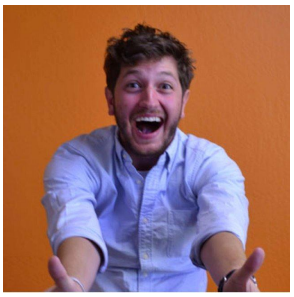
Intro to Data Science

Lesson 3 Notes

Data Analysis

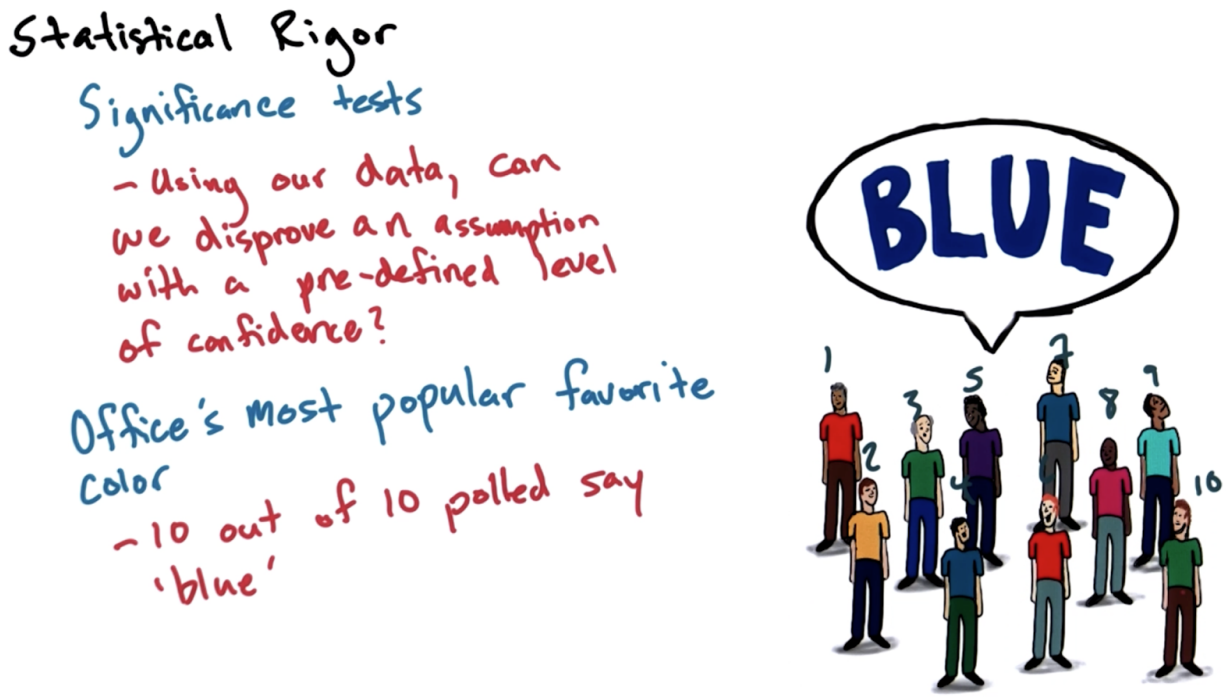
Welcome to Lesson 3



Welcome to the third lesson in Introduction to Data Science. During this lesson, we're going to be focusing on some basic methods in statistics and machine learning that you can use to analyze data. During the last lesson, we mainly focused on methods to acquire and clean data from a variety of sources, such as relational databases, APIs, and flat files. However we also discuss some basic methods that you can use to verify the integrity of your data. And while you're doing that you might learn some very basic things. For example, you might know who the tallest shortstop to ever play in major league baseball was, or, how many people over the age of 65 enrolled in the Aadhaar program in a particular district. You might find, though, that you want to answer some more subtle questions. For example, we might want to know, are left handed batters any better than right handed batters? Or, hey, is there any correlation between age and the Aadhaar rejection rate? We'll use statistics and machine learning to answer these types of questions. We'll also find that these methods are really useful for answering questions about ridership on the New York City subway. For example, we might be able to discover whether time of day, or the weather outside, influence how many people are riding the subway. During this lesson, we're going to assume some basic familiarity with stats. Either you haven't seen the stuff or if you feel lost, it might be a good idea to enroll in Stats 95. I've included a link in the instructor comments. Alright, let's jump in.

Statistical Rigor Part 1

When performing an analysis, we can usually achieve statistical rigor by way of a significance test. These test to find whether or not a sample of data can disprove some conventional wisdom or assumption, with a predefined level of confidence. Let's first discuss at a high level why statistical rigors important.

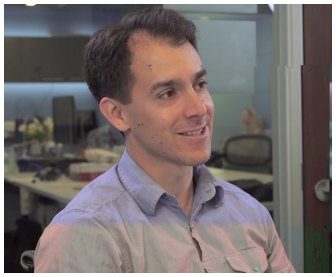


Let's say that you worked in an office of 1000 people, and you wanted to find out what the most popular favorite color in your office was. It would be a lot of work to ask all to get a pretty good idea of the office's favorite color. When you ask these 10 people, each one says "blue." With that statistic, you might say," Well, since these color, blue must be everyone in the office's favorite color." Of course both you and I have some intuitive idea that this probably isn't true. We just happened to come across ten people who all liked blue. Statistics formalizes this kind of logic. So we can assess the feasibility of a result that we get with a smaller sample when trying to say something about a larger population.

Statistical Rigor Part 2

You can also imagine a case where at first glance, our results might suggest that there's no significant difference but it turns out that there is. For example, say a website was running an AB test between two different versions of our landing page. They're trying to see which version of the page converts more visits to clicks. Say they run a test for weeks and weeks, placing 500,000 visitors into each test group. One strategy had a conversion rate of 50%, another had a conversion rate of 50.5%. If we run a significance test on this data, we can say with a significant, albeit small difference between the two strategies.

Kurt Introduction



So my name's Kurt Smith, I'm a data scientist on the Analytics team here at Twitter. Been here about two and a half years right now and I took, you know, a somewhat long and windy path to getting here. I actually started out in college studying chemical engineering and then ended up doing a PhD in computational fluid dynamics and molecular modeling And spent quite a while working in academia doing computational research I went from there I moved to San Francisco several years back, and I initially started working in the chemicals industry doing a lot of molecular modeling. And from there the first big leap I would, I may that I would say brought me more to the data science direction was, moving to a startup in the healthcare space, that was doing a lot of data analysis on a risk modeling around chronic disease. And that really was the opportunity, or gave me the opportunity to take a lot of my mathematical background, and apply it to data driven problems. After working in that area for a few years, I started to look around in other fields with interesting data challenges. And that brought me to moving to social media and Twitter.

Why Is Stats Useful

Okay, why is statistics useful in data science? There's a lot of reasons, obviously. The, the, the first reason, I would say, is, the most basic one and this goes back to why statistics was, you know, developed a hundred or so years ago. And it really is to make sure that you're making reasonable inferences from data. So you hear a lot of people talk about big data and the value of big data and how we can you know, have data on just about everything nowadays. And it's easy to get caught up in that and it's important to remember, that anytime, you're looking at any sort of data whether it's observational or experimental. You want to make sure that you're making valid inferences, right? You want to do things like check for statistical significance, understand confidence intervals. Make sure that you're not pointing you're working with in, in certain conc, wrong direction because you've over interpreted some data so I think that's the first, probably most important reasons why statistics is important to data science there's a lot more just working with data, day in and day out there's a lot of what I would call. Sort of, you know very fine-grain technical skill, to just make sure, you're doing the right thing, and you're extracting the most information you can out of data.

Quiz: Statistical Rigor Exercise

Alright, so just to recap, why are statistical significance test useful? Check all that apply. They provide a formalized framework for comparing, and evaluating data. They can make a bad result look good. They enable us to evaluate whether perceived effects in our dataset, reflect differences across the entire population.

[x] They provide a formalized framework for comparing and evaluating data

[ ] They can make a bad result look good
[x] They enable us to evaluate whether perceived effects in our dataset reflect differences across the whole population

Answer:

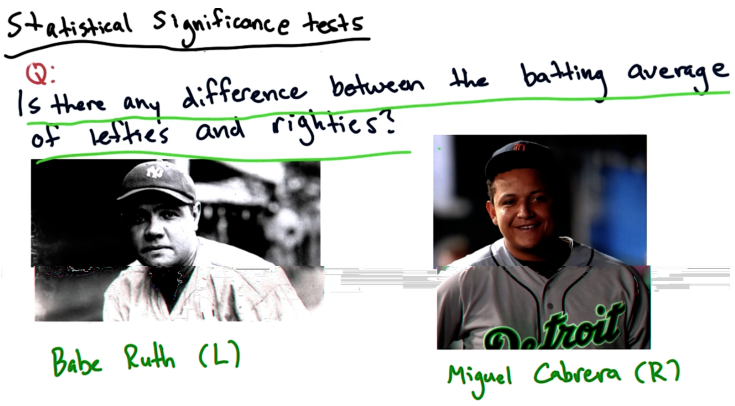
Alright. So, why are statistical significance tests useful? So, they do provide a formalized framework for comparing and evaluating data. And they do enable us to evaluate whether perceived effects in our data set reflect differences across the whole population. They do not make a bad result look good. Significant sets are useful because they provide a formalized framework for comparing and evaluating data. Different tests have different assumptions and rules that they incorporate, and using a particular test ensures that everyone is on the same page in so far as what we're assuming about our data. Significance tests also enable us to evaluate whether perceived effects in our data set reflect differences across the whole population. As was the case with our company, where ten out of ten people polled preferred the color blue. Sometimes an effect that we seen in a small sample does not reflect what might be true across the entire population. A statistical significance test let's us formally determine whether or not this might be the case. Unfortunately, a bad result is not going to look any better or worse as a result of using a statistical significance test. If our data's bad, or there's really no difference between our two samples. We're not going to be able to undo that with a test. It is possible though that different tests might give us different results. The really important thing and we'll go into this a bit more is that you need to use the right test in the right situations. Why don't we talk a little bit about how we might actually run a statistical significance test.

Quiz: Statistical Test

Let's say that we wanted to compare the batting averages of left handed and right handed hitters in major league baseball. So we're curious in answering the question, is there any difference between the batting averages of lefties and righties? We could just look at the data and try to answer this question without a significance test. But then we really couldn't trust our answer very much. It's a much better idea to answer this question using a statistical test. Many statistical tests that you might use to analyze data make an assumption about the probability distribution that your data will follow. There are many different probability distributions out there, but one of the most common and the one that I want to discuss is the normal distribution, which is also sometimes referred to as the Gaussian

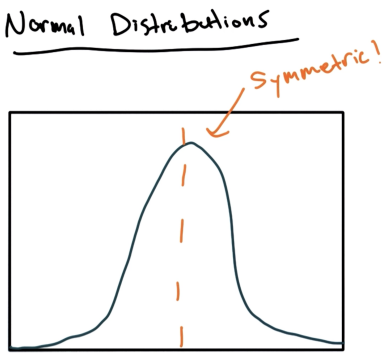
distribution or a Bell curve. If you've taken an introduction to statistics course, the normal distribution should be familiar. Which of the images below depicts a normal distribution? Check the box below the correct one.

Answer:



Normal distributions generally look like this. Although their width and center can change based on the distribution's parameters. There are two parameters associated with a normal distribution. The mean, mu. And the standard deviation, sigma. These two parameters plug in to the following probability density function, which describes a Gaussian distribution. The expected value of a variable described by a Gaussian distribution is the mean, mu. and the variance is the standard deviation, sigma squared. Normal distributions are also symmetric about their mean. If you've taken an introduction to stats course, the normal distribution should be a familiar tool.

Introduction to Normal Distribution



There are two parameters associated with a normal distribution. The mean, mu, and the standard deviation, sigma. These two parameters plug in to the following probability density function, which describes a Gaussian distribution. The expected value of a variable described by a Gaussian distribution is the mean, mu, and the variance is the standard deviation, sigma squared. Normal distributions are also symmetric about their mean. If you've taken an introduction to stats course, the normal distribution should be a familiar tool.

Quiz: Normal Distribution

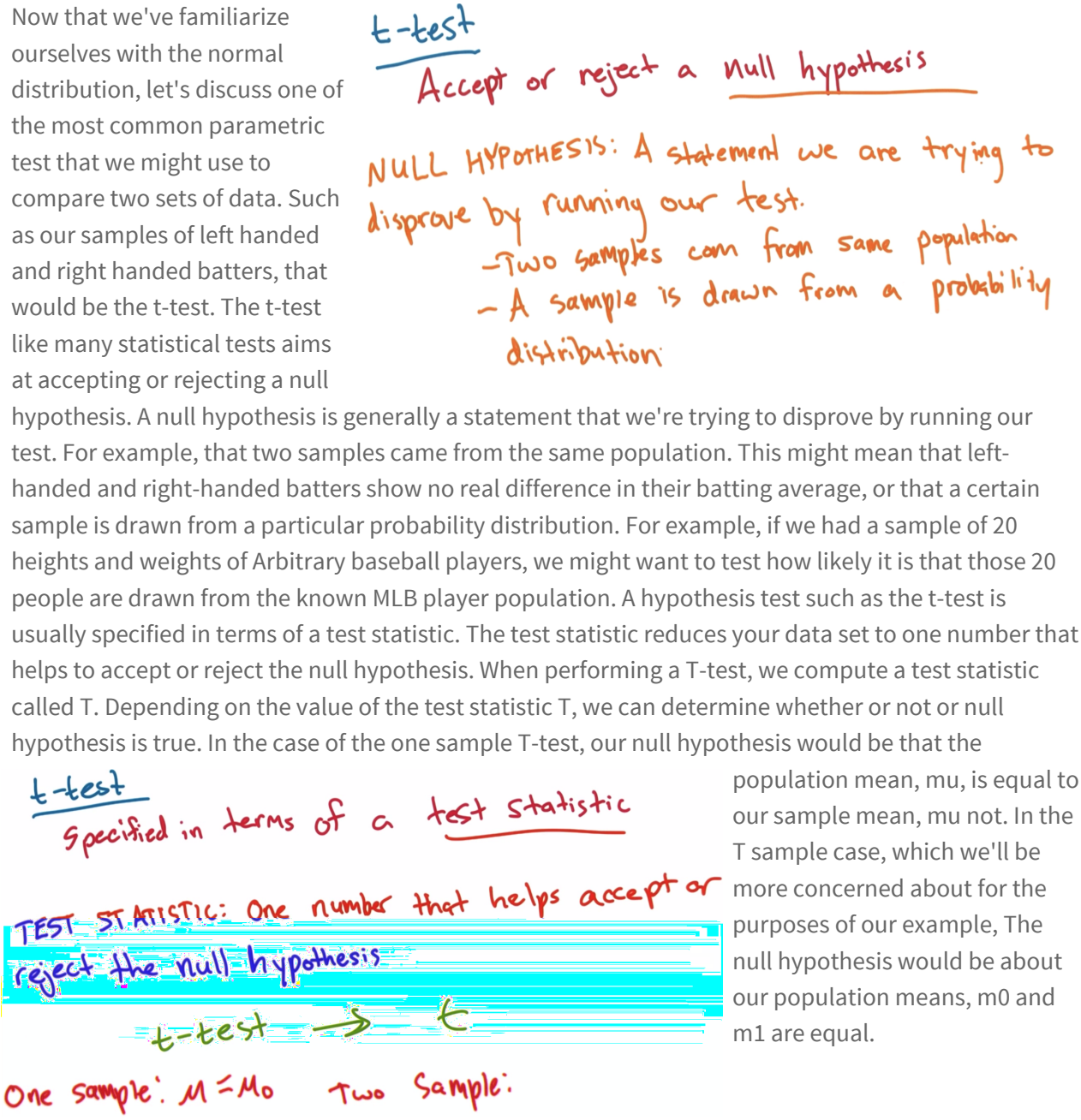
Here are the equations for a few different normal distributions. Can you determine from their probability distributions, the mean, standard deviation, and variance? Fill in your answers in these boxes below. So, mean would go in these. Standard deviation would go in here, and variance in these guys.

Answer:

Recall that the equation for a normal distribution is minus x minus mu quantity squared, over 2 sigma squared, where mu is the mean and sigma is the standard deviation. Sigma squared is the variance. If we know the form of this equation. It's not hard to look at all of these equations and figure out the values for the mean. Standard deviation and variance. They are five, and 25. Eight, one, and one. And zero, one, and one.

T Test

Now that we've familiarize ourselves with the normal distribution, let's discuss one of the most common parametric test that we might use to compare two sets of data. Such as our samples of left handed and right handed batters, that would be the t-test. The t-test like many statistical tests aims at accepting or rejecting a null hypothesis. A null hypothesis is generally a statement that we're trying to disprove by running our test. For example, that two samples came from the same population. This might mean that left-handed and right-handed batters show no real difference in their batting average, or that a certain sample is drawn from a particular probability distribution. For example, if we had a sample of 20 heights and weights of Arbitrary baseball players, we might want to test how likely it is that those 20 people are drawn from the known MLB player population. A hypothesis test such as the t-test is usually specified in terms of a test statistic. The test statistic reduces your data set to one number that helps to accept or reject the null hypothesis. When performing a T-test, we compute a test statistic called T. Depending on the value of the test statistic T, we can determine whether or not or null hypothesis is true. In the case of the one sample T-test, our null hypothesis would be that the population mean, mu, is equal to our sample mean, mu not. In the T sample case, which we'll be more concerned about for the purposes of our example, The null hypothesis would be about our population means, m0 and m1 are equal.



Welch T Test

Let's talk more about the two sample t-test, since we'll want to compare two different samples in our class project. There are a few different versions of the t-test that one might employ ,and they depend on really on what assumptions we make about the data. So we might want to ask questions such as ,do our samples have the same size ?,and do they have the same variance? . Let's discuss a variant of the t-test called Welch's t-test in more depth. Since it's the most general. It doesn't assume equal sample size ,or equal variance. In Welch's t-test ,we compute a t-statistic using following equation. T equals mu1 minus mu2, divided by the square root of sigma1 squared over n1. Plus sigma 2 squared over n2. Where mu I ,is the sample mean for the Ith sample. Sigma squared I is the sample variance for the Ith sample. And NI is the sample size for the Ith sample. We'll also want to estimate the number of degrees of freedom, nu, using the following equation. Nu is approximately equal to. Quantity sigma1 squared ,over n1 ,plus sigma2 squared over n2 ,squared over sigma1 of the Where mu I is equal to mi minus one ,and this is the degrees of freedom associated with the Ith variance estimate. If you're unfamiliar with degrees of freedom again it might be a good idea to brush up on your stats concepts with the audacity's intro to stats course. A link is provided in the instructor comments. Alright so once we have these two values, we can estimate the P value. Conceptually, the P-value is the probability of obtaining the test statistic at least as extreme as the one that was actually observed ,assuming that the null hypothesis was true. The P value is not the probability of the null hypothesis is true given the data. So again, just as a thought experiment. Say we were testing whether left handed or right handed baseball players. Were better batters by looking at their average batting average. If the P value is .05, this would mean that ,even if there is no difference between left handed and right handed batters, since that's our null hypothesis. So, even if this was true, we would see a value of t ,equal or greater to the one that we saw 5% of the time. When performing a statistical test like this, we usually set some critical value of P. Let's call it P critical. If P falls below P critical, then we would reject the null hypothesis. In the two sample case, this is equivalent to stating that the mean for our two samples is not equal. Calculating this P value for a given set of data can be kind of of tedious. Thankfully, we seldom have to perform this calculation explicitly.

Programming Quiz: Calculating T And Nu

All right. Why don't we try computing these two quantities, t and nu, given some example batting average data. So, let's say that I had two samples. One with 150 data points, a mean batting average of 0.299, and a variance of of 0.307, and a variance of 0.08. What are the values of t and nu given this data? Fill in your answers in these boxes below, and round them to three decimal places.

Answer:

We can just plug the information provided into our equations for t and new. If we look at the equations for t, t is 0.307 minus 0.299 over the square root of 0.05 over 150 plus 0.08 over the quantity 0.05 over 150 plus 0.08 over 165 squared. Over 0.05 squared over 150 squared times 149 plus is going to be 307.199. The t statistic here, again is sort of some idea of, you know, how extreme is our result and how likely is it to disprove our null hypothesis? Meanwhile, the nu here is kind of some idea of how many independent variables went into calculating this t-value?

Programming Quiz: Welch T Test in Python

Alright, so we've established that we can do this t-test and compute these t-values and nu values and, and p-values in the abstract mathematical sense, but you might be wondering how do I do this in python? Is there a simple way to do this? Well, the answer is yes, this can be done in the following way. First you simply import scipy.stats. And then you call scipy.stats.ttest_ind and then supply as arguments two lists, which are your two sets of data, and then an optional argument, equal_var=false. This indicates whether or not we think the variance of our two samples is equal. This equal_var=false argument makes this particular call of the t-test equal to Welch's t-test. This function will return a tuple. The first value is the t-value for your data. And the second value is the corresponding p-value for a two-tailed test. The values returned by scipy.stats.ttest_ind assumes that you are performing a two-sided t-test where we're only testing for whether the means of our two samples are different. Say we wanted to do a one-sided t-test, that is, we're interested in testing whether one mean in particular is greater than or less than the other. How might we do this given the output that this function produces? Write your answer in the text box below. Here's a hint. We can still use the t value and p value that this function returns. Don't worry. This won't be graded.

Answer:

Alright, so with the symmetric distributions like our normal distribution, the one sided p-value is simply going to be half of our two sided p-value. Remember that the p-value is the probability that given the null hypothesis is true, we would observe a test statistic at least as extreme as what we saw. So instead of having both sides of this distribution, we're really only going to have one side of the distribution. So we have a one-sided t-test where we're checking whether the mean of sample one is greater than the mean of sample two. We still want p over two to be less than our p critical, maybe that's 0.05, but we want our t value to be greater than zero. Whereas for a less than one-sided t-test, where we're testing whether the mean of sample one is less than the mean of sample two, we want p over 2 to be less than p critical, but we want our t value to be less than 0.

Programming Quiz: Welch T Test Exercise

Okay, why don't we work through an example. Let's say we had a CSV file located at this directory, and that it contained a number of columns. Among them are a player's name. We call that column, name. There's also a column called handedness, L for left-handed R for right-handed, and a column for the player's career batting average, called avg. Write a function that will read that data into a Pandas dataframe, and then run Welch's t-test on the two cohorts defined by handedness. With a significance level of 95%, if there is no difference between the two cohorts, return a tuple consisting of true, and then the tuple returned by scipy.stats.ttest. If there is a difference, return a tuple consisting of false, and then the tuple returned by scipy.stats.ttest. So an example of what your output might look like is false and then the tuple 0.405, 0.006. Your code should go here. Since we're using Welch's t-test, we're assuming our data is sampled at random and that both samples follow a normal distribution. We don't make any assumptions about the variance of both samples. They may be the same, but they also may be different.

Answer:

Alright, why don't we walk through our solution function compare_averages. Remember that we're performing a t-test on two sets of baseball data, left-handed and right-handed hitters. We want to perform a Welch's t-test and determine with 95% confidence whether or not the average batting average of the two cohorts is different. In other words the left handed hitters have a statistically significant different average batting average than right handed hitters. First, we read our data into a panda's data frame. So baseball date equals pandas.read_csv and then the location of our csv file. Then we're going to split the data into two data frames, one for left handed hitters and one for right handed hitters. We do this by saying baseball_data_left equals baseball_data, and then index on baseball_data handedness equals L. We can do the same thing for right-handed hitters. Then we'll use scipy.stats.ttest_ind to perform a Welch's t-test on these two data frames. So results equals scipi.stats.ttest_ind and then we pass in baseball data left average, baseball data right average, and we say equal_var equals false. Now we're going to produce an output in the desired format. So if result one is less than or equal to .05, that is if our p value is less than or equal to .05, will return false and then the output of scipy.stats.ttest_ind. Else will return true, and then the result of scipy.stats.ttest_ind. So this is a basic implementation of how we can do Welch's t-test using Python.

Non Parametric Test

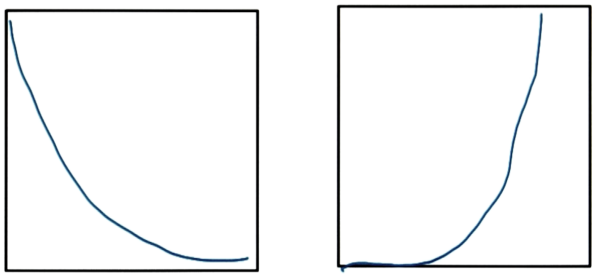
All right. So let's say that we look at our data and it's clearly non-normal. Or we use a statistical test like the Shapiro-Wilk Test and find that our data's non-normal that way. Is there anything that we can do? Well, first off there's some math that says that we have enough data. That we have, you know, a large enough sample size we can actually use tests that assume normality. For example, the t test. Even when our data is not normal. But there also exists nonparametric tests that we can use

to compare two samples. A non-parametric test is a statistical test that does not assume our data is drawn from any particular underlying probability distribution. One such test is the Mann-Whitney U Test which is also sometimes referred to as the Mann-Whitney Wilcoxon Test. This is a test of the null hypothesis that two populations are the same. Again I don't want to go to in depth into the theory of how this test works, but I did want to tell you that things like this exist and show you that this test can also easily be implemented in Python using Scipy. We'd simply say U and P are equal to scipy.stats.mannwhitneyu. And provides as arguments are two samples which we'll call here X and Y. This function will return U, the Mann-Whitney test statistic. As well as P, which is the one sided P value for this test. The P value here again acts as it did for the T test. Note that the Mann-Whitney U test simply tests whether or not these samples came from the same population. But not necessarily which one has a higher mean or a higher median or anything like that. Because of this it's usually useful to report Mann-Whitney U test results along with some other information. Like, the two sample means, or the sample medians, or something like that.

Non Normal Data

When performing a t-test, we assume that our data is normal. In the wild, you'll often encounter probability distributions. They're distinctly not normal. They might look like this, or like this, or completely different. As you'd imagine, there are still statistical tests that we can utilize when our data is not normal. Why don't we briefly discuss what you might do in situations like this. First off, we should have some machinery in place for determining whether or not our data is Gaussian in the first place. A crude, inaccurate way of determining whether or not our data is normal is simply to plot a histogram of our data and ask, does this look like a bell curve? In both of these cases, the answer would definitely be no. But, we can do a little bit better than that. There are some statistical tests that we can use to measure the likelihood that a sample is drawn from a normally distributed population. One such test is the shapiro-wilk test. I don't want to go into great depth with regards to the theory behind this test, but I do want to let you know that it's implemented in scipy. You can call it really easily like this. W and P are going to be equal to scipy.stats.shapiro data, where our data here is just an array, or list containing all of our data points. This function's going to return these two values. The first, W is the Shapiro-Wilk Test statistic. The second value in this two-pole is going to be our P value, which should be interpreted in the same way that we would interpret the p-value for our t-test. That is, given the null hypothesis that this data

is drawn from a normal distribution, what is the likelihood that we would observe a value of W that was at least as extreme as the one that we see?



Quiz: Definition of Non Parametric Test

Just to quickly recap, what is the definition of a non-parametric test? A non-parametric test is a statistical test that assumes the data is drawn from a non-parametric probability distribution? Assumes the data is drawn from a non-Gaussian probability distribution? Does not assume the data is drawn from any particular underlying probability distribution? Or is the same as Welch's t-test? Check the correct answer.

[ ] Assumes the data is drawn from a non-parametric probability distribution
[ ] Assumes the data is drawn from a non-Gaussian probability distribution
[x] Does not assume the data is drawn from any particular underlying probability distribution
[ ] Is the same as Welch's t-test

Definition of Non Parametric Test

The correct answer is that a non-parametric test does not assume the data is not drawn from any particular probability distribution. There's not really any such thing as a non-parametric probability distribution. This terminology non parametric really describes test types or statistical methods. Assuming a non-Gaussian probability distribution would still assume some probability distribution. And finally, Welch's t-test is definitely not a non-parametric test, since it assumes a certain underlying probability distribution so there's no way that the non-parametric test is the same as Welch's t-test.

Just the Tip of the Iceberg

These have just been some of the methods that we can use when performing statistical tests on data. As you can imagine, there are a number of additional ways to handle data from different probability distributions or data that looks like it came from no probability distribution whatsoever. Data scientists can perform many statistical procedures. But it's vital to understand the underlying

structure of the data set and consequently, which statistical tests are appropriate given the data that we have. There are many different types of statistical tests and even many different schools of thought within statistics regarding the correct way to analyze data. This has really just been an opportunity to get your feet wet with statistical analysis. It's just the tip of the iceberg.

Quiz: Predicting Future Data

Now we know how to analyze existing data. However, is there any way we can make predictions about data we'll collect in the future, using data we've collected in the past? Write some ideas on ways to do this, in the text box below. Try to think about methods we used to predict Titanic survivors in Assignment number one.

Answer:

In addition to statistics, many data scientists are well versed in machine learning. You might be wondering what exactly is machine learning. Well, machine learning is a branch of artificial intelligence that's focused on constructing systems that learn from large amounts of data to make predictions. Machine learning could also be useful to predict which movies you might like on Netflix. Or how many home runs a batter may hit over the course of his career. These are all potential applications of machine learning.

Why Is Machine Learning Useful

Okay, why is machine learning useful? So that's, that, that's a great question and, and it, you know, I, I think it touches on this, this sort of classic question, or this long running discussion, on what is the difference between statistics and machine learning. there's, there, there's a lot of great papers. There's one classic paper, I, I forget the name exactly, but it's something along the lines of the two cultures, and talking about why these two different approaches developed. So I think, if you look at machine learning, it, it grew out more of a computer science direction, and, and it grew out of a lot of areas where people had very practical hands on questions, right? How can we build the best recommendation system? How can we make the best predictions or classifiers for, for a given problem? And machine learning has developed a lot of really, really good techniques for doing that, for doing things that work very well in practice. And so I, I think that's there, there are a lot of applications where that's really what you're getting at particularly when you have the opportunity to build a product that will you know, take some action based on, based on some algorithms. Those are areas where machine learning is really at the cutting edge of coming up with the most effective ways of making decisions in real time based on data. I see.

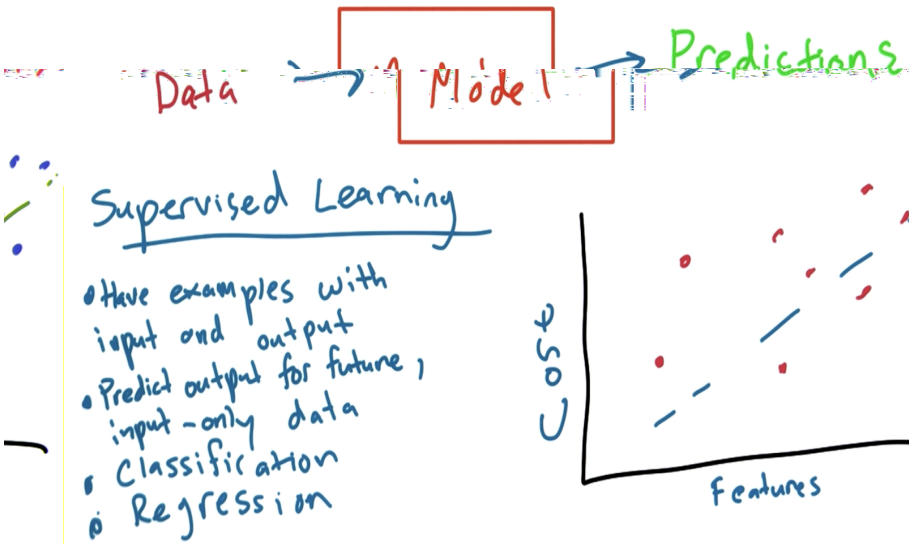
Stats vs. Machine Learning

You might be wondering what the difference is between statistics and machine learning. In short, the answer is not much. More and more, the 2 fields are converging and they share many of the same methods. However there are couple of significant philosophical differences between the 2 subjects. Generally, statistics is focused on analysing existing data, in drawing valid conclusions, whereas machine learning is focused on making predictions. What this translates to is the following: In statistics, we care a lot about how our data was collected, and drawing conclusions about that existing data using probability models. For example, we might try to answer the question are left handed hitters statistically better than right handed ones? In the case of machine learning, we're a little bit more focused on making accurate predictions and less committed to using a probability model if there's a more accurate method that doesn't use them at all. As long as our machine learning method consistently makes accurate predictions for example, how many home runs will a player hit, we aren't too worried about what assumptions the model makes.

Different Types of Learning

There are many different types of machine learning problems, but two common ones are supervised learning and unsupervised learning. Machine learning typically involves generating some type of model regarding the problem we're trying to solve. We'll feed data into this model and then try to make predictions. In supervised learning, there are labeled inputs that we train our model on. Training our model simply means teaching the model what the answer looks like. So, for example, if we were detecting spam using supervised learning, we might have 100 emails where we know whether or not they're spam. We can use these emails, where we know whether or not the email is spam, and train a model that can predict whether or not a future email will be spam based on a bunch of characteristics, maybe its contents, or whether or not it has an attachment, or things like that. Another example of supervised learning would be estimating the cost of a new house, given that we have a number of examples where we know about a bunch of features like the square footage, or the number of rooms, or the location. And we also know how much that house sold for. We could train a model, and then predict

how much a future house will sell for, given we know all the same parameters. This is an example of regression. When performing unsupervised learning, we don't have any such training examples. Instead we have a bunch of unlabeled data points, and we're trying to understand the structure of the data, often by clustering similar data points together. For example, if we were to feed an unsupervised learning algorithm a bunch of photos, it might split the photos into groups, say photos of people, photos of horses, photos of buildings, without being told a priori what those groups should be. It might not know that the groups are people, or horses, or buildings, but it can tell that these distinct groups exist.



Kurt's Favorite Machine Learning Algorithm

Sure, sure. That's a great question. So I would say, you know, just as a, as a caveat, that really depends a lot of the type of problems any given data scientists are working on, the type of questions you're, you're looking at. For me personally, a lot of the questions that, that I look at and a lot of the things that interest me, tend to be fairly high level and somewhat loosely defined and I mentioned, looking a lot more at the social aspects of how people use social networks. And a lot of topics like that, sometimes the most useful thing is really to just get a broad and qualitative understanding of, of the data you're looking at. So, two of the most valuable techniques I found for that are clustering. There are various approaches, kamines, clustering, hierarchical clustering. And then other dimensionality reduction techniques or things like principal component analysis, PCA. One, one thing, just speaking very specifically that I found very useful is the combination of K-means, clustering and PCA. So if, if you generate clusters and then use PCA to take the most, meaningful vectors to plot those clusters on, you can often do a really nice job of reducing dimensionality of a data set and understanding, the significant differences between clusters in a visual way.

Prediction With Regression

Let's talk a little bit more about a specific example of supervised learning. We might want to ask the question, can we write an equation that takes as input a bunch of information. For example, height, weight, birth year or position about baseball players and predicts their lifetime number of home runs. To do this, we want to write an algorithm that does the following. Takes in the data points for which

we have all these input attributes, and the player's lifetime number of home runs. Then builds the most accurate equation to predict lifetime number of home runs, using these input variables. We can then use this equation to predict the lifetime number of home runs for players for whom we did not initially have number of home runs. But we do have all the other data. So, their height and weight and birth year and position maybe. One way that we might be able to solve this problem is using linear regression and one basic implementation on machine learning is to perform linear regression by using gradient descent. What is this? How does this work? Lets learn about gradient descent by working through an example that utilizes our baseball data set.

Quiz: Linear Regression With Gradient Descent

When performing linear regression, we usually have a number of data points. Each data point has an output variable, which we'll call Y, and then a number of input variables which we'll call X1 through Xn. So in our baseball example Y here is lifetime number of home runs, and our X1 through Xn are things like their height and weight. Generally speaking, we're trying to build a model that predicts values of the output variable for each data point by multiplying the input variables by some set of coefficients that we're going to call theta1 through thetaN. Each theta, which from here on out we'll call the parameters, or weights, of our model, tell us how important an input variable is when predicting a value for the output variable. So if theta 1 is really small, then X1 is not very important in predicting Y. Whereas if theta N is very big, then Xn is a big contributor, it's the value of Y. This model is built in a way that we multiply each X by the corresponding theta, and add them up to get Y. So that our equation would look something like this. The best equation is the one that minimizes the difference across all data points between our predicted Y and our observed Y. What we need to do is find the thetas that produce the best predictions, that is minimizing this difference. Since our prediction can be too high or too low, rather than just look at the difference between our prediction, which is here, the green line or our actual value, which are these blue dots, we're going to look at the error squared, rather than just the errors themselves to measure the quality of a set of thetas. The lower the sum of the error squared, the better. Why don't we make sure that the distinction between input variables and output variables is clear? Say that we had age, weight, height and batting average of a bunch of Major League baseball players. We're trying to train our model to predict batting average. What are the input variables? Check all the boxes corresponding to input variables.

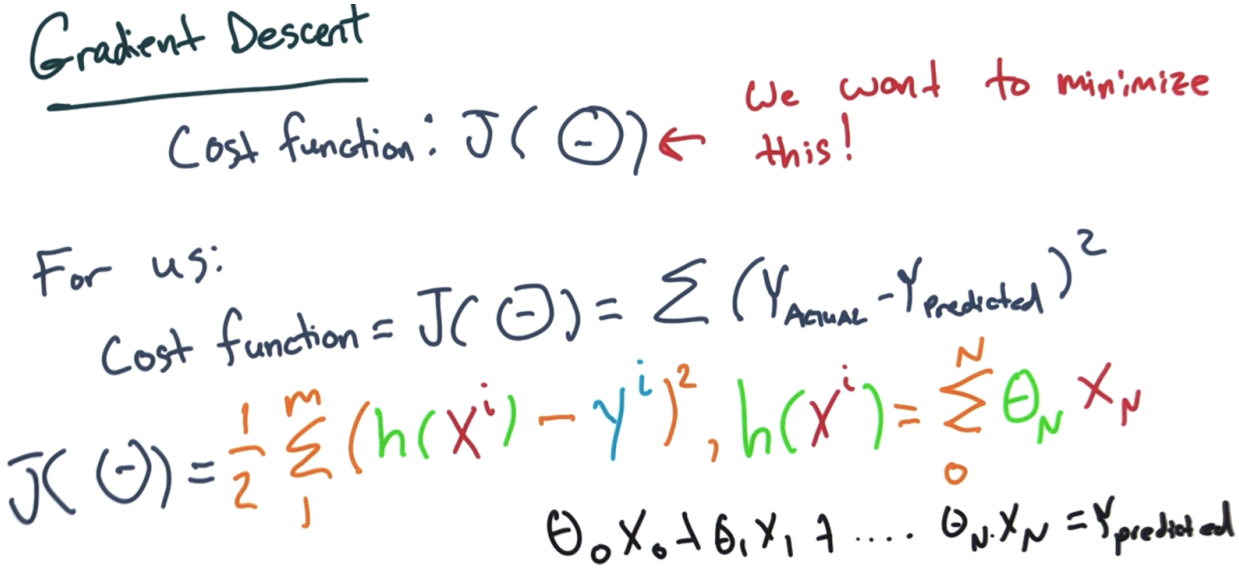
[x] Age
[x] Height
[x] Weight
[ ] Batting Average

Linear Regression With Gradient Descent

In this example, where we have age, weight, height, and batting average, and we're trying to predict batting average, our input variables are going to be age, height, and weight. We're hoping to use these input variables to predict the output variable, batting average.

Cost Function

So we've determined that we want to choose a set of parameters that provide the best predictions for our output variable. But how are we going to do that? We'll use a method called gradient descent. In order to do gradient descent, we're first going to need to define some cost function. Which we'll call J of theta. I'm going to use a big theta here to represent our entire set of thetas. And I'll use this notation throughout the rest of this lesson. The cost function is meant to provide a measure of how well our current set of thetas does at modeling our data. So we want to minimize the value of the cost function. As we discussed just a moment ago, when we're doing linear regression, our cost function J of theta. Can simply measure the sum of the squares of the difference between our predicted and actual output values. I'm going to formalize this a little bit and say this, J of theta equals one half sum from one to M of h of X super script I minus Y super script i squared. Where h of X superscript i is going to be the sum from zero to N, of little theta, sub N, little x, sub N, so, there's a lot going on here. I've color-coded this. Let's walk through this. Let's first talk about this, h of X superscript i. So I've kind of introduced a new notation here, X superscript i is going to represent our entire collection of X's. So we have n X's and since this X superscript i is a vector of al N of those X's, note that we start the sum here at zero. We're going to create a new X of zero which is just going to be one and there'll be a corresponding theta sub zero.

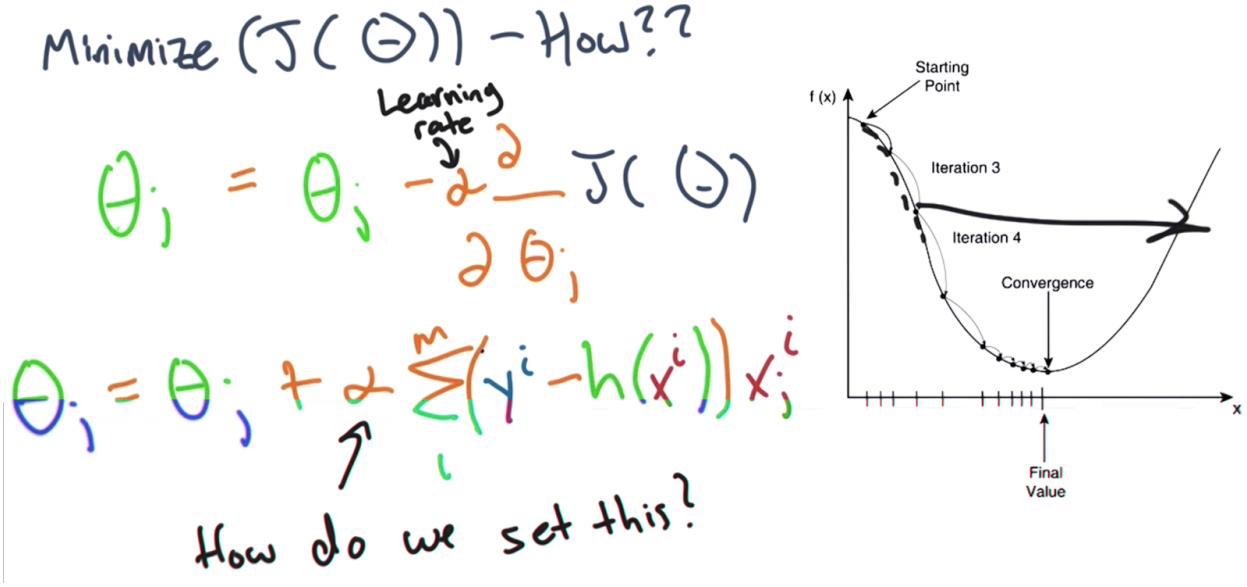


This is in case there's need to be a constant term in our equation. We're summing up theta and times X N for all N. So this is equivalent to the equation that we wrote earlier. We're basically writing theta 0, x0, plus data 1 x 1 plus etcetera, until we get to theta N times xN. So that's what h of X is. And you'll recall that this is actually going to produce our predicted y. That's what our model says. We're going

to add up all of these thetas times exes. And then we'll get a Y predicted. Great so now we go back here and we're saying okay, we're going to take our Y predicted, which takes in this X superscript i, which again is this, you know, collection of smaller X's. And then we're going to subtract the actual Y value. And we're going to do this from the number of data points we have. So if we had we'd say hey for every point, using the model we have, the thetas that we have right now, let's calculate the predicted Y, subtract the actual Y and square that difference. And we're going to add this up over all data points. So this is just a very mathematically vigorous way of expressing this. Basically that our cost function. J of big theta, is going to be the sum over all of our data, of Y actual minus Y predicted squared.

How To Minimize Cost Function

Alright, so how do we find the correct values of theta to minimize our cost function j of theta? We're going to use a search algorithm that takes some initial guess for theta and iteratively changes theta; such that, j of theta keeps on getting smaller and smaller until it converges to some minimum value. This algorithm is called Gradient Descent. So it starts with some initial theta. And we're going to update our values of theta according to this equation. For each theta j, our new theta j is going to be equal to that same theta j, minus some constant alpha. Times the derivative of j of big theta, with respect to that theta j. Note the derivative here, d, d theta j of j of theta. If you're less familiar with calculus, what we're doing here is we're calculating the rate of change of j with respect to this particular theta j. We're going to simultaneously perform this update for every single theta j in our big theta. That is, every single theta j in our entire set of thetas.



This alpha here is a parameter of the algorithm called the Learning Rate. What we're essentially saying in this equation is hey, in the space of all possible values of theta, what's the tiniest step I can make in a certain direction such that I will make the value of j of theta smaller. I'm not going to go into the calculus here, but based on our cost function, j of big theta, and the dependence it has on all these

various theta j's. If we were to actually perform this differentiation, the actual equation that we'll use to update our theta values is this. Theta j is equal to theta j, plus alpha, the sum from one to m, of y superscript i, minus h of x superscript i. In all of this, we are going to multiply it by x sub j superscript i. So, basically, what we are saying is, we'll take our existing level of theta, we'll increment just a little bit by this learning parameter. We are going to sum up the difference between our observed y value and our predicted y value and for a particular value of theta j. Remember these j's are running from zero to n and these could be our input parameters like height or weight. We're going to multiply this difference, times the actual x value here. You may be wondering how we set a value for the learning parameter alpha. A rigorous discussion is beyond the scope of this class. But the important thing to note is that the smaller Learning Rates will cause the algorithm to take longer to converge on the optimal theta values. Since you're taking smaller and smaller steps down this curve. Larger values of alpha can converge more quickly, but you are more prone to skip over the cost functions minimum values. This can cause j to increase rather than decrease when you update these theta values. The best way to make sure your alpha is suitable is simply to keep track of your cost functions value as you iteratively update it, and just make sure that it's always going down. If it's not, your learning rate's too high.

Programming Quiz: Gradient Descent in Python

Okay, so in the general mathematical sense that's how we would optimize the parameters on our model theta by using batch gradient descent to minimize our cost function, j of theta. Below is an implementation of linear regression and gradient descent to try and determine how many lifetime home runs a player in a baseball data set will hit. Given their height and weight. I'd encourage you to really look through all of this code and make sure you understand how it works. But for now, I'd like you to write some code here that updates the values of theta a number of times equal to num_iterations. You should initialize an array called the cost_history, and every time you've computed the cost for a given set of thetas You should append it to cost history. The function should return both the final values of theta in your theta array, and cost history array. Your code should go here. Feel free to use other helper functions in this code, such as compute cost.

Answer:

So why don't we walk through our solution here. First we initialize this variable m, which we say is equal to length of values. So we're basically saying here hey, this is how many data points we have. We also initialize this variable called cost history, which is going to track how our cost function evolves over our iterations. We say for i in range num iterations, which basically says hey we're going to update theta in the costs. And number of times equal to the number of iterations. Let's calculate the predicted values. We'll do that by taking the dot product of the features and theta. Then we're going to update our values of theta. This looks pretty complicated, but you'll note that it's the same

as the equation that we just discussed, which we'll pull up again here. So, we subtract our observed values from our predicted values, take the dot product of this vector with the features, and then essentially multiply by this alpha over m feature and subtract that from theta. This is effectively gradient descent in action, so we're taking a very small step in the direction of the steepest gradient. Then we're going to use another function called compute_cost which we see up here. So you compute the cost function given our features, our values and our theta. Then we'll append the newest cost to the cost history list. And once we've done this a number of times equal to the number of iterations, we'll not only return our final set of thetas, but also the entire cost history. You'll see that if we run this function, we get this array, which are theta values. About 45 minus 9 and 13, and these values kind of indicate how significant each factor is in determining our predicted value. So this 45 corresponds to a constant term, the minus 9 corresponds to height and the 13 and change corresponds to the weight. And then we see here how the cost function has evolved over time. Ideally, if we had a perfect model, this cost function would approach zero. You'll see that our cost function doesn't, you know, get close to zero. It, it does decrease a good amount as we train this model, but it doesn't do super well. And so this might lead you logically to a natural question. Which is, hey you know not all models are created equal. Maybe our model is actually bad. Is there some way we can evaluate our models? Why don't we discuss this a little bit more?

Coefficients of Determination

Alright, so say that we've determined the coefficients for a linear model for our data using gradient descent, or even some other method. Just because we're able to come up with some model doesn't mean that it's a good one. The data could be distinctly non-linear. Or, maybe the attributes that we've trained our model on have little to no bearing on our output variable. We need some way to evaluate the effectiveness of our model. One way we can measure this is a quantity called the coefficient of determination, also referred to as R squared. If we have a bunch of values, yi through yn. A bunch of predictions, fi through fn and an average value for our data, y bar. We can define the coefficient of determination as R squared is equal to fi squared, over the sum over n of yi minus y bar squared. The closer R squared is to one, the better our model. And the closer it is to zero, the poorer our model. The coefficient of determination gives us a pretty effective way to evaluate how good our model is.

Programming Quiz: Calculating R Squared

Let's make sure that you're comfortable with the coefficient of determination and computing it. Write a function that, given two input numpy arrays, data, and predictions, that returns the coefficient of determination, r squared, for the model that produced the predictions. Numpy has a couple of functions, numpy.mean and numpy.sum, that you might find useful here, but you don't have to use them. Your code goes here.

Answer:

All right, let's walk through this solution. So first we're going to compute the denominator, so the data minus the mean of the data squared, and then we'll take the sum of that. Then we're going to compute the numerator, which means that we'll compute the predictions minus the data squared, and sum that up. And then we'll compute R squared. So we'll just subtract this fraction, this guy over this guy, from one. Then we'll return R squared.

Other Considerations

What we've covered so far in this class is a very simplistic and a very partial coverage of linear regression with gradient descent. If you were ever to try and use linear regression to solve a real world problem, there are a lot of additional considerations that you would want to think about in order to seriously implement linear regression with or without gradient descent. First of all, gradient descent is only one implementation of linear regression. There are a bunch of other ones, and in some sense, they may be better. Ordinary Least Squares for example, is always guaranteed to find the optimal solution when performing linear regression, whereas gradient descent is not. Additionally, we haven't really talked about parameter estimation and putting confidence intervals on those parameters. In the models that we've built, we've given exact values for all of our thetas. But as you can imagine, there's some confidence that we have in those values. You can imagine doing more thorough statistical analysis in saying what are the confidence intervals on these parameters? And we could answer questions like what is the likelihood that I would get this value for this parameter if this parameter actually had no effect on our output variable? We also might worry about issues like over or under fitting. This isn't so much a problem with linear regression, but with more complicated models, you might expect our model to over-fit. Say we had some data like these red points that was approximately linear. If we had a model with many different parameters, we might be able to fit the data points exactly whereas the actual underlying model is this black line. One way to deal with this is usually to split our data into a training set and a test set. Then we can train a model on the training data, and then test it on the test data, just as the name would imply. This is called cross validation. We might also see under-fitting, where we have data that's clearly nonlinear, but we only try and fit a linear model to it. This is also a problem that you might encounter. Another issue worth mentioning again is that our cost function may have numerous local minima. That means that when using gradient descent, our algorithm can become trapped in a local minimum that isn't actually the global minimum of the cost function. This isn't a problem of other implementations of linear regression, but when using gradient descent, this means that maybe we perform gradient descent numerous times, randomizing our initial values of theta with different random values every time. You may also already know this, but when using a random number generator to generate random values, it's important to seed your random values and to store that seed. That way, you have a result that's replicable and other researchers can look at your findings and also produce them on their own machines when they run the same script. This has been a whirlwind tour of a bunch of additional considerations that we

didn't really touch in this lesson. These are all really important in machine learning, statistics, using linear regression in the real world. I really wanted to focus here on just implementing gradient descent and the basics of how it works. But if you really want to use any of these things to solve a real life problem, you'd want to think about local minima, you know other implementations of linear aggression, the confidence intervals on your parameters and if you're using a statistical modeling package in R or Python or Stata, they'll usually give you these confidence intervals. And use a reliable implementation of linear aggression.

Kurts Advice for Machine Learning Best Practices

So what, what are the most important do's and don'ts to keep in mind in data science. I think I would mention two. One is a bit more qualitative and one is a bit more quantitative so, on the qualitative side, I would say the first thing is any problem you're looking at it's always very valuable I find to start by thinking about. What sort of things do we know, what sort of expectations do we have and what sort of qualitative things can we get from an exploratory analysis of the data. So as I mentioned, using things like k-means and PCA are a good way to start to do some sort of dimensionality reduction, some way of getting the data down to a point where you can. Look and get some qualitative insights. Understand the general structure of it, and quite often, in, in certain topics that we work on, you have some intuition of, of what you would expect things to look like. You can start to see patterns emerge in data that, you know, that makes sense. That, that. either confirm or, or you know, possibly go against other theories or ingrained belief that people have. so, so, I think getting data down to that point is very important. The more quantitative suggestion I would have, has to do with trying to understand causal connections. And so this is a very common thing that comes up In building predictive models, you may have a number of features, that you put into a model to predict some outcome. And it's very, very important when doing that. So often the question comes up not just being able to predict an outcome but understanding which features, are actually causing it. It's important to use a lot of caution around that. To never just dump a bunch of data into a module with lots of features, and then just naively look at the things that have the strongest weights in your model and say, Oh, here's what's driving it. So it's very important when you're asking those kinds of colossal questions to proceed very carefully and, and rigorously.

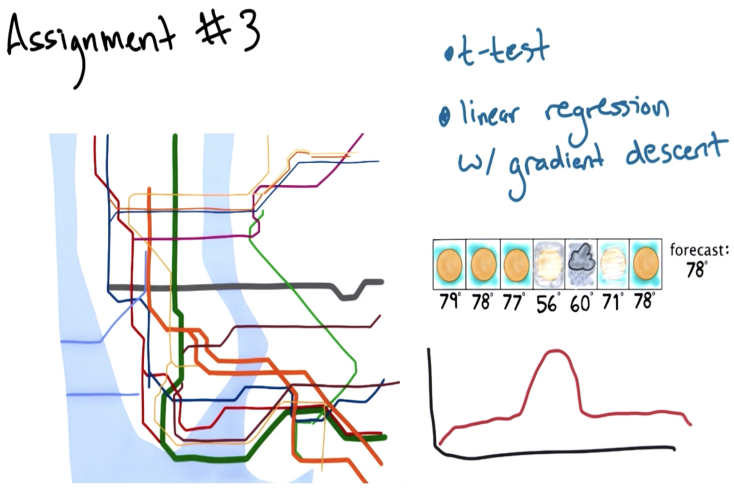
Kurts Advices for Aspiring Data Scientist

Okay. Do I have tips or advice for data scientists? >> To become a data scientist. >> Yeah, for aspiring data scientists. That's a great question and I think going back to the issue of what is data science. Like I said, I think there's really I would, I would divide it into 3 somewhat different parts. And, and so one of the first pieces of advice that I would give to people looking to get in the field is to think about what you really like, what, what part of this kind of work do you really enjoy. I think for some people it really is the process of building things, of writing code, of building a product that let's say you know,

surfaces recommendations or finds, you know similarities between users of a product. So you know, if you really like that, that sort of aspect of building things there's a lot you can do in that direction particularly around improving your coding skills, things like that. If you really enjoy the analysis part, the statistical and mathematical side of things, there's a lot more you can do there in terms of coming up to speed with new machine learning techniques, learning statistics more in depth. And then finally, on the communications and strategies side there obviously a lot that you can do to improve your communication still, skills. Understand how to abstract from the details, how to abstract out the high level issues that are important to a company. So the first piece of advice I would really give anyone looking to move into this field, is to understand which of those 3 areas you enjoy the most or what the right mix is for you. And make sure you're focusing on the skills that, that will help you get ahead in the particular direction that's of the best fit for you.

Assignment 3

So, that's gradient descent. As we mentioned, gradient descent is just one machine learning technique and one of the most basic. There are a number of sophisticated machine learning algorithms for predicting and classifying our data points. And if this is interesting to you, I'd really recommend you to go out there and read more about machine learning, and really dive into some of those algorithms. Let's talk about Assignment 3. In this assignment, you'll use methods we learned during this lesson. So the t-test and linear regression using gradient descent, to make some conclusions and predictions about our data. First you'll use the t-test in order to compare various subsets of our subway data to try and make some conclusions about subway ridership. For example, do people ride the subway more if it's raining? Or maybe they ride more if it's the weekend. Then we'll use linear regression to try and build a model that predicts how many total riders the New York City subway will have. On a particular day and time, given a variety of factors. For example, the time of day, whether or not it's a weekday, how the weather is, et cetera.



Lesson 3 Recap

In this lesson, we are discussing basics, statistics and machine learning methods that you can use to analyze your data. Not only can we draw conclusions that are statistically sound about existing

data, we can also make predictions about the data that we might collect in the future. During this lesson, we have chosen to really zero in on a couple of methods. Specifically Welch's T-test and linear regression with gradient descent, but as you can imagine, this is just the tip of the iceberg. There are tons of statistics and machine learning techniques out there. If you're interested, I would really encourage you to go out there and read more about those stats and machine learning. Now that you're familiar with these statistics and machine learning techniques, you'll use them to make predictions about subway ridership in New York and also to draw some conclusions about the existing subway data that we have. Once you have made these conclusions and predictions you might find that you'd really love to communicate them to other people. Be they your friends, or your teachers, or your family. One of the most effective ways to do that is through data visualization. That's what we'll cover in the next lesson.