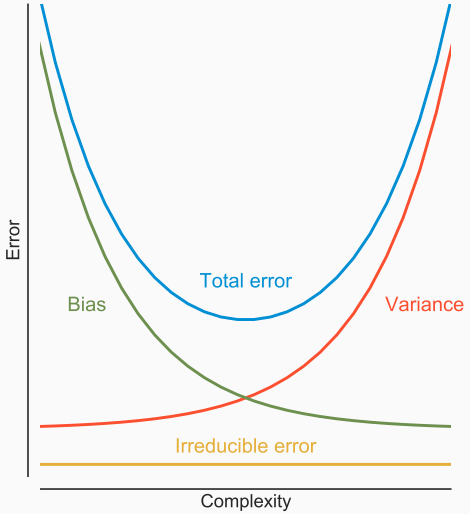


Shrinkage methods

- Least squares models tend to have low bias error but high variance error, *i.e.* they often overfit the data.
- If we use an overfitted model, then we are more likely to make mistakes when we evaluate new data.
  - Overfitted models tend to be unstable.
  - Small deviations in the magnitude of the input can produce large deviations in the magnitude of the output.



- Typically, this flaw is overcome by deliberately *biasing* the regression.
  - By building a model with increased bias error, we *should* lower our variance error.
  - If the model is overfit, then this will result in a model with lower overall error.
- Biasing typically has the effect of reducing the magnitude or number of features of a linear regression model, and vice-versa.



- Shrinkage methods aim to reduce the magnitudes of the linear regression coefficients, so that they have less of an effect on the final prediction:
  - If a feature is unimportant, then its coefficient may shrink to zero, in which case it is eliminated (see Equation 6.2).
  - If a feature is not *very* important, but still adds *some* value, then the magnitude of its coefficient is reduced, and so it has less of an effect on the predicted value, but does still have an effect.
  - If a feature is important, then the magnitude of its coefficient should remain more or less unchanged.
- The use of shrinkage methods can mitigate the effects of multicollinearity, if present.
- Examples include ridge regression and the lasso technique.

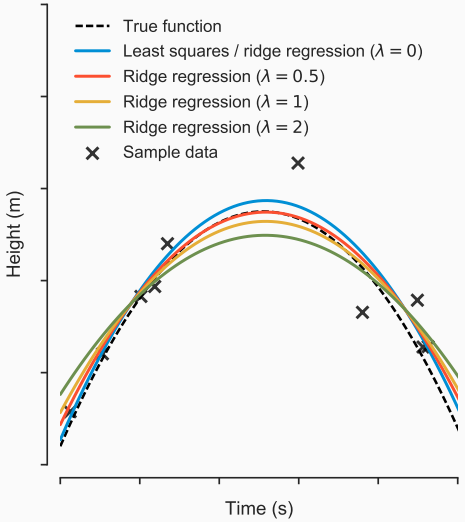
- Ridge regression is a form of shrinkage where the regression coefficients are computed as

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{6.12}$$

  where $\boldsymbol{I}$ denotes an *identity matrix* with the same number of columns as $\boldsymbol{X}$ and $\lambda \in [0, \infty)$ is known as the ridge parameter.
- By adjusting the value of λ, we can directly introduce bias error into the model which, ideally, will move it closer to having minimum total error:
  - The *smaller* the value of λ, the *lesser* the amount of bias/shrinkage.
  - The *larger* the value of λ, the *greater* the amount of bias/shrinkage.
  - When $\lambda = 0$, ridge regression is equivalent to the least squares technique.

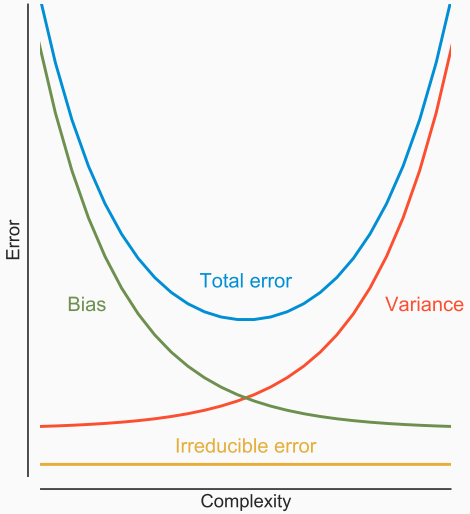
- The ridge parameter is an example of a *hyperparameter*, *i.e.* it is an adjustable parameter that controls some aspect of the model building process.
- Generally, adjusting the hyperparameters of a model building algorithm affects the quality of model that is produced.
- Using model selection via cross validation, we can determine the optimum value of a particular hyperparameter or set of hyperparameters and, therefore, choose the optimal model.

- If we apply ridge regression to our earlier rocket trajectory problem, we can produce a number of variations on the least squares fit.
- As λ increases, the fit becomes more biased.
- Can use model selection to pick the best model.



Subset selection

- The term *subset selection* describes a class of algorithms that can be used to select a subset of features to build a model with.
- Selecting just a subset of features should produce a less complex model which should, in turn, reduce the error from variance.
- Subset selection can mitigate against the effects of multicollinearity.

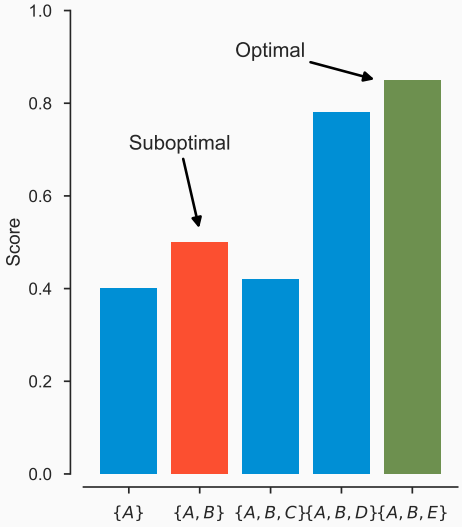


- *Forward stepwise selection* is a kind of subset selection algorithm.
- It takes a *bottom up* approach, as follows:
  1. Start with a data set consisting of only the all-ones feature. Build a model on this data set and measure its error.
  2. Add a feature to the data set, build a model and measure its error. Remove the feature from the data set once this is complete.
  3. Repeat Step 2 for each of the remaining features and evaluate the error resulting from each of the generated models. Choose the set of features that led to the greatest reduction in error.
  4. Repeat Steps 2 and 3 until the error ceases to decrease or a desired number of features is reached.

- *Backward stepwise selection* is an alternative subset selection algorithm.
- It takes a *top down* approach, as follows:
  1. Start with a data set consisting of all of the available features. Build a model on this data set and measure its error
  2. Remove a feature from the data set, build a model and measure its error. Add the feature back into the data set once this is complete.
  3. Repeat Step 2 for each of the remaining features and evaluate the error resulting from each of the generated models. Choose the set of features that led to the greatest reduction in error.
  4. Repeat Steps 2 and 3 until the error ceases to decrease or a desired number of features is reached.

- Both forward and backward stepwise selection are forms of model selection.
- However, they have very different characteristics:
  - Forward stepwise selection may halt before reaching a particular combination of features that gives a big decrease in error, but tends to be faster when there are many redundant features.
  - Backward stepwise selection works back from the full set of features and so never misses a valuable feature, but tends to be slower when there are many redundant features.
- In addition, backward stepwise selection can only be used when the number of samples is greater than the number of features, although this is usually not a problem in many cases.

- Forward and backward stepwise selection are both forms of *hill climbing*.
- Consequently, the final result is not guaranteed to be optimal in all cases!
- Only brute force feature selection is guaranteed to produce an optimal choice, but this usually requires significantly more computational resources.



- Subset selection excludes entire features from a model:
  - Reduces complexity, but can drop valuable features.
  - If too many features are dropped, there is a risk of underfitting (high bias).
- Shrinkage methods reduce the effect of features in a model:
  - Reduces complexity, but can include poor features if coefficients aren't shrunk to zero.
  - If too many features are retained, there is a risk of overfitting (high variance).
- Subset selection can be combined with shrinkage methods to get the benefits of both, *e.g.* forward stepwise selection with ridge regression.
- Again, can use model selection to optimise hyperparameters (ridge parameter, subset size) as required.

Summary

- Linear regression:
  - Least squares: straightforward, but tends to overfit (no hyperparameters).
  - Shrinkage methods: reduce the effect of weak features.
  - Subset selection: eliminate weak features entirely.
  - Hybrid methods: combine benefits of subset selection and shrinkage methods.
- This week's lab:
  - Build a linear regression model using the least squares method.
  - Engineer new features to create a model with lower overall error.
- Next week: decision tree classification and regression.

1. Hastie et al. *The elements of statistical learning: data mining, inference and prediction.* 2nd edition, February 2009. (stanford.io/2i1T6fN)
2. Ullman et al. *Mining of massive data sets.* Cambridge University Press, 2014. (stanford.io/1qtgAYh)