Probability concepts explained: Introduction

Definitions and Notation

Probability is often associated with at least one event. This event can be anything. Toy examples of events include *rolling a die* or *pulling a coloured ball out of a bag.* In these examples the outcome of the event is random (you can’t be sure of the value that the die will show when you roll it), so the variable that represents the outcome of these events is called a **random variable** (often abbreviated to RV).

We are often interested in knowing the probability of a random variable taking on a certain value. For example, what is the probability that when I roll a fair 6-sided die it lands on a 3? The word “fair” is important here because it tells us that the probability of the die landing on any of the six faces; 1, 2, 3, 4, 5 and 6 is equal. Now intuitively, you might tell me that the answer is 1/6. Correct! But how do we write this mathematically? Well firstly, we need to understand that the random variable here is the outcome of the event related to rolling the die. Typically, random variables are denoted by capital letters, here, we will denote it with X. Therefore, we want to know what the probability is that X = 3. But as mathematicians are lazy when it comes to writing things down, the shorthand for asking “what is the probability?” is to use the letter P. Therefore we can write “what is the probability that when I roll a fair 6-sided die it lands on a 3?” mathematically as “P(X=3)”

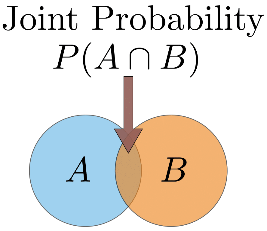
The 3 types of probability

Above introduced the concept of a random variable and some notation on probability. However, probability can get quite complicated. Perhaps the first thing to understand is that there are different types of probability. It can either be *marginal*, *joint* or *conditional*.

***Marginal Probability:*** If A is an event, then the marginal probability is the probability of that event occurring, P(A). *Example: Assuming that we have a pack of traditional playing cards, an example of a marginal probability would be the probability that a card drawn from a pack is red: P(red) = 0.5.*

***Joint Probability:*** The probability of the intersection of two or more events. Visually it is the intersection of the circles of two events on a Venn Diagram (see figure below). If A and B are two events then the joint probability of the two events is written as P(A **∩** B). *Example: the probability that a card drawn from a pack is red and has the value 4 is P(red and 4) = 2/52 = 1/26. (There are 52 cards in a pack of traditional playing cards and the 2 red ones are the hearts and diamonds).* We’ll go through this example in more detail later.

***Conditional Probability:*** The conditional probability is the probability that some event(s) occur given that we know other events have already occurred. If A and B are two events then the conditional probability of A occurring given that B has occurred is written as P(A|B). *Example: the probability that a card is a four given that we have drawn a red card is P(4|red) = 2/26 = 1/13. (There are 52 cards in the pack, 26 are red and 26 are black. Now because we’ve already picked a red card, we know that there are only 26 cards to choose from, hence why the first denominator is 26).*



Venn diagram showing the ‘space’ of outcomes of 2 events A and B. In the diagram the 2 events overlap. This overlap represents the joint probability, i.e. the probability of both event A and event B happening. If there was no overlap between the events then the joint probability would be zero.

Linking the probability types: The general multiplication rule

The general multiplication rule is a beautiful equation that links all 3 types of probability:

https://miro.medium.com/max/461/1*T97Dasjs2Cl132KH0tpcBg.png

Further explanation of the examples

Sometimes distinguishing between the joint probability and the conditional probability can be quite confusing, so using the example of picking a card from a pack of playing cards let’s try to hammer home the difference.

In the case where we want to find the probability of picking a card that is red and a 4 i.e. the joint probability *P(red and 4)* I want you to imagine having all 52 cards face down and picking one at random. Of those 52 cards, 2 of them are red and 4 (4 of diamonds and 4 of hearts). So the joint probability is therefore 2/52 = 1/26

In the case where we want to find the probability of picking a card that is 4 given that I know the card is already red i.e. the conditional probability, *P(4|red),* I want you to again imagine having all 52 cards. However, before picking a card at random you sort through the cards and select all of the 26 red ones. Now you put those 26 cards face down and pick a card randomly. Again, 2 of those red cards are 4’s so the conditional probability is 2/26 = 1/13

Alternatively, if you prefer the maths, we can use the general multiplication rule that we defined above to calculate the joint probability. We first rearrange to make the joint probability, P(A **∩** B), the subject of the equation (in other words, lets put P(A **∩** B) on the left hand side of the equals sign and put everything else on the right). After rearranging we get P(A **∩** B) = P(A|B) ✕ P(B). Let A be the event that the card is a 4 and B is the event that the card is red. P(A|B) = 1/13 as we said above and P(B) = 1/2 (half of the cards are red). Therefore P(A **∩** B) = 1/13 ✕ 1/2 = 1/26.

Probability rules: ‘and’ and ‘or’

‘and’ rule

We’ve already seen the ‘and’ scenario disguised as joint probability, however we don’t yet know how to calculate the probability in the ‘and’ scenario. So let’s go through an example. Let’s suppose we have two events: event A — tossing a fair coin, and event B — rolling a fair die. We might be interested in knowing the probability of rolling a 6 and the coin landing on heads. So to calculate the joint probability of rolling a 6 and the coin landing heads we can rearrange the general multiplication rule above to get P(A **∩** B) = P(A|B) ✕ P(B)**.** We know that event A is tossing a coin and B is rolling a die. So P(A|B) term asks “*what is the probability of the coin landing on heads given that I’ve rolled a 6 on the die?”* This is where we intuitively understand that the outcome of tossing the coin doesn’t depend on the roll of the die. The events are said to be **independent.** In this scenario the result of the coin toss would be the same no matter what we rolled on the die. Mathematically we express this as P(A|B) = P(A). Therefore when the **events are independent, the joint probability is just the product of the individual marginal probabilities of the events: P(A ∩ B) = P(A) ✕ P(B).** So P(coin landing heads and rolling a 6) = P(A=heads, B=6) = 1/2 ✕ 1/6 = 1/12.

Notice that I wrote P(A=heads, B=6). The comma between the events is shorthand for joint probability (you will see this written in the literature).

It should be noted that in many real world scenarios events are assumed to be independent even when this is not the case in reality. This is mainly because it makes the maths **a lot** easier. The bonus is that the results are often very useful. The [Naive Bayes’](https://en.wikipedia.org/wiki/Naive_Bayes_classifier) method is possibly the most common example of this in data science and typically gives fairly good results in [text classification](https://medium.com/@theflyingmantis/text-classification-in-nlp-naive-bayes-a606bf419f8c) problems.

‘or’ rule

With the ‘and’ rule we had to multiply the individual probabilities. When we’re in the ‘or’ scenario we have to **add the individual probabilities and subtract the intersection.** Mathematically we write this as P(A **∪** B) = P(A) + P(B) - P(A **∩** B). Why do we have to do this you ask? Well it goes back to the Venn diagram in the above figure. If we add the circle for A and the circle for B then it means that we’re adding the intersection twice. Therefore we need to subtract the intersection.

So let’s change our example above to find the probability of rolling a 6 or the coin landing on heads. This is P(coin landing heads or rolling a 6) = P(A=heads **∪** B=6) = 1/2 + 1/6 - 1/12 = 6/12 + 2/12 - 1/12 = 7/12

Note that the **∪** symbol is known as ‘union’ and is used in the ‘or’ scenario.

There are occasions when we don’t have to subtract the intersection. This happens when the two circles in the Venn diagram don’t overlap. When the circles for two events do not overlap we say that these events are **mutually exclusive.** This implies that the intersection is zero, written mathematically as P(A ∩ B) = 0. Let’s do an example that covers this case. Suppose we roll a die and we want to know the probability of rolling a 5 or a 6. These events are mutually exclusive because I can’t roll a 5 and a 6. Therefore, their circles in a Venn diagram do not overlap. So the probability of rolling a 5 or a 6 is equal to 1/6 + 1/6 = 2/6 = 1/3 (we haven’t subtracted anything).

Wrap up

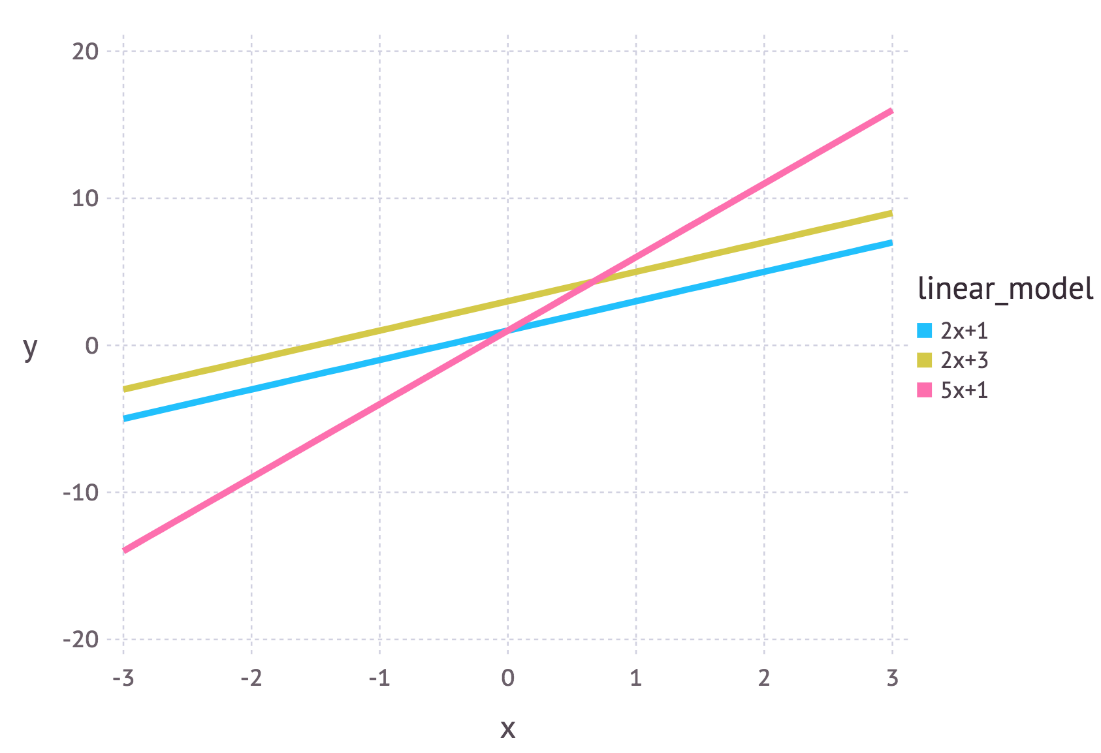
Thank you for making it this far. If anything, I hope my rambling has been accessible to you even if you have learned nothing new. If there is anything that is unclear or I’ve made some mistakes in the above feel free to leave a comment. In future posts in this series I’ll go through some more advanced concepts. The [next post will explain maximum likelihood](https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1) and work through an example.

Probability concepts explained: Maximum likelihood estimation

What are parameters?

Often in machine learning we use a model to describe the process that results in the data that are observed. For example, we may use a random forest model to classify whether customers may cancel a subscription from a service (known as [churn modelling](https://towardsdatascience.com/machine-learning-predicting-customer-churn-dd38a42774cf)) or we may use a linear model to predict the revenue that will be generated for a company depending on how much they may spend on advertising (this would be an example of [linear regression](https://towardsdatascience.com/simple-and-multiple-linear-regression-in-python-c928425168f9)). Each model contains its own set of parameters that ultimately defines what the model looks like.

For a linear model we can write this as*y = mx + c*. In this example *x*could represent the advertising spend and *y*might be the revenue generated. *m* and *c* are parameters for this model. Different values for these parameters will give different lines (see figure below).



Three linear models with different parameter values.

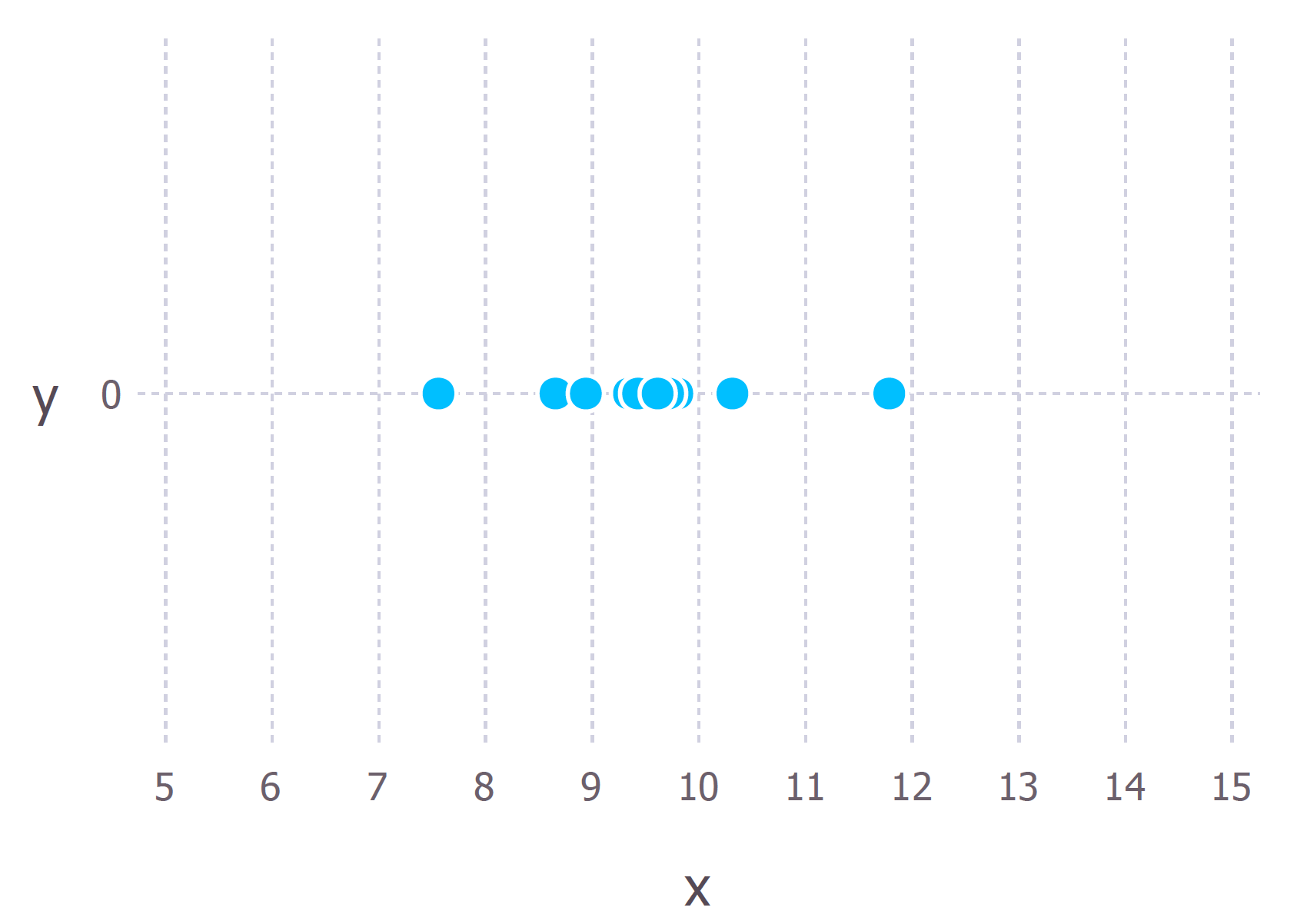
So parameters define a blueprint for the model. It is only when specific values are chosen for the parameters that we get an instantiation for the model that describes a given phenomenon.

Intuitive explanation of maximum likelihood estimation

Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are found such that they maximise the likelihood that the process described by the model produced the data that were actually observed.

The above definition may still sound a little cryptic so let’s go through an example to help understand this.

Let’s suppose we have observed 10 data points from some process. For example, each data point could represent the length of time in seconds that it takes a student to answer a specific exam question. These 10 data points are shown in the figure below

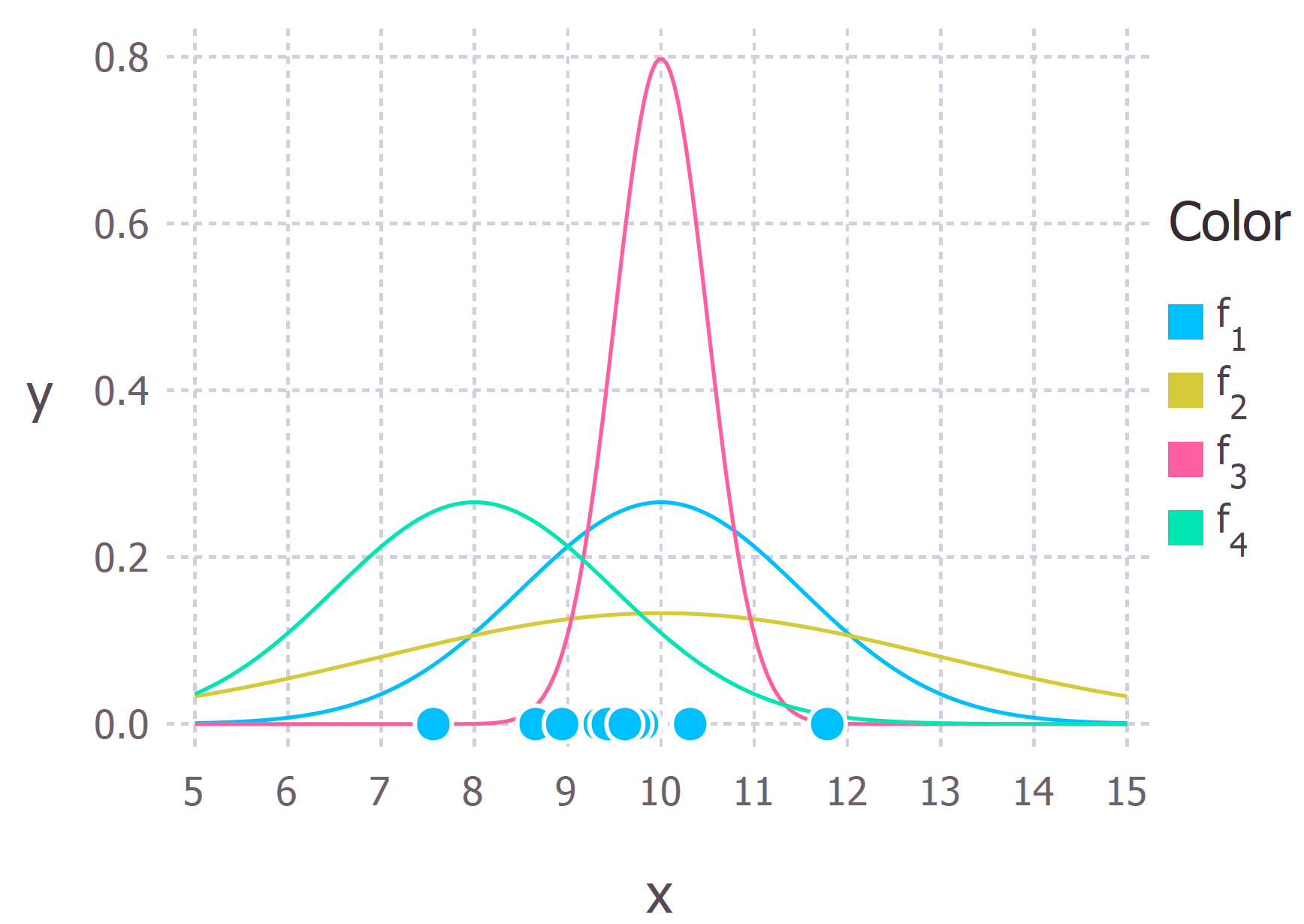


The 10 (hypothetical) data points that we have observed

We first have to decide which model we think best describes the process of generating the data. This part is very important. At the very least, we should have a good idea about which model to use. This usually comes from having some domain expertise but we wont discuss this here.

For these data we’ll assume that the data generation process can be adequately described by a Gaussian (normal) distribution. Visual inspection of the figure above suggests that a Gaussian distribution is plausible because most of the 10 points are clustered in the middle with few points scattered to the left and the right. (Making this sort of decision on the fly with only 10 data points is ill-advised but given that I generated these data points we’ll go with it).

Recall that the Gaussian distribution has 2 parameters. The mean, μ, and the standard deviation, σ. Different values of these parameters result in different curves (just like with the straight lines above). We want to know*which curve was most likely responsible for creating the data points that we observed?*(See figure below). Maximum likelihood estimation is a method that will find the values of μ and σ that result in the curve that best fits the data.



The 10 data points and possible Gaussian distributions from which the data were drawn. f1 is normally distributed with mean 10 and variance 2.25 (variance is equal to the square of the standard deviation), this is also denoted f1 ∼ N (10, 2.25). f2 ∼ N (10, 9), f3 ∼ N (10, 0.25) and f4 ∼ N (8, 2.25). The goal of maximum likelihood is to find the parameter values that give the distribution that maximise the probability of observing the data.

The true distribution from which the data were generated was f1 ~ N(10, 2.25), which is the blue curve in the figure above.

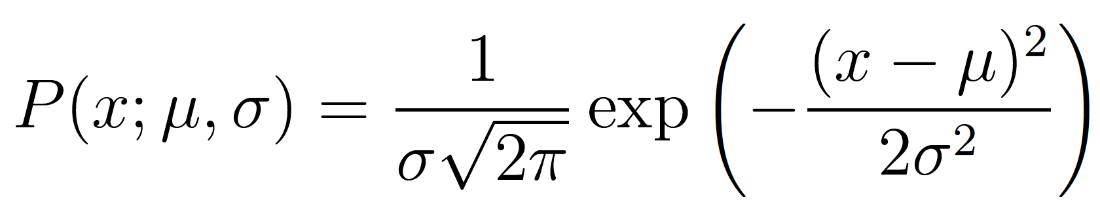
Calculating the Maximum Likelihood Estimates

Now that we have an intuitive understanding of what maximum likelihood estimation is we can move on to learning how to calculate the parameter values. The values that we find are called the maximum likelihood estimates (MLE).

Again we’ll demonstrate this with an example. Suppose we have three data points this time and we assume that they have been generated from a process that is adequately described by a Gaussian distribution. These points are 9, 9.5 and 11. *How do we calculate the maximum likelihood estimates of the parameter values of the Gaussian distribution μ and σ?*

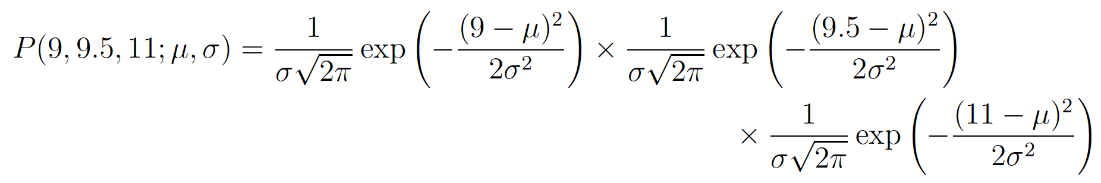
What we want to calculate is the total probability of observing all of the data, i.e. the joint probability distribution of all observed data points. To do this we would need to calculate some conditional probabilities, which can get very difficult. So it is here that we’ll make our first assumption. *The assumption is that each data point is generated independently of the others*. This assumption makes the maths much easier. If the events (i.e. the process that generates the data) are independent, then the total probability of observing all of data is the product of observing each data point individually (i.e. the product of the marginal probabilities).

The probability density of observing a single data point *x,*that is generated from a Gaussian distribution is given by:



The semi colon used in the notation *P(x; μ, σ)*is there to emphasise that the symbols that appear after it are parameters of the probability distribution. So it shouldn’t be confused with a conditional probability (which is typically represented with a vertical line e.g. *P(A| B)).*

In our example the total (joint) probability density of observing the three data points is given by:

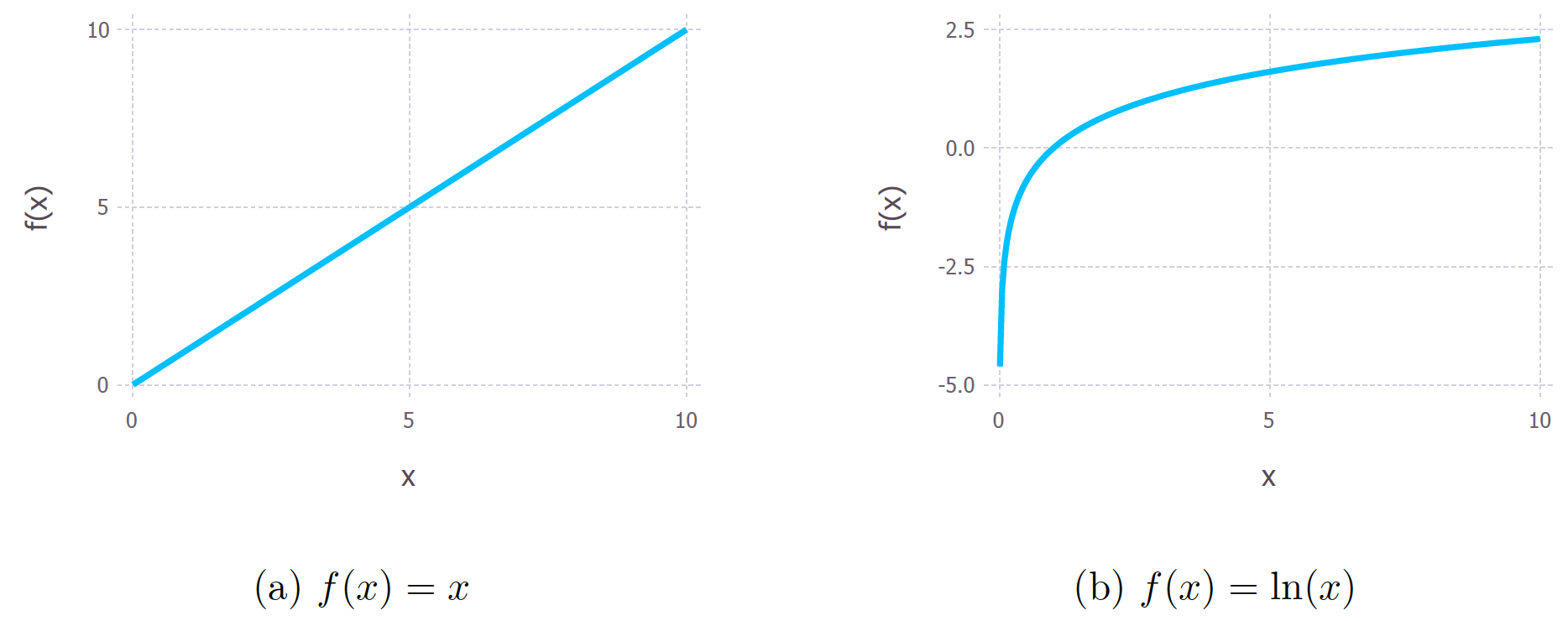


We just have to figure out the values of *μ*and*σ*that results in giving the maximum value of the above expression.

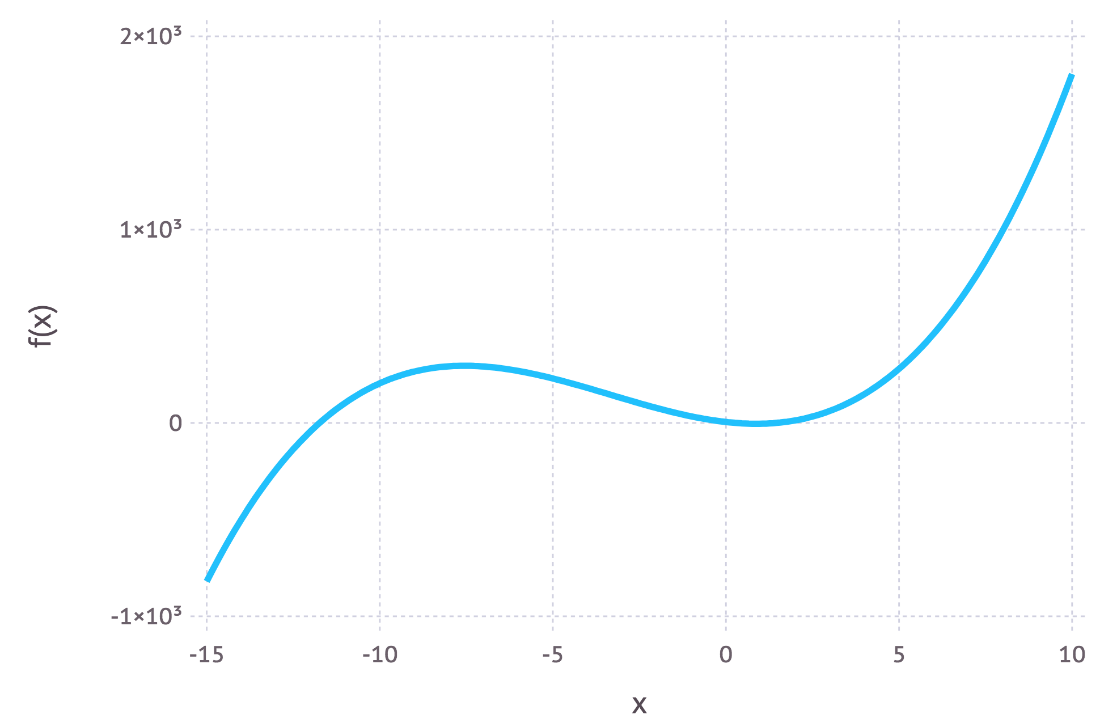
If you’ve covered calculus in your maths classes then you’ll probably be aware that there is a technique that can help us find maxima (and minima) of functions. It’s called *differentiation.* All we have to do is find the derivative of the function, set the derivative function to zero and then rearrange the equation to make the parameter of interest the subject of the equation. And voilà, we’ll have our MLE values for our parameters. I’ll go through these steps now but I’ll assume that the reader knows how to perform differentiation on common functions. If you would like a more detailed explanation then just let me know in the comments.

The log likelihood

The above expression for the total probability is actually quite a pain to differentiate, so it is almost always simplified by taking the natural logarithm of the expression. This is absolutely fine because the natural logarithm is a [monotonically increasing function](https://en.wikipedia.org/wiki/Monotonic_function). This means that if the value on the x-axis increases, the value on the y-axis also increases (see figure below). This is important because it ensures that the maximum value of the log of the probability occurs at the same point as the original probability function. Therefore we can work with the simpler log-likelihood instead of the original likelihood.

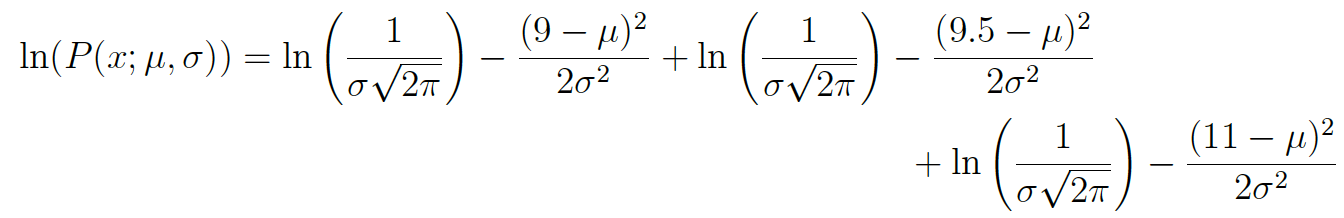


Monotonic behaviour of the original function, y = x on the left and the (natural) logarithm function y = ln(x). These functions are both monotonic because as you go from left to right on the x-axis the y value always increases.

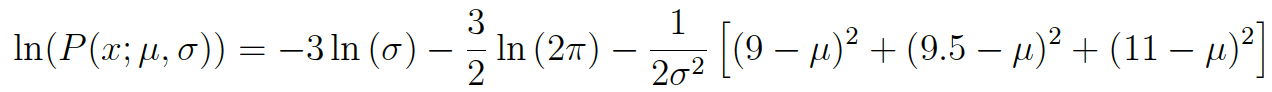


Example of a non-monotonic function because as you go from left to right on the graph the value of f(x) goes up, then goes down and then goes back up again.

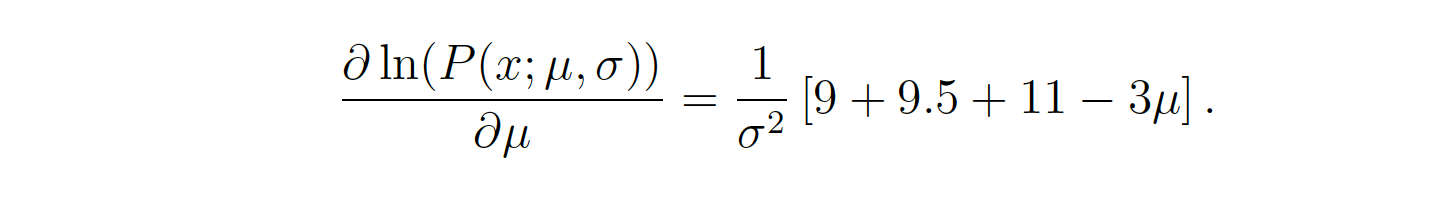
Taking logs of the original expression gives us:



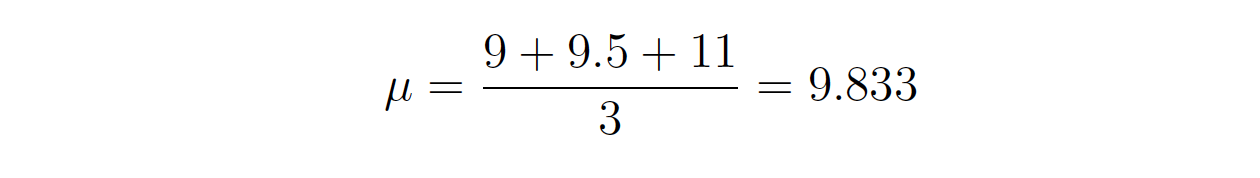
This expression can be simplified again using the laws of logarithms to obtain:



This expression can be differentiated to find the maximum. In this example we’ll find the MLE of the mean, μ. To do this we take the partial derivative of the function with respect to μ, giving



Finally, setting the left hand side of the equation to zero and then rearranging for μ gives:



And there we have our maximum likelihood estimate for μ. We can do the same thing with σ too but I’ll leave that as an exercise for the keen reader.

Concluding remarks

Can maximum likelihood estimation always be solved in an exact manner?

*No* is the short answer. It’s more likely that in a real world scenario the derivative of the log-likelihood function is still analytically intractable (i.e. it’s way too hard/impossible to differentiate the function by hand). Therefore, iterative methods like [Expectation-Maximization algorithms](https://en.wikipedia.org/wiki/Expectation–maximization_algorithm) are used to find numerical solutions for the parameter estimates. The overall idea is still the same though.

So why maximum likelihood and not maximum probability?

Well this is just statisticians being pedantic (but for good reason). Most people tend to use probability and likelihood interchangeably but statisticians and probability theorists distinguish between the two. The reason for the confusion is best highlighted by looking at the equation.

https://miro.medium.com/max/1406/1*XUHA8X_WauSB8anrb6lllA.png

These expressions are equal! So what does this mean? Let’s first define *P(data; μ, σ)*? It means *“the probability density of observing the data with model parameters μ and σ”*. It’s worth noting that we can generalise this to any number of parameters and any distribution.

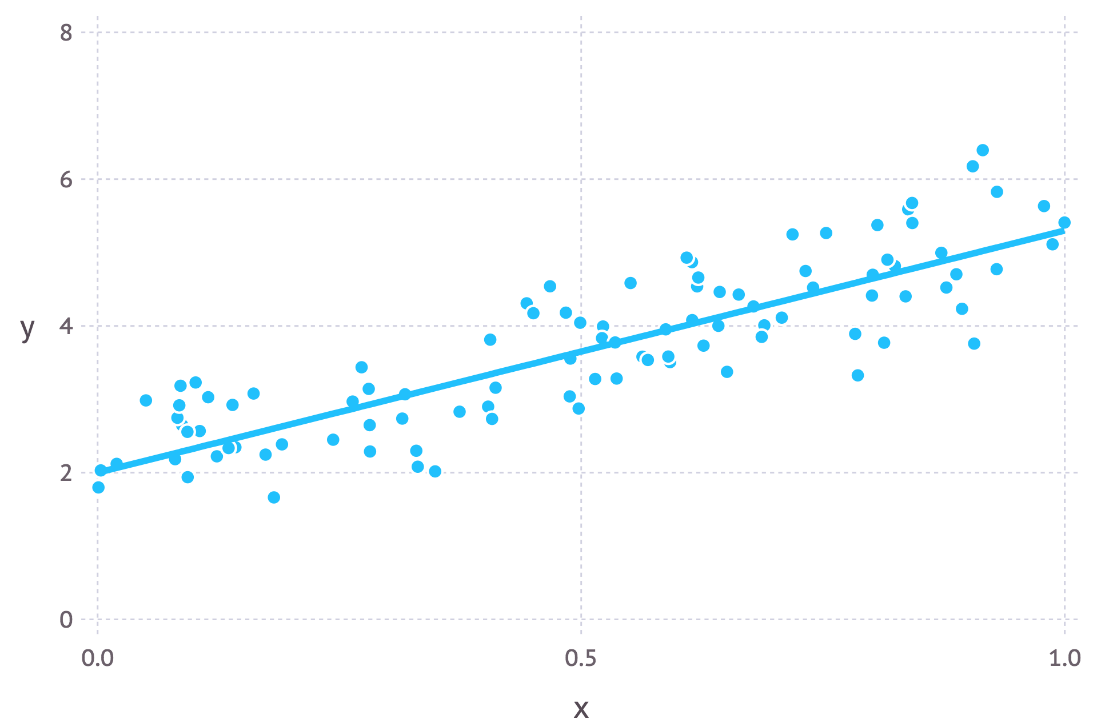
On the other hand *L(μ, σ; data)* means *“the likelihood of the parameters μ and σ taking certain values given that we’ve observed a bunch of data.”*

The equation above says that the probability density of the data given the parameters is equal to the likelihood of the parameters given the data. But despite these two things being equal, the likelihood and the probability density are fundamentally asking different questions — one is asking about the data and the other is asking about the parameter values. This is why the method is called maximum likelihood and not maximum probability.

When is least squares minimisation the same as maximum likelihood estimation?

Least squares minimisation is another common method for estimating parameter values for a model in machine learning. It turns out that when the model is assumed to be Gaussian as in the examples above, the MLE estimates are equivalent to the least squares method. For a more in-depth mathematical derivation check out [these slides](https://web.archive.org/web/20111202153913/http:/www.cs.cmu.edu/~epxing/Class/10701/recitation/recitation3.pdf).

Intuitively we can interpret the connection between the two methods by understanding their objectives. For least squares parameter estimation we want to find the line that minimises the total squared distance between the data points and the regression line (see the figure below). In maximum likelihood estimation we want to maximise the total probability of the data. When a Gaussian distribution is assumed, the maximum probability is found when the data points get closer to the mean value. Since the Gaussian distribution is symmetric, this is equivalent to minimising the distance between the data points and the mean value.



Regression line showing data points with random Gaussian noise

If there is anything that is unclear or I’ve made some mistakes in the above feel free to leave a comment. In the next post I plan to cover [Bayesian inference and how it can be used for parameter estimation](https://towardsdatascience.com/probability-concepts-explained-bayesian-inference-for-parameter-estimation-90e8930e5348).

Maximum Likelihood

Maximum likelihood, also called the maximum likelihood method, is the procedure of finding the value of one or more parameters for a given statistic which makes the *known* [likelihood](http://mathworld.wolfram.com/Likelihood.html) distribution a [maximum](http://mathworld.wolfram.com/Maximum.html). The maximum likelihood estimate for a parameter mu is denoted mu^^.

For a [Bernoulli distribution](http://mathworld.wolfram.com/BernoulliDistribution.html),

|  |  |
| --- | --- |
| d/(dtheta)[(N; Np)theta^(Np)(1-theta)^(Nq)]=Np(1-theta)-thetaNq=0, | (1) |

so maximum likelihood occurs for theta=p. If p is not known ahead of time, the [likelihood function](http://mathworld.wolfram.com/LikelihoodFunction.html) is

|  |  |  |  |
| --- | --- | --- | --- |
| f(x_1,...,x_n|p) | = | P(X_1=x_1,...,X_n=x_n|p) | (2) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline8.gif | = | p^(x_1)(1-p)^(1-x_1)...p^(x_n)(1-p)^(1-x_n) | (3) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline11.gif | = | p^(sumx_i)(1-p)^(sum(1-x_i))=p^(sumx_i)(1-p)^(n-sumx_i), | (4) |

where x=0 or 1, and i=1, ..., n.

|  |  |
| --- | --- |
| lnf=sumx_ilnp+(n-sumx_i)ln(1-p) | (5) |
| (d(lnf))/(dp)=(sumx_i)/p-(n-sumx_i)/(1-p)=0. | (6) |

Rearranging gives

|  |  |
| --- | --- |
| sumx_i-psumx_i=np-psumx_i, | (7) |

so

|  |  |
| --- | --- |
| p^^=(sumx_i)/n. | (8) |

For a [normal distribution](http://mathworld.wolfram.com/NormalDistribution.html),

|  |  |  |  |
| --- | --- | --- | --- |
| f(x_1,...,x_n|mu,sigma) | = | product1/(sigmasqrt(2pi))e^(-(x_i-mu)^2/(2sigma^2)) | (9) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline20.gif | = | ((2pi)^(-n/2))/(sigma^n)exp[-(sum(x_i-mu)^2)/(2sigma^2)] | (10) |

so

|  |  |
| --- | --- |
| lnf=-1/2nln(2pi)-nlnsigma-(sum(x_i-mu)^2)/(2sigma^2) | (11) |

and

|  |  |
| --- | --- |
| (partial(lnf))/(partialmu)=(sum(x_i-mu))/(sigma^2)=0, | (12) |

giving

|  |  |
| --- | --- |
| mu^^=(sumx_i)/n. | (13) |

Similarly,

|  |  |
| --- | --- |
| (partial(lnf))/(partialsigma)=-n/sigma+(sum(x_i-mu)^2)/(sigma^3)=0 | (14) |

gives

|  |  |
| --- | --- |
| sigma^^=sqrt((sum(x_i-mu^^)^2)/n). | (15) |

Note that in this case, the maximum likelihood [standard deviation](http://mathworld.wolfram.com/StandardDeviation.html) is the sample [standard deviation](http://mathworld.wolfram.com/StandardDeviation.html), which is a [biased estimator](http://mathworld.wolfram.com/BiasedEstimator.html) for the population [standard deviation](http://mathworld.wolfram.com/StandardDeviation.html).

For a weighted [normal distribution](http://mathworld.wolfram.com/NormalDistribution.html),

|  |  |
| --- | --- |
| f(x_1,...,x_n|mu,sigma)=product1/(sigma_isqrt(2pi))e^(-(x_i-mu)^2/2sigma_i^2) | (16) |
| lnf=-1/2nln(2pi)-nsumlnsigma_i-sum((x_i-mu)^2)/(2sigma_i^2) | (17) |

|  |  |
| --- | --- |
| (partial(lnf))/(partialmu)=sum((x_i-mu))/(sigma_i^2)=sum(x_i)/(sigma_i^2)-musum1/(sigma_i^2)=0 | (18) |

gives

|  |  |
| --- | --- |
| mu^^=(sum(x_i)/(sigma_i^2))/(sum1/(sigma_i^2)). | (19) |

The [variance](http://mathworld.wolfram.com/Variance.html) of the [mean](http://mathworld.wolfram.com/Mean.html) is then

|  |  |
| --- | --- |
| sigma_mu^2=sumsigma_i^2((partialmu)/(partialx_i))^2. | (20) |

But

|  |  |
| --- | --- |
| (partialmu)/(partialx_i)=partial/(partialx_i)(sum(x_i/sigma_i^2))/(sum(1/sigma_i^2))=(1/sigma_i^2)/(sum(1/sigma_i^2)), | (21) |

so

|  |  |  |  |
| --- | --- | --- | --- |
| sigma_mu^2 | = | sumsigma_i^2((1/sigma_i^2)/(sum(1/sigma_i^2)))^2 | (22) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline26.gif | = | sum(1/sigma_i^2)/([sum(1/sigma_i^2)]^2) | (23) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline29.gif | = | 1/(sum(1/sigma_i^2)). | (24) |

For a [Poisson distribution](http://mathworld.wolfram.com/PoissonDistribution.html),

|  |  |
| --- | --- |
| f(x_1,...,x_n|lambda)=(e^(-lambda)lambda^(x_1))/(x_1!)...(e^(-lambda)lambda^(x_n))/(x_n!)=(e^(-nlambda)lambda^(sumx_i))/(x_1!...x_n!) | (25) |
| lnf=-nlambda+(lnlambda)sumx_i-ln(productx_i!) | (26) |

|  |  |
| --- | --- |
| (d(lnf))/lambda=-n+(sumx_i)/lambda=0 | (27) |
| lambda^^=(sumx_i)/n. | |

Maximum Likelihood Estimation

[Printer-friendly versionPrinter-friendly version](https://online.stat.psu.edu/stat414/print/book/export/html/191/)

Statement of the Problem

Suppose we have a random sample X1, X2,..., Xn whose assumed probability distribution depends on some unknown parameter θ. Our primary goal here will be to find a point estimator u(X1, X2,..., Xn), such that u(x1, x2,..., xn) is a "good" point estimate of θ, where x1, x2,..., xn are the observed values of the random sample. For example, if we plan to take a random sample X1, X2,..., Xn for which the Xi are assumed to be normally distributed with mean μ and variance σ2, then our goal will be to find a good estimate of μ, say, using the data x1, x2,..., xn that we obtained from our specific random sample.

The Basic Idea

It seems reasonable that a good estimate of the unknown parameter θ would be the value of θ that **maximizes** the probability, errrr... that is, the **likelihood**... of getting the data we observed. (So, do you see from where the name "maximum likelihood" comes?) So, that is, in a nutshell, the idea behind the method of maximum likelihood estimation. But how would we implement the method in practice? Well, suppose we have a random sample X1, X2,..., Xn for which the probability density (or mass) function of each Xi is f(xi; θ). Then, the joint probability mass (or density) function of X1, X2,..., Xn, which we'll (not so arbitrarily) call ***L*(*θ*)** is:

L(θ)=P(X1=x1,X2=x2,…,Xn=xn)=f(x1;θ)⋅f(x2;θ)⋯f(xn;θ)=n∏i=1f(xi;θ)L(θ)=P(X1=x1,X2=x2,…,Xn=xn)=f(x1;θ)⋅f(x2;θ)⋯f(xn;θ)=∏i=1nf(xi;θ)

The first equality is of course just the definition of the joint probability mass function. The second equality comes from that fact that we have a random sample, which implies by definition that the Xiare independent. And, the last equality just uses the shorthand mathematical notation of a product of indexed terms. Now, in light of the basic idea of maximum likelihood estimation, one reasonable way to proceed is to treat the "**likelihood function**" L(θ) as a function of θ, and find the value of θ that maximizes it.

Is this still sounding like too much abstract gibberish? Let's take a look at an example to see if we can make it a bit more concrete.

Example

Suppose we have a random sample X1, X2,..., Xn where:

* Xi = 0 if a randomly selected student does not own a sports car, and
* Xi = 1 if a randomly selected student does own a sports car.

Assuming that the Xi are independent Bernoulli random variables with unknown parameter p, find the maximum likelihood estimator of p, the proportion of students who own a sports car.

**Solution.** If the Xi are independent Bernoulli random variables with unknown parameter p, then the probability mass function of each Xi is:

f(xi;p)=pxi(1−p)1−xif(xi;p)=pxi(1−p)1−xi

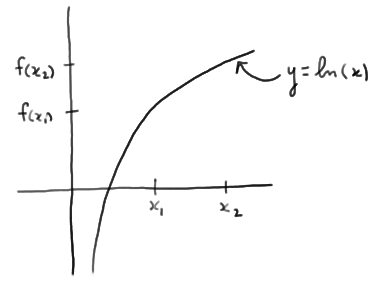
for xi = 0 or 1 and 0 < p < 1. Therefore, the likelihood function L(p) is, by definition:

L(p)=n∏i=1f(xi;p)=px1(1−p)1−x1×px2(1−p)1−x2×⋯×pxn(1−p)1−xnL(p)=∏i=1nf(xi;p)=px1(1−p)1−x1×px2(1−p)1−x2×⋯×pxn(1−p)1−xn

for 0 < p < 1. Simplifying, by summing up the exponents, we get :

L(p)=p∑xi(1−p)n−∑xiL(p)=p∑xi(1−p)n−∑xi

Now, in order to implement the method of maximum likelihood, we need to find the p that maximizes the likelihood L(p). We need to put on our calculus hats now, since in order to maximize the function, we are going to need to differentiate the likelihood function with respect to p. In doing so, we'll use a "trick" that often makes the differentiation a bit easier.  Note that the natural logarithm is an increasing function of x:

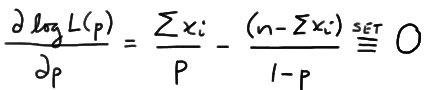


That is, if x1 < x2, then f(x1) < f(x2). That means that the value of p that maximizes the natural logarithm of the likelihood function ln(L(p)) is also the value of p that maximizes the likelihood function L(p). So, the "trick" is to take the derivative of ln(L(p)) (with respect to p) rather than taking the derivative of L(p). Again, doing so often makes the differentiation much easier. (By the way, throughout the remainder of this course, I will use either ln(L(p)) or log(L(p)) to denote the natural logarithm of the likelihood function.)

In this case, the natural logarithm of the likelihood function is:

logL(p)=(∑xi)log(p)+(n−∑xi)log(1−p)logL(p)=(∑xi)log(p)+(n−∑xi)log(1−p)

Now, taking the derivative of the log likelihood, and setting to 0, we get:



Now, multiplying through by p(1−p), we get:

(∑xi)(1−p)−(n−∑xi)p=0(∑xi)(1−p)−(n−∑xi)p=0

Upon distributing, we see that two of the resulting terms cancel each other out:

eqn

leaving us with:

∑xi−np=0∑xi−np=0

Now, all we have to do is solve for p. In doing so, you'll want to make sure that you always put a hat ("^") on the parameter, in this case p, to indicate it is an estimate:

^p=n∑i=1xinp^=∑i=1nxin

or, alternatively, an estimator:

^p=n∑i=1Xinp^=∑i=1nXin

Oh, and we should technically verify that we indeed did obtain a maximum. We can do that by verifying that the second derivative of the log likelihood with respect to p is negative. It is, but you might want to do the work to convince yourself!

Now, with that example behind us, let us take a look at formal definitions of the terms (1) likelihood function, (2) maximum likelihood estimators, and (3) maximum likelihood estimates.

|  |
| --- |
| **Definition.** Let X1, X2,..., Xn be a random sample from a distribution that depends on one or more unknown parameters θ1, θ2,..., θmwith probability density (or mass) function f(xi; θ1, θ2,..., θm). Suppose that (θ1, θ2,..., θm) is restricted to a given parameter space Ω. Then:  (1) When regarded as a function of θ1, θ2,..., θm, the joint probability density (or mass) function of X1, X2,..., Xn:  L(θ1,θ2,…,θm)=n∏i=1f(xi;θ1,θ2,…,θm)L(θ1,θ2,…,θm)=∏i=1nf(xi;θ1,θ2,…,θm)  ((θ1, θ2,..., θm) in Ω) is called the **likelihood function**.  (2) If:  [u1(x1,x2,…,xn),u2(x1,x2,…,xn),…,um(x1,x2,…,xn)][u1(x1,x2,…,xn),u2(x1,x2,…,xn),…,um(x1,x2,…,xn)]  is the m-tuple that maximizes the likelihood function, then:  ^θi=ui(X1,X2,…,Xn)θ^i=ui(X1,X2,…,Xn)  is the **maximum likelihood estimator** of θi, for i = 1, 2, ..., m.  (3) The corresponding observed values of the statistics in (2), namely:  [u1(x1,x2,…,xn),u2(x1,x2,…,xn),…,um(x1,x2,…,xn)][u1(x1,x2,…,xn),u2(x1,x2,…,xn),…,um(x1,x2,…,xn)]  are called the **maximum likelihood estimates** of θi, for i = 1, 2, ..., m. |

**Example**

Suppose the weights of randomly selected American female college students are normally distributed with unknown mean μ and standard deviation σ. A random sample of 10 American female college students yielded the following weights (in pounds):

115   122   130   127   149   160   152   138  149   180

Based on the definitions given above, identify the likelihood function and the maximum likelihood estimator of μ, the mean weight of all American female college students. Using the given sample, find a maximum likelihood estimate of μ as well.

**Solution.** The probability density function of Xi is:

f(xi;μ,σ2)=1σ√2πexp[−(xi−μ)22σ2]f(xi;μ,σ2)=1σ2πexp[−(xi−μ)22σ2]

for −∞ < x < ∞. The parameter space is Ω = {(μ, σ): −∞ < μ < ∞ and 0 < σ < ∞}. Therefore, (you might want to convince yourself that) the likelihood function is:

L(μ,σ)=σ−n(2π)−n/2exp[−12σ2n∑i=1(xi−μ)2]L(μ,σ)=σ−n(2π)−n/2exp[−12σ2∑i=1n(xi−μ)2]

for −∞ < μ < ∞ and 0 < σ < ∞. It can be shown (we'll do so in the next example!), upon maximizing the likelihood function with respect to μ, that the maximum likelihood estimator of μ is:

^μ=1nn∑i=1Xi=¯Xμ^=1n∑i=1nXi=X¯

Based on the given sample, a maximum likelihood estimate of μ is:

^μ=1nn∑i=1xi=110(115+⋯+180)=142.2μ^=1n∑i=1nxi=110(115+⋯+180)=142.2

pounds. Note that the only difference between the formulas for the maximum likelihood estimator and the maximum likelihood estimate is that:

* the estimator is defined using capital letters (to denote that its value is random), and
* the estimate is defined using lowercase letters (to denote that its value is fixed and based on an obtained sample)

Okay, so now we have the formal definitions out of the way. The first example on this page involved a joint probability mass function that depends on only one parameter, namely p, the proportion of successes. Now, let's take a look at an example that involves a joint probability density function that depends on two parameters.

Example

Let X1, X2,..., Xn be a random sample from a normal distribution with unknown mean μ and variance σ2. Find maximum likelihood estimators of mean μ and variance σ2.

**Solution.** In finding the estimators, the first thing we'll do is write the probability density function as a function of θ1 = μ and θ2= σ2:

f(xi;θ1,θ2)=1√θ2√2πexp[−(xi−θ1)22θ2]f(xi;θ1,θ2)=1θ22πexp[−(xi−θ1)22θ2]

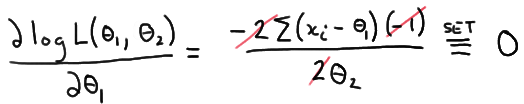
for −∞ < θ1 < ∞ and 0 < θ2 < ∞. We do this so as not to cause confusion when taking the derivative of the likelihood with respect to σ2. Now, that makes the likelihood function:

L(θ1,θ2)=n∏i=1f(xi;θ1,θ2)=θ−n/22(2π)−n/2exp[−12θ2n∑i=1(xi−θ1)2]L(θ1,θ2)=∏i=1nf(xi;θ1,θ2)=θ2−n/2(2π)−n/2exp[−12θ2∑i=1n(xi−θ1)2]

and therefore the log of the likelihood function:

logL(θ1,θ2)=−n2logθ2−n2log(2π)−∑(xi−θ1)22θ2logL(θ1,θ2)=−n2logθ2−n2log(2π)−∑(xi−θ1)22θ2

Now, upon taking the partial derivative of the log likelihood with respect to θ1, and setting to 0, we see that a few things cancel each other out, leaving us with:



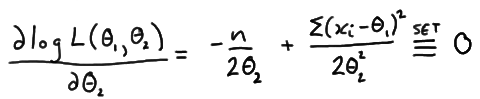
Now, multiplying through by θ2, and distributing the summation, we get:

∑xi−nθ1=0∑xi−nθ1=0

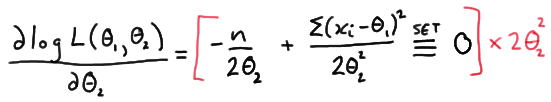
Now, solving for θ1, and putting on its hat, we have shown that the maximum likelihood estimate of θ1is:

^θ1=^μ=∑xin=¯xθ^1=μ^=∑xin=x¯

Now for θ2. Taking the partial derivative of the log likelihood with respect to θ2, and setting to 0, we get:



Multiplying through by 2θ222θ22:



we get:

−nθ2+∑(xi−θ1)2=0−nθ2+∑(xi−θ1)2=0

And, solving for θ2, and putting on its hat, we have shown that the maximum likelihood estimate of θ2is:

^θ2=^σ2=∑(xi−¯x)2nθ^2=σ^2=∑(xi−x¯)2n

(I'll again leave it to you to verify, in each case, that the second partial derivative of the log likelihood is negative, and therefore that we did indeed find maxima.) In summary, we have shown that the maximum likelihood estimators of μ and variance σ2 for the normal model are:

^μ=∑Xin=¯Xμ^=∑Xin=X¯   and   ^σ2=∑(Xi−¯X)2nσ^2=∑(Xi−X¯)2n

respectively.

Note that the maximum likelihood estimator of σ2 for the normal model is not the sample variance S2. They are, in fact, competing estimators. So how do we know which estimator we should use for σ2 ? Well, one way is to choose the estimator that is "unbiased." Let's go learn about unbiased estimators now.

Likelihood

Likelihood is the hypothetical probability that an event that has already occurred would yield a specific outcome. The concept differs from that of a probability in that a probability refers to the occurrence of future events, while a likelihood refers to past events with known outcomes.

A likelihood function L(a) is the probability or probability density for the occurrence of a sample configuration  given that the probability density  with parameter “a” is known,

Likelihood Ratio

A quantity used to test nested hypotheses. Let H' be a nested hypothesis with n' degrees of freedom within H (which has n degrees of freedom), then calculate the maximum likelihood of a given outcome, first given H', then given H. Then

Comparison of -2ln(LR) to the critical value of the chi-squared distribution with n-n' degrees of freedom then gives the significance of the increase in likelihood.

The term likelihood ratio is also used (especially in medicine) to test nonnested complementary hypotheses as follows,

Maximum Likelihood

Maximum likelihood, also called the maximum likelihood method, is the procedure of finding the value of one or more parameters for a given statistic which makes the known [likelihood](http://mathworld.wolfram.com/Likelihood.html) distribution a [maximum](http://mathworld.wolfram.com/Maximum.html). The maximum likelihood estimate for a parameter mu is denotedmu^^.

For a [Bernoulli distribution](http://mathworld.wolfram.com/BernoulliDistribution.html),

|  |  |
| --- | --- |
| d/(dtheta)[(N; Np)theta^(Np)(1-theta)^(Nq)]=Np(1-theta)-thetaNq=0, | (1) |

so maximum likelihood occurs for theta=p. If p is not known ahead of time, the [likelihood function](http://mathworld.wolfram.com/LikelihoodFunction.html) is

|  |  |  |  |
| --- | --- | --- | --- |
| f(x_1,...,x_n|p) | = | P(X_1=x_1,...,X_n=x_n|p) | (2) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline8.gif | = | p^(x_1)(1-p)^(1-x_1)...p^(x_n)(1-p)^(1-x_n) | (3) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline11.gif | = | p^(sumx_i)(1-p)^(sum(1-x_i))=p^(sumx_i)(1-p)^(n-sumx_i), | (4) |

where x=0 or 1, and i=1, ..., n.

|  |  |
| --- | --- |
| lnf=sumx_ilnp+(n-sumx_i)ln(1-p) | (5) |
| (d(lnf))/(dp)=(sumx_i)/p-(n-sumx_i)/(1-p)=0. | (6) |

Rearranging gives

|  |  |
| --- | --- |
| sumx_i-psumx_i=np-psumx_i, | (7) |

so

|  |  |
| --- | --- |
| p^^=(sumx_i)/n. | (8) |

For a [normal distribution](http://mathworld.wolfram.com/NormalDistribution.html),

|  |  |  |  |
| --- | --- | --- | --- |
| f(x_1,...,x_n|mu,sigma) | = | product1/(sigmasqrt(2pi))e^(-(x_i-mu)^2/(2sigma^2)) | (9) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline20.gif | = | ((2pi)^(-n/2))/(sigma^n)exp[-(sum(x_i-mu)^2)/(2sigma^2)] | (10) |

so

|  |  |
| --- | --- |
| lnf=-1/2nln(2pi)-nlnsigma-(sum(x_i-mu)^2)/(2sigma^2) | (11) |

and

|  |  |
| --- | --- |
| (partial(lnf))/(partialmu)=(sum(x_i-mu))/(sigma^2)=0, | (12) |

giving

|  |  |
| --- | --- |
| mu^^=(sumx_i)/n. | (13) |

Similarly,

|  |  |
| --- | --- |
| (partial(lnf))/(partialsigma)=-n/sigma+(sum(x_i-mu)^2)/(sigma^3)=0 | (14) |

gives

|  |  |
| --- | --- |
| sigma^^=sqrt((sum(x_i-mu^^)^2)/n). | (15) |

Note that in this case, the maximum likelihood [standard deviation](http://mathworld.wolfram.com/StandardDeviation.html) is the sample [standard deviation](http://mathworld.wolfram.com/StandardDeviation.html), which is a [biased estimator](http://mathworld.wolfram.com/BiasedEstimator.html) for the population [standard deviation](http://mathworld.wolfram.com/StandardDeviation.html).

For a weighted [normal distribution](http://mathworld.wolfram.com/NormalDistribution.html),

|  |  |
| --- | --- |
| f(x_1,...,x_n|mu,sigma)=product1/(sigma_isqrt(2pi))e^(-(x_i-mu)^2/2sigma_i^2) | (16) |
| lnf=-1/2nln(2pi)-nsumlnsigma_i-sum((x_i-mu)^2)/(2sigma_i^2) | (17) |

|  |  |
| --- | --- |
| (partial(lnf))/(partialmu)=sum((x_i-mu))/(sigma_i^2)=sum(x_i)/(sigma_i^2)-musum1/(sigma_i^2)=0 | (18) |

gives

|  |  |
| --- | --- |
| mu^^=(sum(x_i)/(sigma_i^2))/(sum1/(sigma_i^2)). | (19) |

The [variance](http://mathworld.wolfram.com/Variance.html) of the [mean](http://mathworld.wolfram.com/Mean.html) is then

|  |  |
| --- | --- |
| sigma_mu^2=sumsigma_i^2((partialmu)/(partialx_i))^2. | (20) |

But

|  |  |
| --- | --- |
| (partialmu)/(partialx_i)=partial/(partialx_i)(sum(x_i/sigma_i^2))/(sum(1/sigma_i^2))=(1/sigma_i^2)/(sum(1/sigma_i^2)), | (21) |

so

|  |  |  |  |
| --- | --- | --- | --- |
| sigma_mu^2 | = | sumsigma_i^2((1/sigma_i^2)/(sum(1/sigma_i^2)))^2 | (22) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline26.gif | = | sum(1/sigma_i^2)/([sum(1/sigma_i^2)]^2) | (23) |
| http://mathworld.wolfram.com/images/equations/MaximumLikelihood/Inline29.gif | = | 1/(sum(1/sigma_i^2)). | (24) |

For a [Poisson distribution](http://mathworld.wolfram.com/PoissonDistribution.html),

|  |  |
| --- | --- |
| f(x_1,...,x_n|lambda)=(e^(-lambda)lambda^(x_1))/(x_1!)...(e^(-lambda)lambda^(x_n))/(x_n!)=(e^(-nlambda)lambda^(sumx_i))/(x_1!...x_n!) | (25) |
| lnf=-nlambda+(lnlambda)sumx_i-ln(productx_i!) | (26) |

|  |  |
| --- | --- |
| (d(lnf))/lambda=-n+(sumx_i)/lambda=0 | (27) |
| lambda^^=(sumx_i)/n. | |

Maximum Likelihood Estimator

A maximum likelihood estimator is a value of the parameter a such that the [likelihood function](http://mathworld.wolfram.com/LikelihoodFunction.html) is a maximum

**A Basic Introduction to Separable Convolutions**

The architecture of MobileNet will undoubtedly come across the concept of separable convolutions. But what is that, and how is it different from a normal convolution?

There are two main types of separable convolutions:

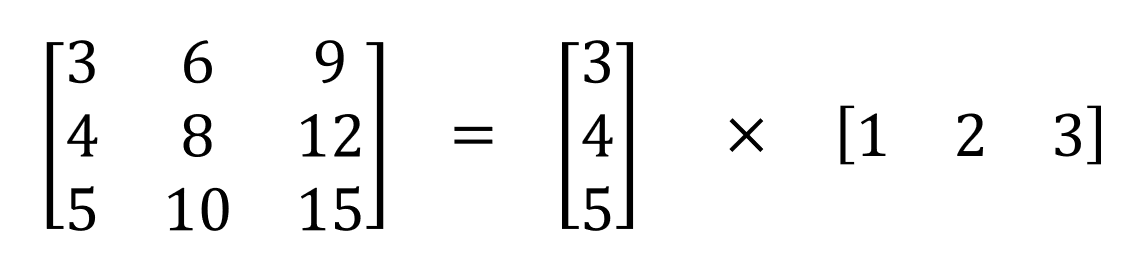
1. Spatial separable convolutions
2. Depthwise separable convolutions.

**Spatial Separable Convolutions**

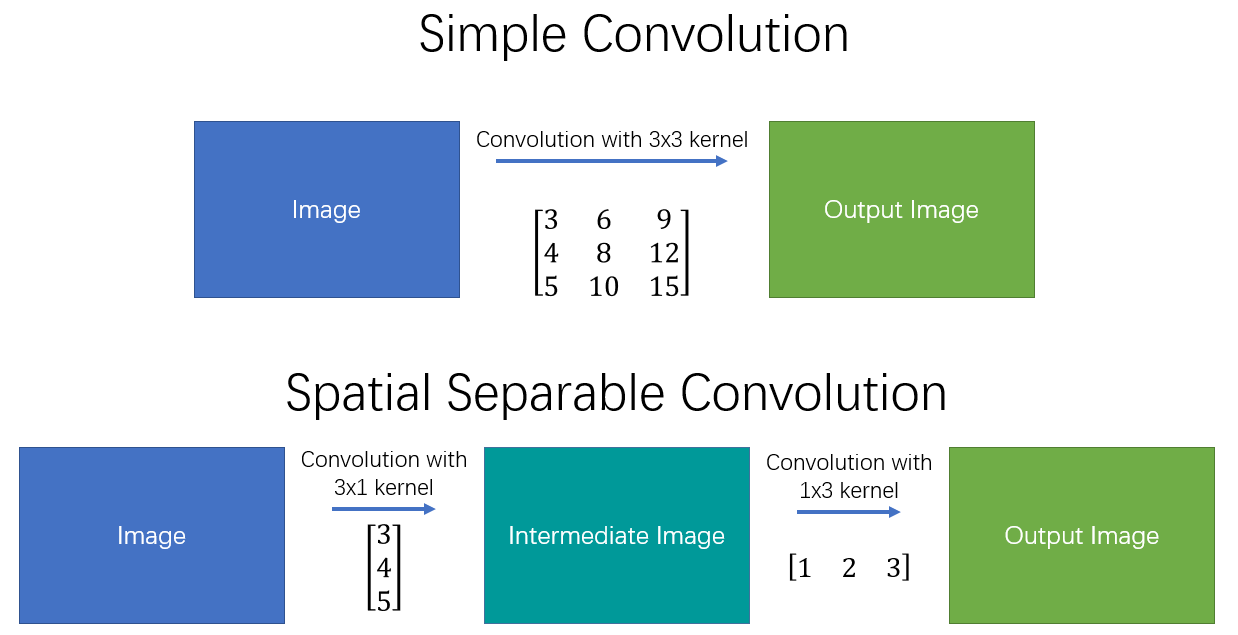
Conceptually, this is the easier one out of the two, and illustrates the idea of separating one convolution into two well. Unfortunately, spatial separable convolutions have some significant limitations, meaning that it is not heavily used in deep learning.

The spatial separable convolution is so named because it deals primarily with the spatial dimensions of an image and kernel: the width and the height. (The other dimension, the “depth” dimension, is the number of channels of each image).

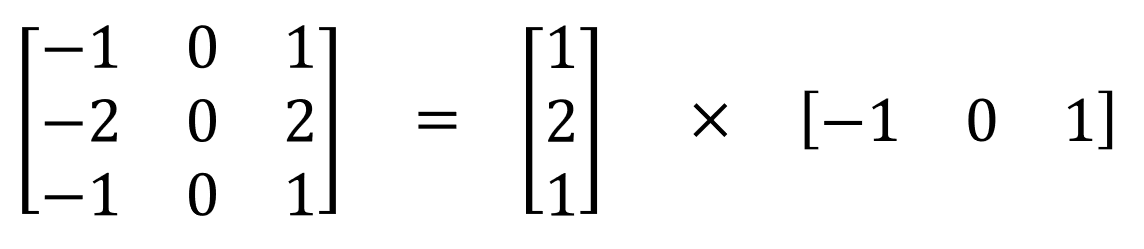
A spatial separable convolution simply divides a kernel into two, smaller kernels. The most common case would be to divide a 3x3 kernel into a 3x1 and 1x3 kernel, like so:



Now, instead of doing one convolution with 9 multiplications, we do two convolutions with 3 multiplications each (6 in total) to achieve the same effect. With less multiplications, computational complexity goes down, and the network is able to run faster.



One of the most famous convolutions that can be separated spatially is the Sobel kernel, used to detect edges:



The main issue with the spatial separable convolution is that not all kernels can be “separated” into two, smaller kernels. This becomes particularly bothersome during training, since of all the possible kernels the network could have adopted, it can only end up using one of the tiny portion that can be separated into two smaller kernels.

**Depthwise Separable Convolutions**

Unlike spatial separable convolutions, depthwise separable convolutions work with kernels that cannot be “factored” into two smaller kernels. Hence, it is more commonly used. *This is the type of separable convolution seen in keras.layers.SeparableConv2D or tf.layers.separable\_conv2d.*

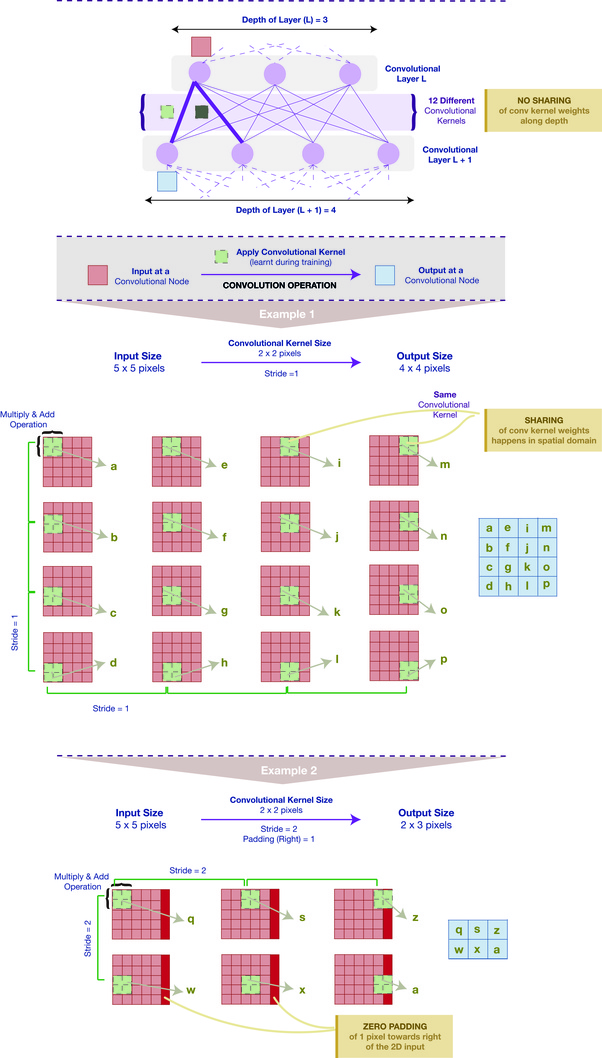
The depthwise separable convolution is so named because it deals not just with the spatial dimensions, but with the depth dimension — the number of channels — as well. An input image may have 3 channels: RGB. After a few convolutions, an image may have multiple channels. You can image each channel as a particular interpretation of that image; in for example, the “red” channel interprets the “redness” of each pixel, the “blue” channel interprets the “blueness” of each pixel, and the “green” channel interprets the “greenness” of each pixel. An image with 64 channels has 64 different interpretations of that image.

Similar to the spatial separable convolution, a depthwise separable convolution splits a kernel into 2 separate kernels that do two convolutions: the depthwise convolution and the pointwise convolution. But first of all, let’s see how a normal convolution works.

**Normal Convolution:**

If you don’t know how a convolution works from a 2-D perspective, read [this article](https://towardsdatascience.com/what-is-a-neural-network-6010edabde2b) or check out [this site](http://setosa.io/ev/image-kernels/).

A typical image, however, is not 2-D; it also has depth as well as width and height. Let us assume that we have an input image of 12x12x3 pixels, an RGB image of size 12x12.

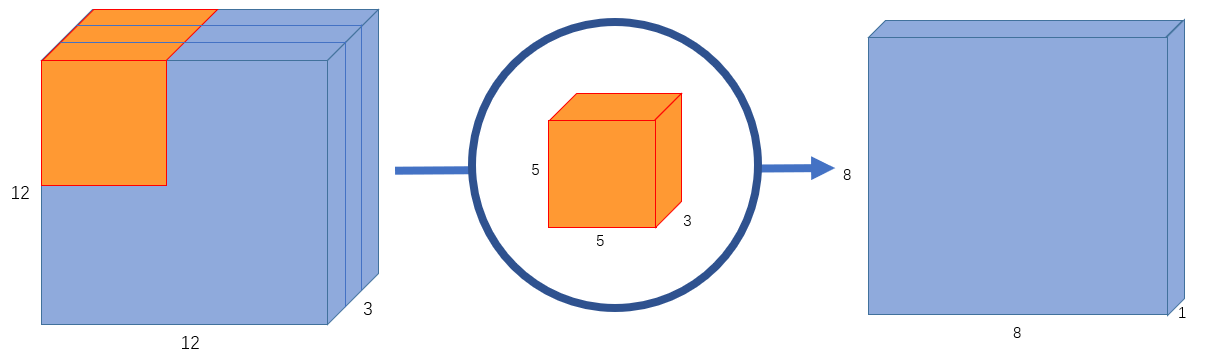


|  |
| --- |
| ***Stride is the number of pixels a convolutional filter moves, like a sliding window, after passing on the weighted average value of all the pixels it just covered.***  ***The old weighted average value becomes one pixel in the featuremap in the next layer. The next weighted average comes from a new group of pixels and it forms the next pixel in the featuremap in the next layer.*** |

Let’s do a 5x5 convolution on the image with no padding and a stride of 1. If we only consider the width and height of the image, the convolution process is kind of like this: 12x12 — (5x5) — >8x8. The 5x5 kernel undergoes scalar multiplication with every 25 pixels, giving out1 number every time. We end up with a 8x8 pixel image, since there is no padding (12–5+1 = 8).

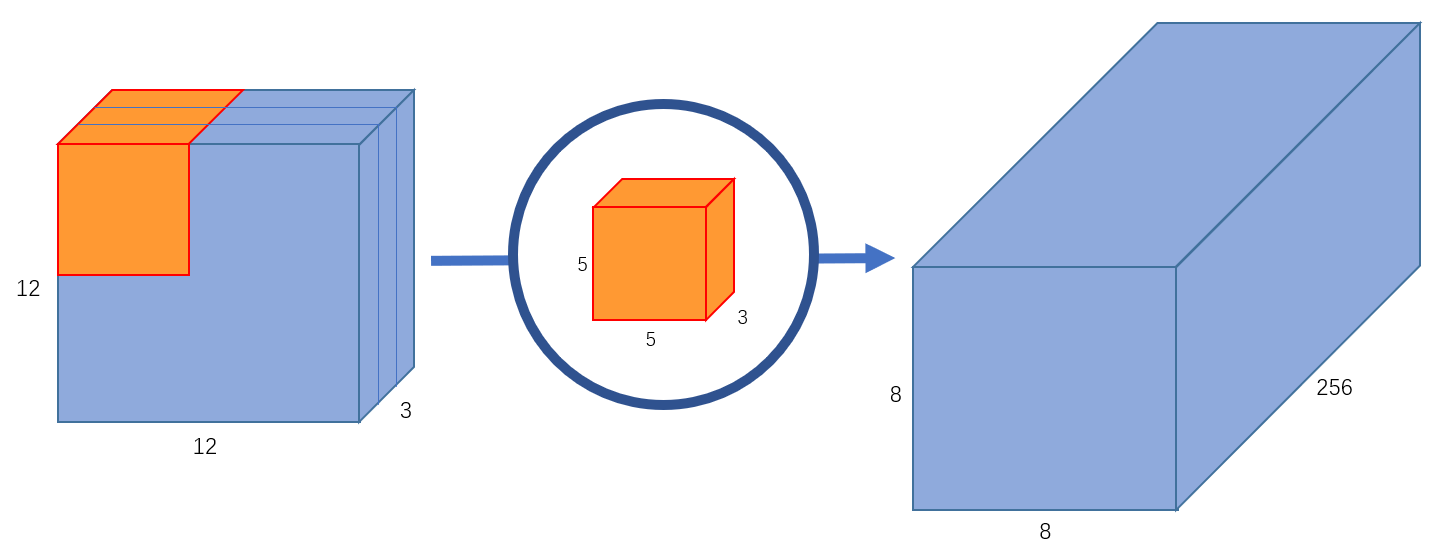
However, because the image has 3 channels, our convolutional kernel needs to have 3 channels as well. This means, instead of doing 5x5=25 multiplications, we actually do 5x5x3=75 multiplications every time the kernel moves.

Just like the 2-D interpretation, we do scalar matrix multiplication on every 25 pixels, outputting 1 number. After going through a 5x5x3 kernel, the 12x12x3 image will become a 8x8x1 image.



What if we want to increase the number of channels in our output image? What if we want an output of size 8x8x256?

Well, we can create 256 kernels to create 256 8x8x1 images, then stack them up together to create a 8x8x256 image output.

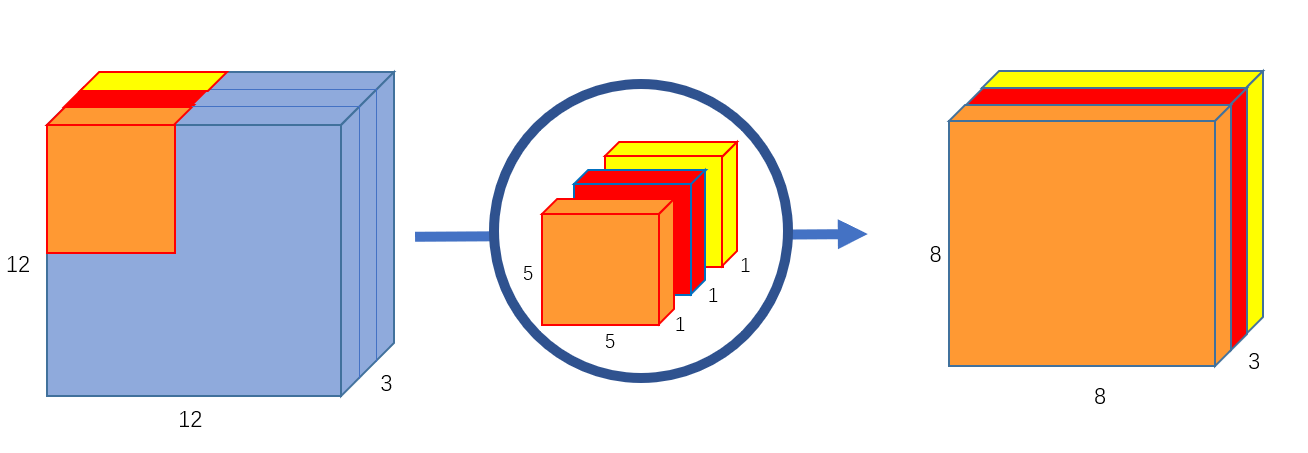


This is how a normal convolution works. I like to think of it like a function: 12x12x3 — (5x5x3x256) — >12x12x256 (Where 5x5x3x256 represents the height, width, number of input channels, and number of output channels of the kernel). Not that this is not matrix multiplication; we’re not multiplying the whole image by the kernel, but moving the kernel through every part of the image and multiplying small parts of it separately.

A depthwise separable convolution separates this process into 2 parts: a depthwise convolution and a pointwise convolution.

**Part 1 — Depthwise Convolution:**

In the first part, depthwise convolution, we give the input image a convolution without changing the depth. We do so by using 3 kernels of shape 5x5x1.

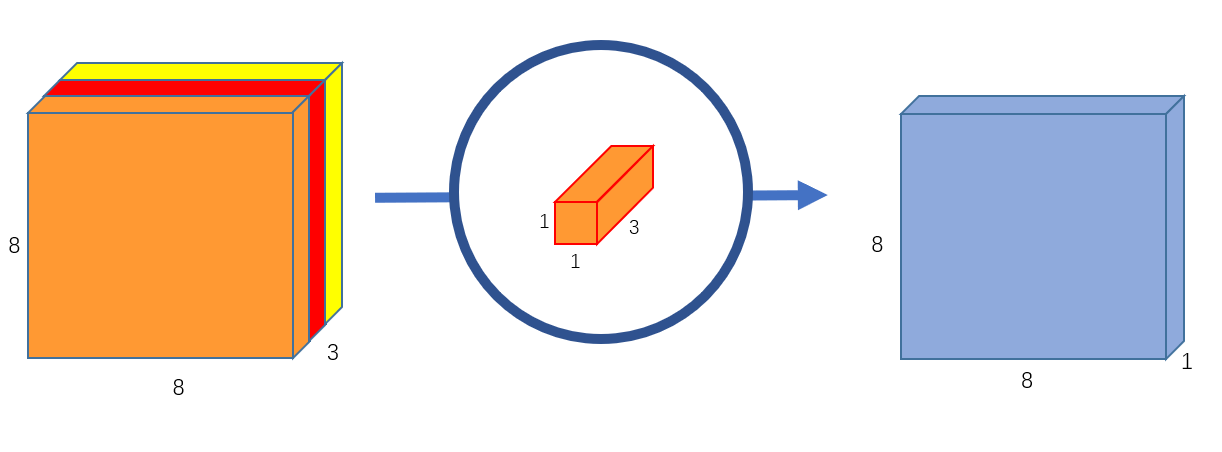


Each 5x5x1 kernel iterates 1 channel of the image (note: 1 channel, not all channels), getting the scalar products of every 25 pixel group, giving out a 8x8x1 image. Stacking these images together creates a 8x8x3 image.

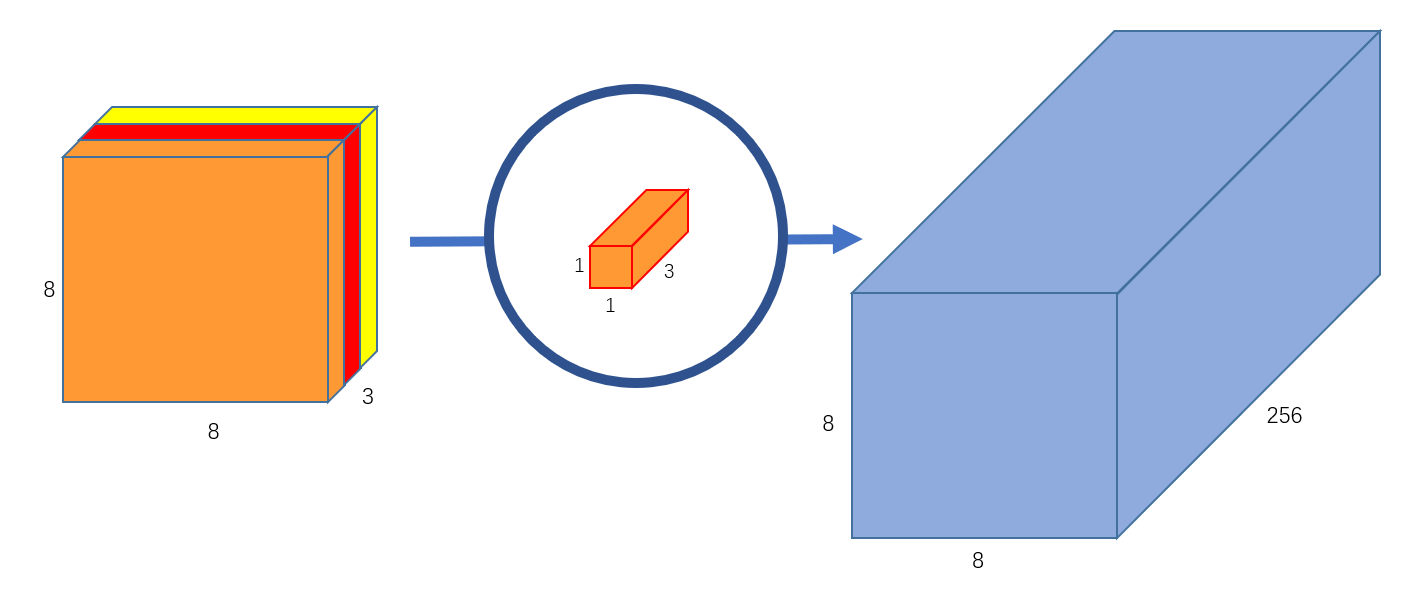
**Part 2 — Pointwise Convolution:**

Remember, the original convolution transformed a 12x12x3 image to a 8x8x256 image. Currently, the depthwise convolution has transformed the 12x12x3 image to a 8x8x3 image. Now, we need to increase the number of channels of each image.

The pointwise convolution is so named because it uses a 1x1 kernel, or a kernel that iterates through every single point. This kernel has a depth of however many channels the input image has; in our case, 3. Therefore, we iterate a 1x1x3 kernel through our 8x8x3 image, to get a 8x8x1 image.



We can create 256 1x1x3 kernels that output a 8x8x1 image each to get a final image of shape 8x8x256.



And that’s it! We’ve separated the convolution into 2: a depthwise convolution and a pointwise convolution. In a more abstract way, if the original convolution function is 12x12x3 — (5x5x3x256) →12x12x256, we can illustrate this new convolution as 12x12x3 — (5x5x1x1) — > (1x1x3x256) — >12x12x256.

Alright, but what’s the point of creating a depthwise separable convolution?

Let’s calculate the number of multiplications the computer has to do in the original convolution. There are 256 5x5x3 kernels that move 8x8 times. That’s 256x3x5x5x8x8=1,228,800 multiplications.

What about the separable convolution? In the depthwise convolution, we have 3 5x5x1 kernels that move 8x8 times. That’s 3x5x5x8x8 = 4,800 multiplications. In the pointwise convolution, we have 256 1x1x3 kernels that move 8x8 times. That’s 256x1x1x3x8x8=49,152 multiplications. Adding them up together, that’s 53,952 multiplications.

52,952 is a lot less than 1,228,800. With less computations, the network is able to process more in a shorter amount of time.

How does that work, though? The first time I came across this explanation, it didn’t really make sense to me intuitively. Aren’t the two convolutions doing the same thing? In both cases, we pass the image through a 5x5 kernel, shrink it down to one channel, then expand it to 256 channels. How come one is more than twice as fast as the other?

After pondering about it for some time, I realized that transforming the image 256 timestransform the image once simply elongate it to 256 channels

It’s worth noting that in both Keras and Tensorflow, there is a argument called the “depth multiplier”. It is set to 1 at default. By changing this argument, we can change the number of output channels in the depthwise convolution. For example, if we set the depth multiplier to 2, each 5x5x1 kernel will give out an output image of 8x8x2, making the total (stacked) output of the depthwise convolution 8x8x6 instead of 8x8x3. Some may choose to manually set the depth multiplier to increase the number of parameters in their neural net for it to better learn more traits.

Are the disadvantages to a depthwise separable convolution? Definitely! Because it reduces the number of parameters in a convolution, if your network is already small, you might end up with too few parameters and your network might fail to properly learn during training. If used properly, however, it manages to enhance efficiency without significantly reducing effectiveness, which makes it a quite popular choice.

**1x1 Kernels:**

Finally, because pointwise convolutions use the concept, I’d like to touch upon the usages of a 1x1 kernel.

A 1x1 kernel — or rather, n 1x1xm kernels where n is the number of output channels and m is the number of input channels — can be used outside of separable convolutions. One obvious purpose of a 1x1 kernel is to increase or reduce the depth of an image. If you find that your convolution has too many or too little channels, a 1x1 kernel can help balance it out.

For me, however, the main purpose of a 1x1 kernel is to apply non-linearlity. After every layer of a neural network, we can apply an activation layer. Whether it be ReLU, PReLU, Softmax, or another, activation layers are non-linear, unlike convolution layers. “A linear combination of lines is still a line.” Non-linear layers expand the possibilities for the model, as is what generally makes a “deep” network better than a “wide” network. In order to increase the number of non-linear layers without significantly increasing the number of parameters and computations, we can apply a 1x1 kernel and add an activation layer after it. This helps give the network an added layer of depth.