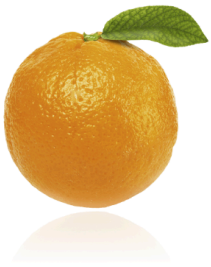
What is a feature (supervised learning primer)?

In this book, we will frequently refer to concepts from supervised learning. This section is a brief introduction to those concepts that you may safely skip if you already know them.

Machine learning is concerned with making accurate predictions. *Features* are measurable properties of entities that we can use to make predictions. For example, if we want to predict if a piece of fruit is an apple or an orange (apple or orange is the fruit's *label*), we could use the fruit's color as a feature to help us predict the correct class of fruit, see figure 1. This is a classification problem: given examples of fruit along with their color and label, we want to classify a fruit as either an apple or orange using the color feature. As we are only considering 2 classes of fruit, we can call this a *binary classification problem*.

feature color: green - RGB(0, 128, 0)
feature weight: 70-250gr



feature color: orange - RGB(255,165,0)
feature weight: 60-300gr

Figure 1. A feature is a measurable property of an entity that has predictive power for the machine learning task. Here, the fruit's color has predictive power of whether the fruit is an apple or an orange.

The fruit's color is a good feature to distinguish apples from oranges, because oranges do not tend to be green and apples do not tend to be orange in color. Weight, in contrast, is not a good feature as it is not predictive of whether the fruit is an apple or an orange. "Roundness" of the fruit could be a good feature, but it is not easy to measure - a feature should be a measurable property.

A *supervised learning* algorithm *trains* a *machine learning model* (often abbreviated to just 'model'), using lots of examples of apples and oranges along with the color of each apple and orange, to predict the label "Apple" or "Orange" for new pieces of fruit using only the new fruit's color. However, color is a single value but rather measured as 3 separate values, one value for each of the red, green, and blue (RGB) channels. As our apples and oranges typically have '0' for the blue channel, we can ignore the blue channel, leaving us with two features: the red and green channel values. In figure 2, we can see some examples of apples (green circles) and oranges (orange crosses), with the red channel value plotted on the x-axis and the green channel value plotted on the y-axis. We can see that an almost straight line can separate most of the apples from the oranges. This line is called the *decision boundary* and we can compute it with a linear model that minimizes the distance between the straight line and all of the circles

and crosses plotted in the diagram. The decision boundary that we learnt from the data is most commonly called the (trained) model.

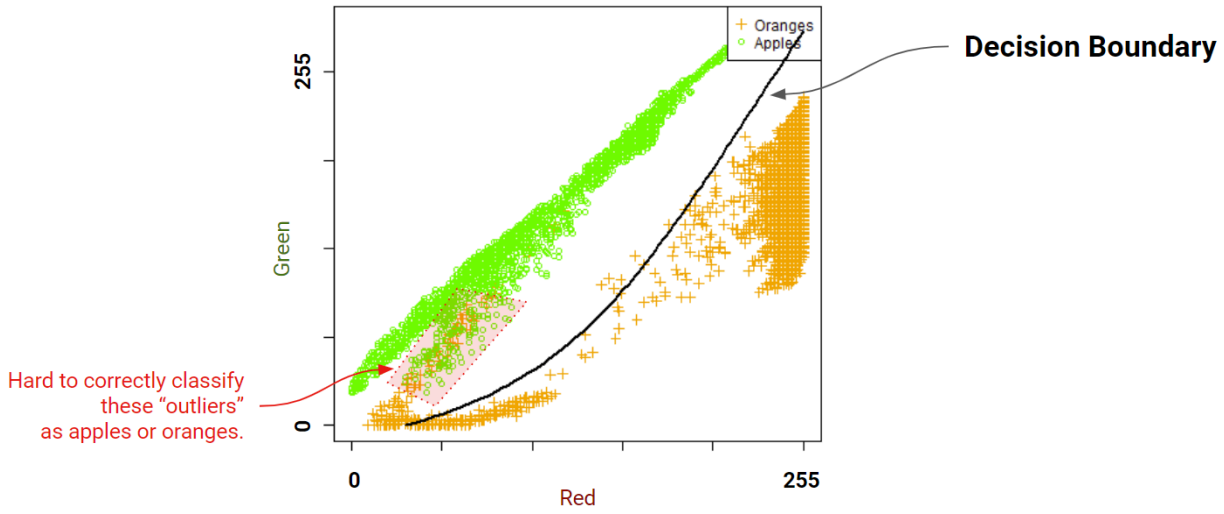


Figure 2. When we plot all of our example apples and oranges using the observed values for the red and green color channels, we can see that most apples are on the left of the decision boundary, and most oranges are on the right. Some apples and oranges are, however, difficult to differentiate based only on their red and green channel colors.

The model can then be used to classify a new piece of a fruit as either an apple or orange using its red and green channel values. If the fruit's red and green channel values place it on the left of the line, then it is an apple, otherwise it is an orange.

In figure 2, you can also see there are a small number of oranges that are not correctly classified by the decision boundary. Our model could wrongly predict that an orange is an apple. However, if the model predicts the fruit is an orange, it will be correct - the fruit will be an orange. We can say that the model's *precision* is 100% for oranges, but is less than 100% for apples.

Another way to look at the model's performance is to consider if the model predicts it is an apple, and it is an apple - it will not be wrong. However, the model will not always predict the fruit is an orange if the fruit is an orange. That is, the model's *recall* is 100% for apples. But if the model predicts an orange, its recall is less than 100%. In machine learning, we often combine precision and recall in a single value called the *F1 Score*, that can be used as one measure of the model's performance. The F1 score is the harmonic mean of precision and recall, and a value of 1.0 indicates perfect precision and recall for the model. Precision, recall, and F1 scores are model performance measures for classification problems.

Let's complicate this simple model. What if we add red apples into the mix? Now, we want our model to classify whether the fruit is an apple or orange - but we will have both red and green apples, see figure 3.

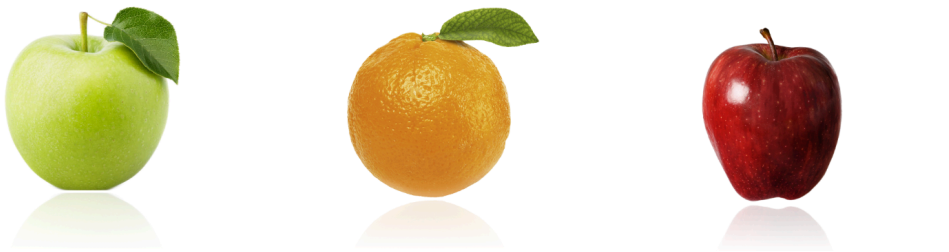


Figure 3. The red apple complicates our prediction problem because there is no longer a linear decision boundary between the apples and oranges using only color as a feature.

We can see that red apples also have zero for the blue channel, so we can ignore that feature. However, in figure 4, we can see that the red examples are located in the bottom right hand corner of our chart, and our model (a linear decision boundary) is broken - it would predict that red apples are oranges. Our model's precision and recall is now much worse.

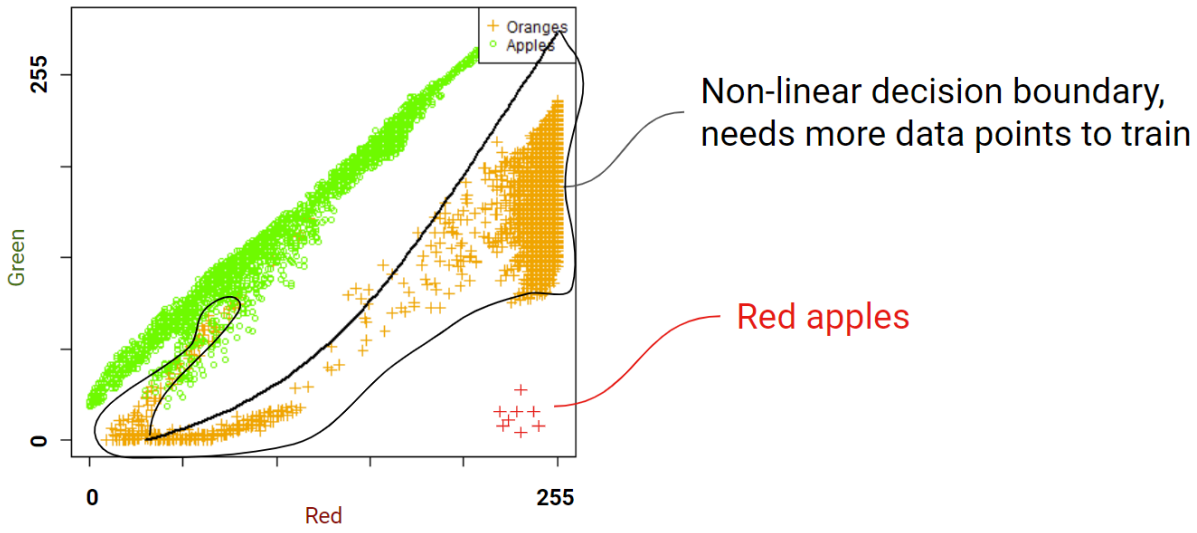


Figure 4. When we add red apples to our training examples, we can see that we can no longer use a straight line to classify fruit as orange or apple. We now need a non-linear decision boundary to separate apples from oranges, and in order to learn the decision boundary, we need a more complex model (with more parameters), more training examples, and m.

Our fruit classifier used examples of features and labels (apples or oranges) to train a model (as a decision boundary). However, machine learning is not just used for classification problems. It is also used to predict numerical values - *regression problems*. An example of a regression problem would be to estimate the weight of an apple. For the regression problem of predicting the weight of an apple, two useful features could be its diameter, and its green color channel value - dark green apples are heavier than light green and red apples. The apple's weight is called the target variable (we typically use the term label for classification problems and target in regression problems).

For this regression problem, a supervised learning model could be trained using examples of apples along with their green color channel value, diameter, and weight. For new apples (not seen during training), our model, see figure 5, can predict the fruit's weight using its type, red channel value, green channel value, and diameter.

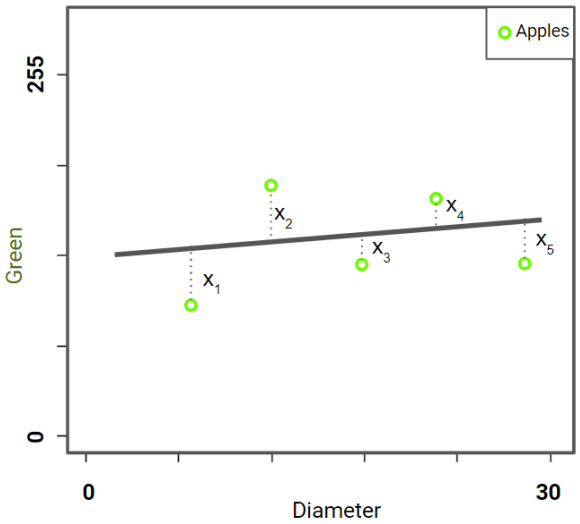


Figure 5. This regression problem of predicting the weight of an apple can be solved using a linear model that minimizes the mean-squared error.

In this regression example, we don't technically need the full power of supervised learning yet - a simple linear model will work well. We can *fit* a straight line (that predicts an apple's weight using its green channel value and diameter) to the data points by drawing the line on the chart such that it minimizes the distance between the line and the data points (X_1, X_2, X_3, X_4, X_5). For example, a common technique is to sum together the distance between all the data points and the line in the *mean absolute error* (MAE). We take the absolute value of the distance of the data points to the line, because if we didn't take the absolute value then the distance for X_1 would be negative and the distance for X_2 would be positive, canceling each other out. Sometimes, we have data points that are very far from the line, and we want the model to have a larger error for those outliers - we want the model to perform better for outliers. For this, we can sum the square of distances and then take the square root of the total. This is called the *root mean-squared error* (RMSE). The MAE and RMSE are both metrics used to help fit our linear regression model, but also to evaluate the performance of our regression model. Similar to our earlier classification example, if we introduce more features to improve the performance of our regression model, we will have to upgrade from our linear regression model to use a supervised learning regression model that can perform better by learning non-linear relationships between the features and the target.

Now that we have introduced supervised learning to solve classification and regression problems, we can claim that supervised learning is concerned with extracting a pattern from data (features and labels/targets) to a model, where the model's value is that it can used to perform *inference* (make predictions) on new (unlabeled) data points (using only feature values). If the model performs well on the new data points (that were not seen during training),

we say the model has good *generalization* performance. We will later see that we always hold back some example data points during model training (a *test set* of examples that we don't train the model on), so that we can evaluate the model's performance on unseen data.

Now we have introduced the core concepts in supervised learning[1], let's look at where the data used to train our models comes from as well as the data that the model will make predictions with.

The source code for the supervised training of our fruit classifier is available on the book's github repository in chapter one. If you are new to machine learning it is a good exercise to run and understand this code.

[1] The source code for the supervised training of our fruit classifier is available on the book's github repository in chapter one. If you are new to machine learning it is a good exercise to run and understand this code.