MODULE 2 UNIT 1
Machine learning model mechanics

IN COLLABORATION WITH getsmarter™

Table of contents

Learning outcomes:

LO1: Recognise machine learning models and the way they are fitted.

LO2: Demonstrate an understanding of the training error and test error functionality in machine learning performance.

1. Introduction

The goal of machine learning is to develop a model that is suitable to solve a specific problem by accurately generating an output from the input data. The output can then be used to predict what the actual outcome might be. To do this, it is essential to understand what a machine learning model is, how it is trained to generate the desired output, and how it is fitted onto the data set to solve the problem.

This lesson interrogates the meaning of a machine learning model and how such a model is fitted onto a data set with specific reference to overfitting, underfitting, cost functions, training errors, and test errors. The general discussion also explores the mechanics of two practical examples of machine learning models, taking a closer look at *k*-nearest neighbours (KNN) and polynomial regression.

2. Definition of a machine learning model

In some instances, the terms “machine learning model” and “machine learning algorithm” are used interchangeably. This is not an accurate use of these two terms. A model is a mathematical representation of a real-world process of mapping inputs into outputs. The model is trained to generate this output by using training data (Bhattacharjee, 2017; Koehrsen, 2018). An algorithm is the hypothesis that is noted before testing the model with real-world data (Bhattacharjee, 2017).

Having gained a better understanding of what a machine learning model is, the next section provides an in-depth discussion of two practical examples of such models.

3. Examples of machine learning models

This section discusses two practical examples of machine learning models. The first focuses on the *k*-nearest neighbours (KNN) classifier used for classification, while the second focuses on polynomial regression used for continuous variables.

3.1 *k*-nearest neighbours (KNN)

The KNN classifier is a supervised machine learning model that can be used to solve both regression and classification problems, although it will only be applied to solve a

classification problem in this lesson. Recall the difference between a classification problem and a regression problem.

Reminder:

A regression problem typically consists of real or continuous output values. A classification problem has a discrete value as the output. An example of this is a person's like or dislike of a particular brand of hummus (Harrison, 2018).

Figure 1 illustrates the method that underpins the KNN classifier to classify data into groups.

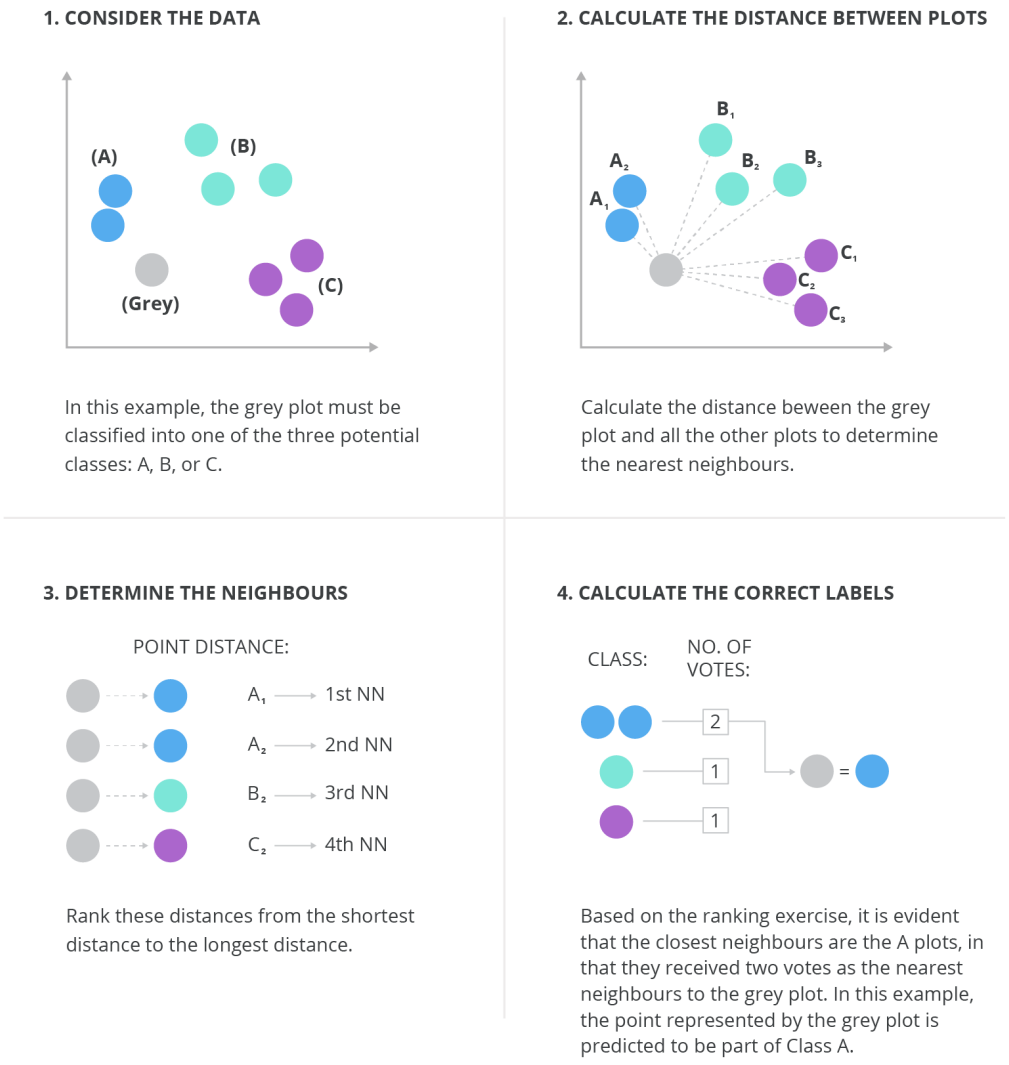


Figure 1: A simple example of a KNN classifier at work.

The KNN classifier, as depicted in Figure 1, functions based on the assumption that similar data points appear close to one another. This assumption is applied to determine the most appropriate group or class into which a new data point can be classified. The distance between data points is also referred to as proximity or closeness (Harrison, 2018).

To calculate the similarity between data points using the KNN classifier, use the following steps:

- **Step 1:** Initialise *k* to the chosen number of neighbours.
- **Step 2:** Calculate the distance between the data point to be classified and every other data point.
- **Step 3:** Rank the calculated distances from the closest to the farthest.
- **Step 4:** Select the closest neighbours from the ranked list.
- **Step 5:** Retrieve the correct labels of the selected neighbours to classify the unknown data point into the correct class.
- **Step 6:** A metric for distance must be chosen, and the variables must be scaled (so they are all on the same range of values) before running the algorithm.

(Harrison, 2018)

The value of *k* is used to determine the number of data points closest to a new input that must be classified. For example, if the aim is to determine whether juice is grape or apple flavoured, the closest neighbours to the unknown juice (the unknown data point) are used to group the data point into either grape juice or apple juice. The classification is based on the number of data points in each class located closest to the unknown data point (Subramanian, 2019). To correctly classify the data point, it is important to choose the correct number of neighbours – the value of *k* – to consider. A simple illustration of this classification process is shown in Figure 2. This figure uses $k = 3$ and $k = 6$.

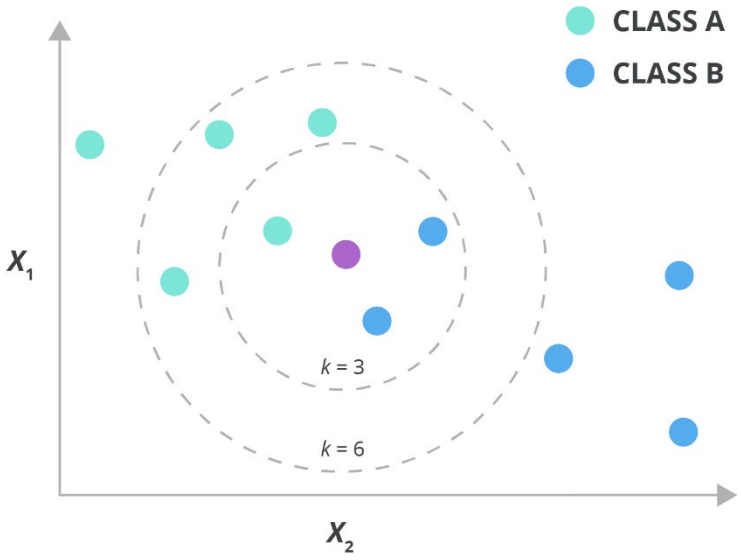


Figure 2: An example of the KNN classifier.

Notice how the class into which the unknown data point is classified changes as the value of *k* is increased from 3 to 6. In this example, the value of *k* therefore has a direct impact on the classification of the data point.

3.1.1 The value of *k*

A key feature of the KNN classifier is that the correct value for *k* must be chosen – one that is suitable for the input data. The value of *k* has a significant effect on the KNN classifier's functionality and the accuracy by which it makes predictions (James et al., 2013:39).

To determine the optimal value for *k*, the KNN classifier is run several times with different values, and the value that returns the least number of errors is allocated (Harrison, 2018). However, be mindful of the following when selecting the value of *k*:

- **A decrease in accuracy:** As the value of *k* decreases to 1, the predictions become increasingly volatile, as a data point located closest to the new data point is not necessarily indicative of the latter data point falling into the former's class. More data points are needed to determine the likelihood of a new data point belonging to a particular class.

- **An increase in errors:** As the value of *k* increases, the predictions become more stable. However, this increase of *k* could return more errors. Test error will be discussed in the Unit 2 Lesson. Thus, the more neighbours that are considered when classifying a new data point, the more distant neighbours can have a negative impact on the outcome. The increase in errors indicates that the value of *k* is too high. The goal is to determine the optimal value of *k* to yield the best result.

- **Odd numbers:** When solving two-class classification problems by predicting the nearest neighbour based on the majority votes, *k* should be a value equal to an odd number to eliminate the possibility of a tie between classes.

(Harrison, 2018)

3.1.2 Advantages and disadvantages of the KNN classifier

The KNN classifier has both advantages and disadvantages to consider before fitting the model onto a data set. Some of the advantages include:

- **Simplicity:** The KNN algorithm is fairly simple and easy to apply. There is no need to build a model, tune the model parameters, or make any assumptions.
- **Versatility:** The KNN algorithm can be used for both classification and regression.

(Harrison, 2018)

Despite the advantages, KNN models become slower and less accurate as the number of variables increases. KNN might therefore not be the best option for complex data sets or instances where the prediction needs to be calculated quickly (Harrison, 2018).

Pause and reflect:

Now that you understand the basics of the KNN classifier, think about a problem within your context where it could be applied.

3.1.3 KNN in practice

To provide a practical application of the KNN model, the MNIST data set is used. In this data set, there are 70,000 observations of handwritten digits, separated into 60,000 training images and 10,000 testing images (LeCun, Cortes & Burges, n.d.). For computational reasons, only 5,000 of the images are used in this example.

The goal of the application is to accurately classify each handwritten observation into the correct digit class. As such, there are 10 potential classes (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) that need to be classified. Figure 3 displays some examples of these images, each shown on a matrix consisting of a grid of black, white, and a variety of grey blocks. The black sections show the highest frequency of the digit's location within the grid. Similarly, the white blocks indicate the lowest frequency of the digit being located within the grid. Simply put, the black blocks show the flow of the digit.

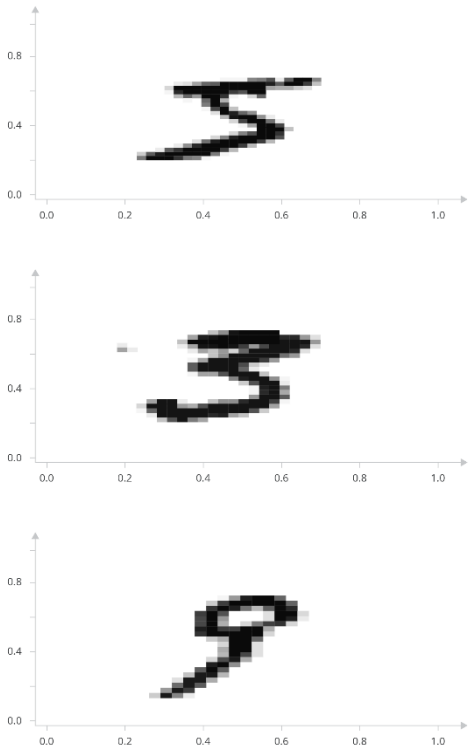


Figure 3: Examples of handwritten digits.

The KNN algorithm can be used to classify digits into different categories because some digits, as seen in Figure 3, have a greater frequency of pixels activated in certain areas, as you will recall from Module 1. For example, the pixel frequency for the number 4 and the number 8 differ significantly. Figure 4 shows where the pixels are concentrated for the numbers 4 and 8.

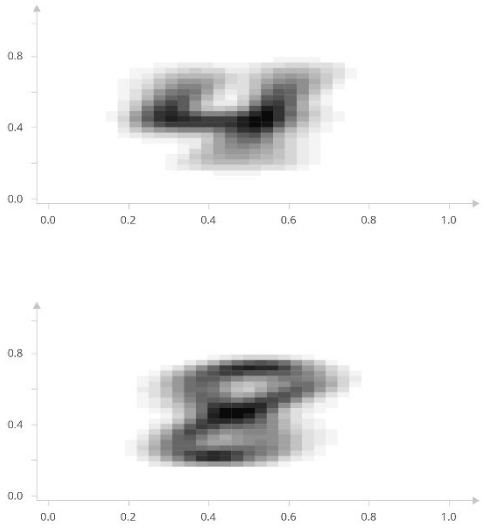


Figure 4: Concentration of pixel frequency.

The pixel concentration for each digit occurs in different areas. These 28×28 pixel images are collapsed into 784 (28×28) variables ranging from 0 (white) to 255 (black), each of which can be considered a feature to train the KNN model. To estimate the KNN model, an initial value of k needs to be chosen. In this example, $k = 5$ is used. Then, the model is estimated whereby each set of 784 colour intensity variables are labelled to correspond to a specific digit. Once the model has been estimated, it uses these corresponding explanatory variables to predict what digit the new data point, also consisting of 784 colour intensity pixels, might be.

After predicting the different data points, a confusion matrix is generated to plot the predicted classes against the actual classes. A confusion matrix measures the performance of a machine learning model where the output can be two or more classes. Essentially, the confusion matrix shows the number of classes that were predicted to form part of a specific group (Narkhede, 2018). For the classification of handwritten digits, the following confusion matrix is generated.

Table 1: The confusion matrix of the MNIST data set showing how the model classified each digit.

Digits	0	1	2	3	4	5	6	7	8	9
0	476	0	2	0	0	0	3	0	0	2
1	1	559	8	2	10	1	3	7	8	3
2	0	2	464	2	0	2	0	0	5	1
3	1	0	1	469	0	4	0	0	8	3
4	0	1	0	0	505	0	0	0	1	3
5	1	1	0	6	0	415	3	0	7	0
6	0	0	3	2	7	7	492	0	5	4
7	0	0	10	6	1	2	0	535	1	9
8	0	0	0	1	0	0	0	0	421	1
9	0	0	0	5	12	3	0	8	6	469

Each row shows what was predicted against the actual class of that prediction. For example, in predicting whether a certain digit is the number 0, 476 predictions were correct, 2 incorrectly predicted it as the digit 2, 3 as the digit 6, and 2 as the digit 9. Essentially, all the numbers on the diagonal line represent the correct predictions, whereas all off-diagonal cells represent the incorrect predictions. Notice how the digit 8 was only misclassified twice, once as the digit 3, and once as the digit 9. Both of these misclassified digits have similar shapes to the digit 8. Also notice how the digits 1 and 9 are both misclassified multiple times as the digit 4, due to the way in which they are written by hand.

To ensure a simple comparison between different models, various metrics can be used to determine a training model fit. A common metric used for classification problems is the accuracy metric. The accuracy metric shows the number of accurate predictions as a percentage of the total number of predictions. For example, using a value of $k = 5$ yields an accuracy of 96.18%. By expressing the accuracy as a percentage, it is possible to experiment with different values of k to improve the accuracy. The relationship between the value of k and the accuracy of the model of the MNIST data set is shown in Table 2.

Table 2: Relationship between the value of *k* and the model's accuracy in the MNIST data set.

Value of *k*	Accuracy of the model
3	97.30%
5	96.18%
7	95.30%

From Table 2, it is clear that for this particular example, the lower the value of *k*, the higher the training accuracy is, which implies a lower training error of the model. However, the lowest value of *k* does not always yield the best results. For instance, if $k = 1$, then the accuracy would become 100%. Although the accuracy of the model where $k = 1$ is 100%, these models are barely useful since this indicates overfitting. The goal is to find the value of *k* where the model accuracy is the highest for each particular data set.

Note:

Fitting the model onto the MNIST data set is illustrated in Video 1 later in this lesson.

3.1.4 Overfitting vs underfitting in KNN

Overfitting occurs when the model fits the training data too well and includes the noise in the data set to generate an output. This negatively impacts the performance of the model on new data. Overfitting is more likely to occur when non-parametric or non-linear models are fitted onto data, since they are more flexible (Brownlee, 2016). An example of a non-parametric model is the KNN model, and refers to a model that does not make an assumption about the data's distribution (Lin, 2018). An example of a non-linear model is polynomial regression (Brownlee, 2016), which is discussed in Section 3.2.

When a model learns from the training data and accounts for every training variable, the error will be low. These models appear to be perfect during training, but are grossly inaccurate later (Koehrsen, 2018).

In the previous section, it was mentioned that when $k = 1$, the accuracy of the KNN model would become 100%. Although a model accuracy of 100% appears beneficial, this model is not useful when making predictions unseen data. Therefore, the goal of the KNN classifier is to determine the value of *k* where the model accuracy is the highest, given a particular data set. Recall from Table 2 that the optimal value of *k* for the MNIST data set is 3. When $k = 3$, overfitting is avoided.

Underfitting, on the other hand, implies that not all the essential data points are considered when calculating a prediction, which greatly affects the model's accuracy. If a value of *k* is chosen that doesn't yield the highest accuracy rate for the model, that might lead to the

KNN model being underfitted. In Table 2, the value of k = 7, for example, would imply that the model is slightly more underfitted as opposed to the optimal value of k = 3 that yields the highest accuracy rate for the given model.

Now that you understand what the KNN classifier is, how it works, and the importance of the k-value, test your knowledge by answering the following quiz questions.

> At this point in the lesson, you have the opportunity to engage with a practice quiz to test your understanding of the content. Access this lesson on the Online Campus to engage with this quiz.

3.2 Polynomial regression

In Module 1, linear regression was introduced by explaining the basics of fitting a regression line onto input data. However, in some instances, the true relationship between the response and the predictors may be non-linear. This is when polynomial regression becomes useful (James et al., 2013:90). Consider the scatterplot shown in Figure 5.

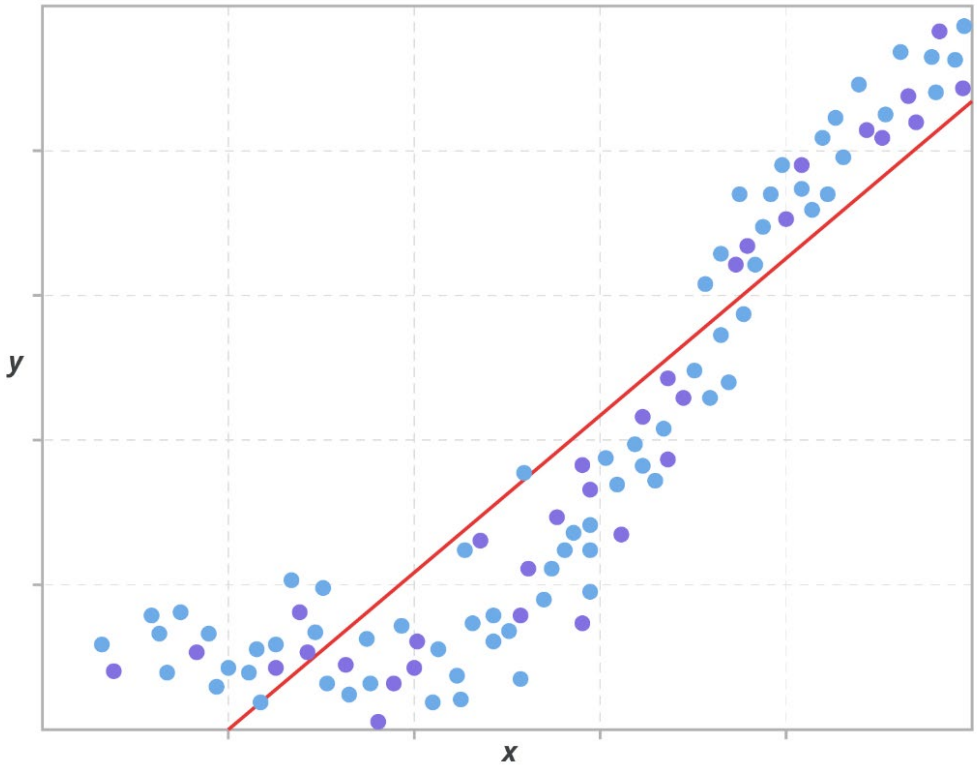


Figure 5: Linear model fitted onto non-linear data.

In Figure 5, the plotted points are scattered in such a way that a linear regression line is not an accurate fit. Fitting the linear regression line onto this data set results in a high error

rate. This is commonly referred to as underfitting (Pant, 2019). The better approach is to increase the complexity of the model by fitting a polynomial regression model onto that particular data set (Agarwal, 2018), as shown in Figure 6.

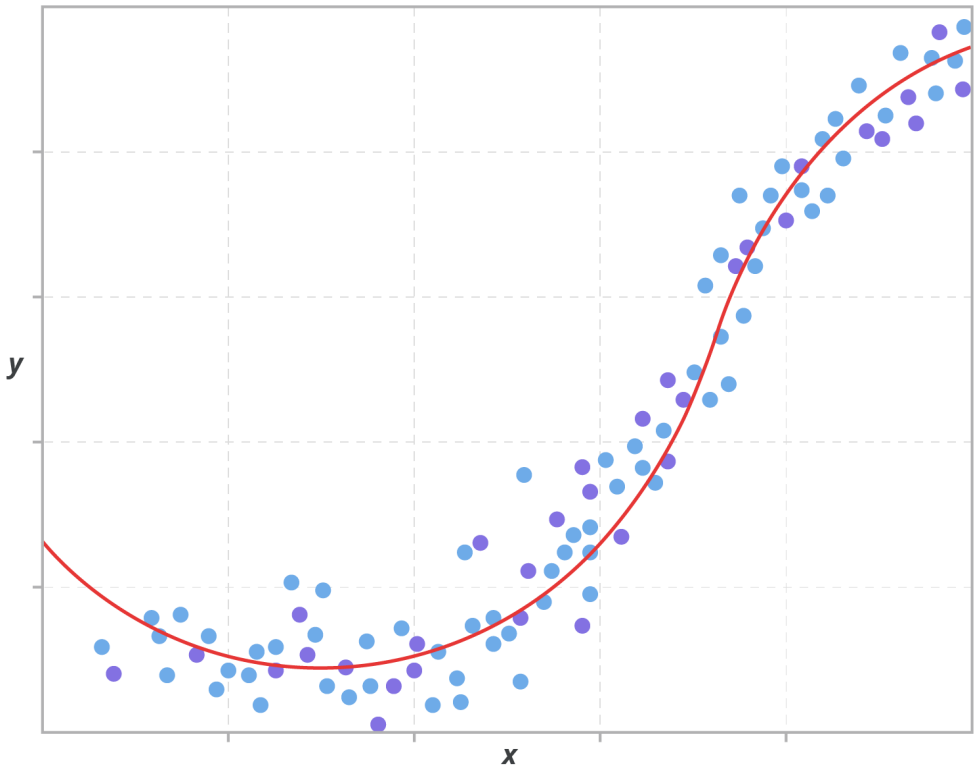


Figure 6: Example of a polynomial regression fit.

From Figure 6, it is clear that the curved line encompasses the different plots more accurately, eliminating underfitting. On the other hand, a model can also be overfitted.

3.2.1 Underfitting vs overfitting in polynomial regression

In Figure 5, the linear regression line is clearly not the best fit for the data set because it is unable to capture the pattern in the data points accurately. The model is too simple (Cai, 2014). This is essentially what underfitting entails. As discussed, underfitting implies that not all the essential data points are considered in calculating a prediction.

As a reminder, overfitting occurs when a model fits the data too well, and includes the noise contained in data when calculating a prediction (Cai, 2014). Overfitting leads to a low training error, creating the illusion that the model functions perfectly during training, but ends up being grossly inaccurate when fitted on new data (Koehrsen, 2018).

In Figure 7, these concepts are illustrated to compare the two extremes.

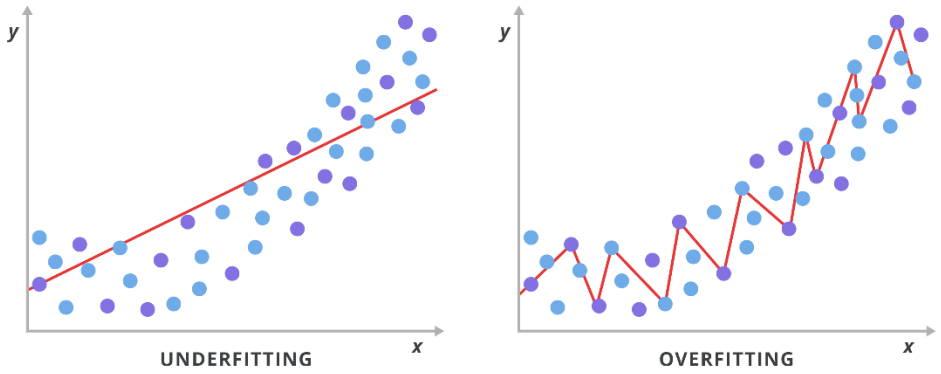


Figure 7: Underfitting vs overfitting.

One of the ways to prevent overfitting is to add more training samples so that the algorithm doesn't learn the noise in the system and can become more generalized. This increases the model's prediction accuracy (Agarwal, 2018). Another tool to prevent overfitting is cross-validation, whereby the initial training data set is split to create multiple sets to train the model. Cross-validation is discussed in Unit 2.

Explore further:

Cross-validation and adding more training samples are not the only ways to overcome overfitting. Read about other ways to prevent overfitting.

3.2.2 Advantages and disadvantages of polynomial regression

Consider the advantages and disadvantages of polynomial regression before fitting this type of model onto a data set.

The advantages of polynomial regression include the following:

- **Broad range of subfunctions:** A broad range of functions can be fitted under polynomial regression in R.
- **Wide range of application:** Polynomial regression models fit a range of curves.

(Pant, 2019)

The disadvantages relate to the effect of outliers, and include the following:

- **Sensitive to outliers:** One or two outliers can have major effects on the output.

- **Difficult to detect outliers:** When compared to linear regression, the polynomial regression model has fewer validation tools to detect outliers.

(Pant, 2019)

Pause and reflect:

Now that you understand the basics of the polynomial regression model, what problems have you encountered that would be suitable to be fitted with this type of model as opposed to a linear regression model?

3.2.3 Polynomial regression in practice

To provide an example of polynomial regression, an artificial data set of two numerical variables was created, *x* and *f*. The data points of this artificial data set are visualised in Figure 8 with the *x* variable on the x-axis, and the *f* variable on the y-axis.

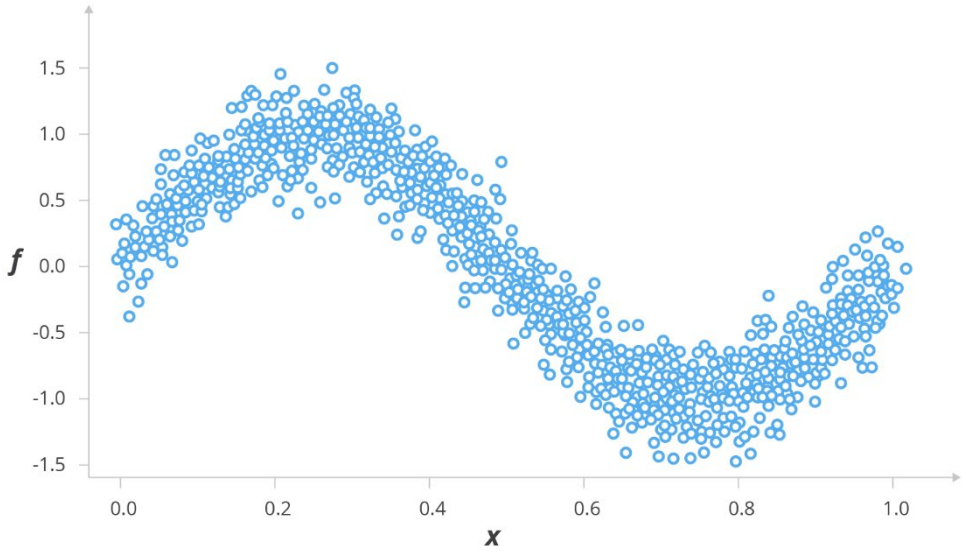
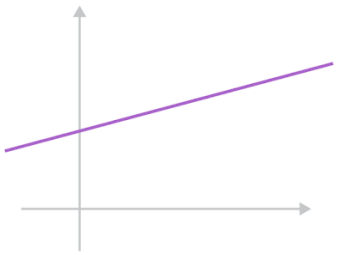


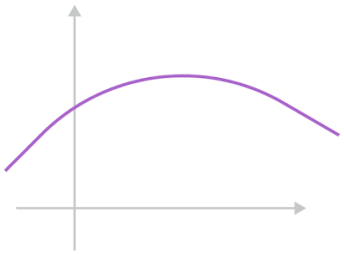
Figure 8: Scatterplot of all the data points in an artificial data set.

The purpose of a polynomial curve is to determine the most accurate positioning of the line to be drawn through the observations. The line's positioning forms the basis for the predictions about unknown values. For example, the polynomial must draw a line that connects the observations in Figure 8 and illustrate the underlying pattern in the data. Depending on the observations, different orders of the polynomials can be used to determine the most appropriate line to draw. These different orders, or degrees, of the polynomials, are referred to as polynomials of the n^{th} degree (Longstreet, n.d.).

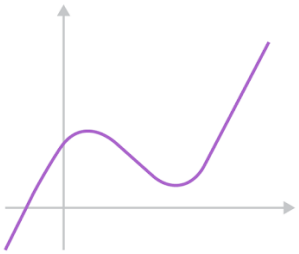
As the order of the polynomial increases, the more complex it becomes. In Figure 9, the first three orders of the polynomials are shown to depict the increase in complexity.



First-degree polynomials are represented as straight lines with no curves.



Second-degree polynomials form lines with only one curve.



Third-degree polynomials consist of multiple curves.

Figure 9: Degrees of polynomials.

As the order, or degree, increases, more terms are added to the power of the order. As mentioned, a first-order polynomial regression – a straight line – is estimated using regression techniques and yields the result shown in Figure 10. The red line shows the polynomial regression line. The blue dots represent the observed points.

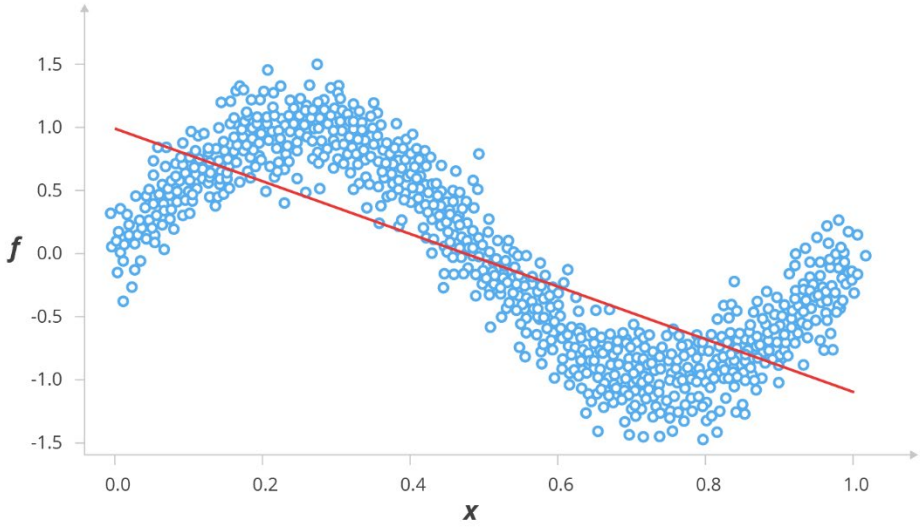


Figure 10: Example of a first-order polynomial regression.

Figure 10 illustrates how the linear regression line is not the best fit for the given data set. To improve the fit of the regression line, the root mean square error (RMSE) value may be used to measure the fit of a model. The RMSE value measures the size of the variation that cannot be explained by the model. In this model, the RMSE value is 0.22. The R^2 statistic is also calculated, and stands at 91.18%, meaning that this model is only able to explain 91.18% of the total variation in the variable f. If the model is increased to a third-order polynomial, such that the estimated line has more curves, and estimated using regression techniques, the results shown in Figure 11 are yielded.

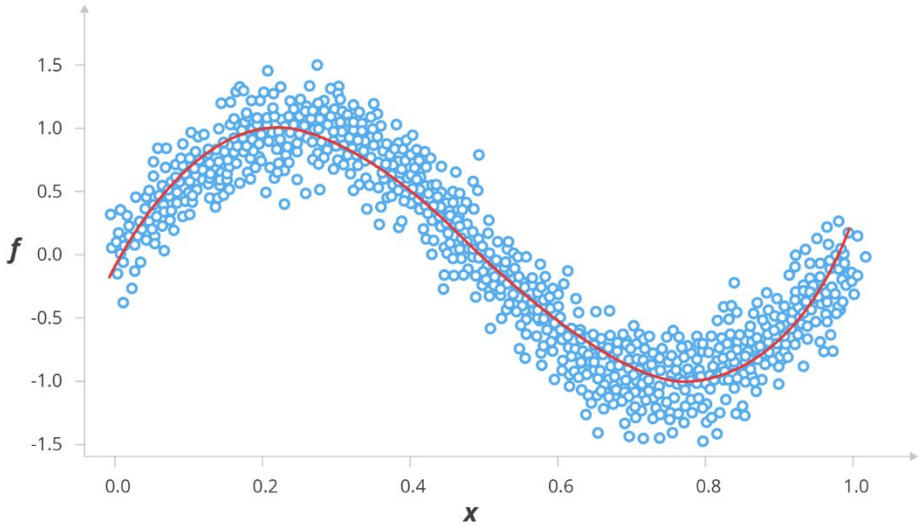


Figure 11: Example of a third-order polynomial regression.

It is clear that this third-order polynomial regression leads to lower variance between the predicted values and the observed values. The lower variance implies a better fit for the data set. This is confirmed by both the lower RMSE, which is now less than 0.21, and

higher R^2 statistic, which is now 92.22%. Finally, a 20th-order polynomial regression is fitted using the regression technique, which yields the results shown in Figure 12.

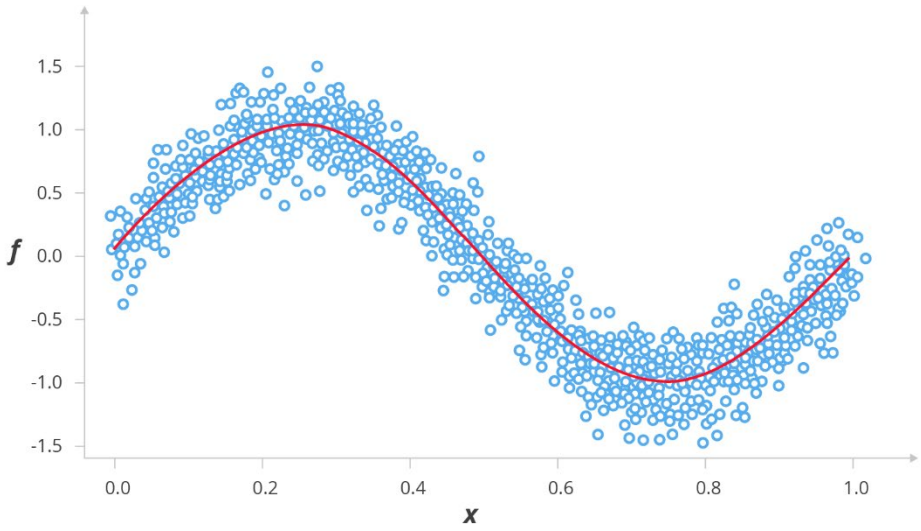


Figure 12: Example of 20th-order polynomial regression.

In Figure 12, the order of the polynomial is increased, leading to a slightly better model fit on the training data.

Pause and reflect:

Given the increase in computational power needed with each increase in the order of the polynomial, do you think this slight decrease in RMSE and slight increase in the R^2 value is worth going through this process?

Note that although the model might fit the training data very well, it might not perform well when new data is introduced, due to overfitting. This leads to an increase in the variance, or error, when fitted on unseen data.

At this point in the lesson, you have the opportunity to engage with a practice quiz to test your understanding of the content. Access this lesson on the Online Campus to engage with this quiz.

4. Fitting a model

When training a model, the cost function and training error are used to determine the model's accuracy. The testing error is then used to determine the accuracy of the model on unseen data after it has been trained.

4.1 Cost function

Recall that the cost function is used to determine the accuracy of the model's output. Ultimately, the cost function describes the error rate between the predicted values and the expected values, expressed in a number form. The cost function can either be minimised or maximised, depending on the goal of the model. If the cost function is minimised, the goal is to determine the model parameters that ensures the lowest possible error.

For example, if the model is used to determine whether a customer falls into a specific customer segment based on age, job title, and income, the error rate should ideally be as low as possible, as the wrong classification may be detrimental to business decisions. In contrast, the cost function, which can also be referred to as reward function, should be maximised if the output it generates is labelled as the reward. In this case, the goal is to find the model parameters that generate the highest possible reward (Krzyk, 2018).

A famous example is in games such as AlphaGo. AlphaGo is a computer program that plays the board game Go. It is the first computer program that defeated a professional human Go player, as well as the world champion Go player (DeepMind, n.d.). If the probability of winning in each step is treated as a reward function, the goal of the model is to maximise this function by taking certain actions.

4.2 Training error

When a model is trained, it searches for patterns in the training data (Bhattacharjee, 2017). The model is therefore taught how to process input data into appropriate outputs. The training error is measured by a metric that compares the prediction outputted by the model to the observations in the training data. Depending on the nature of the tasks, different types of test error metrics should be chosen. For instance, in classification, often the misclassification error is calculated, while in regression, the mean square error (MSE), or the root mean square error (RMSE) is most commonly used. The test error also serves as a tool for model selection. If two possible models can be fitted onto the same training data set, the test error can assist in selecting the most accurate model to solve a given problem (Schönleber, 2018).

Since the training data is used to train the model, the training error should be lower than the test error because the same data with which the model was trained is used for error calculation (Wu & Vos, 2018:166).

4.3 Test error

After a model has been trained, it can be tested to determine its accuracy. The test data set is a sample of data used to evaluate the final model fit on the training data set (Brownlee, 2017). This test set should be different from, or independent of, the training data. The test error refers to the inaccuracies of the model in generating an output from the test data set.

At this point in the lesson, you have the opportunity to engage with a practice quiz to test your understanding of the content. Access this lesson on the Online Campus to engage with this quiz.

4.4 Fitting models in R

The goal of machine learning is to fit a model onto a data set in such a way that it is neither overfitted nor underfitted in order to yield the most accurate predictions from the input data. In Video 1, Dr Yining Chen demonstrates this process with specific reference to the way that the KNN classifier and polynomial regression models are fitted onto data sets.

Video 1: Fitting KNN and polynomial regression onto data. (Access this lesson on the Online Campus to engage with this video.)

Explore further:

Find out how you can standardize your data in R. The article discusses three types of scaling and the steps to implementing these techniques when fitting models in R.

5. Conclusion

In this lesson, the definition of a machine learning model was explored as well as the way a model is fitted onto a data set to generate predictions. The overall discussion aimed to provide an in-depth understanding of the theory by exploring two practical examples of the

KNN classifier and polynomial regression. Both these models can be extremely useful methods to inform business decisions.

The KNN classifier can be used to solve both regression and classification problems. For example, it can successfully classify customers into segments to direct marketing strategies. In some cases, linear regression models are not the best fit for a given data set, and that is why polynomial regression is essential to avoid underfitting. When you are presented with a business problem where the relationship between variables is non-linear, it is important to understand polynomial regression to fit this model onto the data set.

You've now learnt more about how a model is trained to ensure its accuracy when fitted onto new data. Having gained a fundamental understanding of the cost function, training error, and test error in the training process, engage with your peers in the next component to interrogate the relationship between these terms and how they can improve a model.

6. Bibliography

Adi Bronshtein. 2017. *A quick introduction to k-nearest neighbors algorithm* [Blog, 11 April]. Available: https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7 [2020, January 8].

Agarwal, A. 2018. *Polynomial regression*. Available: https://towardsdatascience.com/polynomial-regression-bbe8b9d97491 [2019, November 22].

Bhattacharjee, J. 2017. *Some key machine learning definitions*. Available: https://medium.com/technology-nineleaps/some-key-machine-learning-definitions-b524eb6cb48 [2019, November 21].

Brownlee, J. 2016. *Parametric and nonparametric machine learning algorithms*. Available: https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/ [2020, January 28].

Brownlee, J. 2017. *What is the difference between test and validation datasets?* Available: https://machinelearningmastery.com/difference-test-validation-datasets [2019, November 25].

Cai, E. 2014. *Machine learning lesson of the day – overfitting and underfitting*. Available: https://chemicalstatistician.wordpress.com/2014/03/19/machine-learning-lesson-of-the-day-overfitting-and-underfitting [2019, November 22].

DeepMind. n.d. *AlphaGo*. Available: https://deepmind.com/research/case-studies/alphago-the-story-so-far [2020, January 27].

Harrison, O. 2018. *Machine learning basics with the k-nearest neighbors algorithm*. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 [2019, November 21].

James, G., Witten, D., Hastie, T. & Tibshirani, R. 2013. *Introduction to statistical learning: with applications in R*. New York: Springer-Verlag.

Koehrsen, W. 2018. *Overfitting vs. underfitting: a complete example*. Available: https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765 [2019, November 22].

Krzyk, K. 2018. *Coding deep learning for beginners — linear regression (part 2): cost function*. Available: https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f [2019, November 25].

LeCun, Y., Cortes, C. & Burges, C.J.C. n.d. *The MNIST database of handwritten digits.* Available: http://yann.lecun.com/exdb/mnist/ [2019, November 25].

Lin, T. 2018. *Day 3 — K-nearest neighbors and bias–variance tradeoff*. Available: https://medium.com/30-days-of-machine-learning/day-3-k-nearest-neighbors-and-bias-variance-tradeoff-75f84d515bdb [2020, January 28].

Longstreet, D. n.d. *Polynomials (for the non data scientist)*. Available: https://www.davidlongstreet.com/polynomials.html [2019, December 3].

Miller, L. 2018. *Machine learning week 1: cost function, gradient descent and univariate linear regression*. Available: https://medium.com/@lachlanmiller_52885/machine-learning-week-1-cost-function-gradient-descent-and-univariate-linear-regression-8f5fe69815fd [2019, November 22].

Narkhede, S. 2018. *Understanding confusion matrix*. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 [2019, December 2].

Pant, A. 2019. *Introduction to linear regression and polynomial regression*. Available: https://towardsdatascience.com/introduction-to-linear-regression-and-polynomial-regression-f8adc96f31cb [2019, November 22].

Schönleber, D. 2018. *A "short" introduction to model selection*. Available: https://towardsdatascience.com/a-short-introduction-to-model-selection-bb1bb9c73376 [2019, November 25].

Song, C. 2018. *Introduction to k-nearest neighbors with red wines quality in R*. Available: https://medium.com/nyu-a3sr-data-science-team/k-nn-with-red-wines-quality-in-r-bd55dcba4fd7 [2019, November 27].

Subramanian, D. 2019. *A simple introduction to k-nearest neighbors algorithm*. Available: https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e [2020, January 8].

Tavish Srivastava. 2018. *Introduction to k-nearest neighbors: a powerful machine learning algorithm (with implementation in Python & R)* [Blog, 26 March]. Available: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering [2020, January 8].

Vos, P. & Wu, Q. 2018. Inference and prediction methods. In *Computational analysis and understanding of natural languages: principles, methods and applications*. Volume 38. V.N. Gudivada & C.R. Rao, Eds. North Holland: Elsevier.