# [**Cassie Kozyrkov**](https://hackernoon.com/@kozyrkov), one of the worlds best statisticians and neuroscientists; had the best explanation that I’ve ever heard when she described ML as simply a thing labeler. Simply a way of Labelling Stuff using Examples! @Kozyrkov on twitter.

# She explains that [machine learning](http://bit.ly/quaesita_emperor) is not magic. Supervised machine learning with classification in particular is nothing more than a [**thing-labeler**](http://bit.ly/quaesita_slkid). Something that takes your description of something and tells you what [label](http://bit.ly/quaesita_island) that thing should get. If we explained Machine Learning this way - it would come across as boring but that is basically what it is… a thing labeler.

# Here is the thing though, even though at its core… we are simply talking about mathematics… It is incredibly powerful, but just not really dystopian science fiction.

# People often use the term AI and Machine Learning interchangeably even though technically there is a difference. Academics argue over what “label” AI, ML, DL, and RL should get - the business community typically does not care. They care about solutions to business problems. And if we can have an automatic thing labeler - and that solves business problems… then that is where the interest and investments will go. Today’s AI is not synthetically intelligent robots like Data from star trek or Arnold Shwartzeneggars charactor on the terminator series. It is more about assisting humanity. And as humans, as a species, we are conditioned to sees human traits in everything. We assume our dogs “love” us when their tail wags or they are “sad” when they can’t get in the car with us and go some place. We see shapes and bodies in clouds. We see faces in our toast, and as Cassie pointed out, if I sew two buttons onto a sock, I might end up talking to it like it was a person even though our intellectual brain knows That sock puppet’s are not people... and the point is - neither is AI — it’s important to keep that in mind. AI… or more specifically in this case, ML for Classification use cases is math that allows us to automatically label things. I hope that is not a downer... The real thing is way more powerful.

# This is why you should be excited. What do you see in the photo on your screen?

# 

# What kind of animal is this? Easy, huh? Now tell me what your brain did with those pixels to get that answer.

# You just absorbed a lot of extremely complicated data in through your senses and then something “magical” happened, and your output prediction was you labeled it dog.’ It is the “and then something magical happened” part that we are going to build with these supervised parametric algorithms. Keep this in mind… for you… to label that picture as a dog was easy. So easy that it is hard to appreciate what happened in your brain. Here is the real question though - what if we wanted a computer to do the same task... to classify (label) photos as dog or not-dog?

# Please pay attention here… this is important. Supervised parametric Machine learning is a new programming paradigm. It is a new way of communicating your desires to a computer.

# In the classical programming development and design approach, a programmer would think long and hard about the pixels on the screen and the output labels like maybe dog or not dog that he or she wanted to communicate. They would get a lot of coffee and probably mountain dew if they are coders and look deep inside for inspiration, and finally they would engineer a model. A model is a PhD level word for saying that they would provide a set of instructions that your computer has to follow to turn the pixels in a picture into labels.

# 

# A model is a recipe that a computer uses to turn data into labels. It’s just some code that the machine uses to convert inputs into outputs, and could be handcrafted by a programmer or learned from data by an algorithm.

# But here is the hard part about this approach… think about exactly what those instructions would be. What are you actually going to be doing with those pixels? Can you really express it in terms of if / then statements? Your brain has the benefit of millions of years [of evolution](https://www.youtube.com/watch?v=OcycT1Jwsns) psychology and now it just works without you having to even think about it. Think about why it is so hard for you to explain the concept of time… it is because your brain was “pre-wired” to understand time in a way that gave you an advantage to survive… not necessarily because it is how time actually works. Why is it that you can not even envision what 5 dimensional space looks like? Same reason - it is b/c your brain evolved to understand the world in 3 spatial dimensions and 1 time dimension. Same thing with why it is so hard to program a machine to look at a picture and determine what label to give it. It is because your brain was pre-wired to simply accept the input data, process the data outside of your active awareness, and then give you a prediction as to what it thinks it sees. Because this processing takes place outside of your conscious awareness… in other words… your brain was just pre-wired to understand how to process images… It is extremely difficult to express that in terms that a machine can replicate.

# So here is our solution to that problem… We choose to Explain with examples, not with instructions. This is a revolutionary shift in the way that we have processed information before. We have always provided instructions and told the computer to simply follow those instructions. I will teach you how to provide the machine examples and let the machine learn how to derive patterns and meaning and ultimately labels or classes from the examples that we provide it.

# This is a much better and a radically different approach. You just tell the machine - “look… Here are a bunch of examples of dogs... look at a bunch of examples of not-dogs… Now I need you Mr. Machine to just figure out for yourself if future pictures contain a dog”?

# *That* is really the heart and soul of what this type of machine learning is all about. It is a new and different way of looking at how to program. Now, instead of giving explicit step by step instructions, you are going to program with examples and the machine learning algorithm [finds the patterns](http://bit.ly/quaesita_emperor) in your data and turns them into instructions you couldn’t write yourself. No more engineering solutions… let the machine learn solutions!

# AI allows you to automate what could not even be expressed before.

# So Why is that exciting? This allows us to express our desires to machines in a way that we couldn’t before. We already love to get machines to do stuff for us. But the problem we have run into is… how can we give instructions to the machine if we ourselves can’t even think up what those instructions are because they are really hard to even impossible to think up? If they’re ineffable? If the instructions themselves are indescribable?

# AI and machine learning are about automating the indescribable. They allow you to explain yourself using examples and not with instructions.

# Do not underestimate this… This unlocks a massive set of use cases that expand across every industry that we could not get computers to help us with in the past because we simply could not express the instructions. Now [all of these tasks](http://bit.ly/quaesita_island) are suddenly possible — machine learning represents a fundamental leap in human progress. It is the future and the future is here! There is a massive race across all businesses to discover and exploit these use cases.

# 

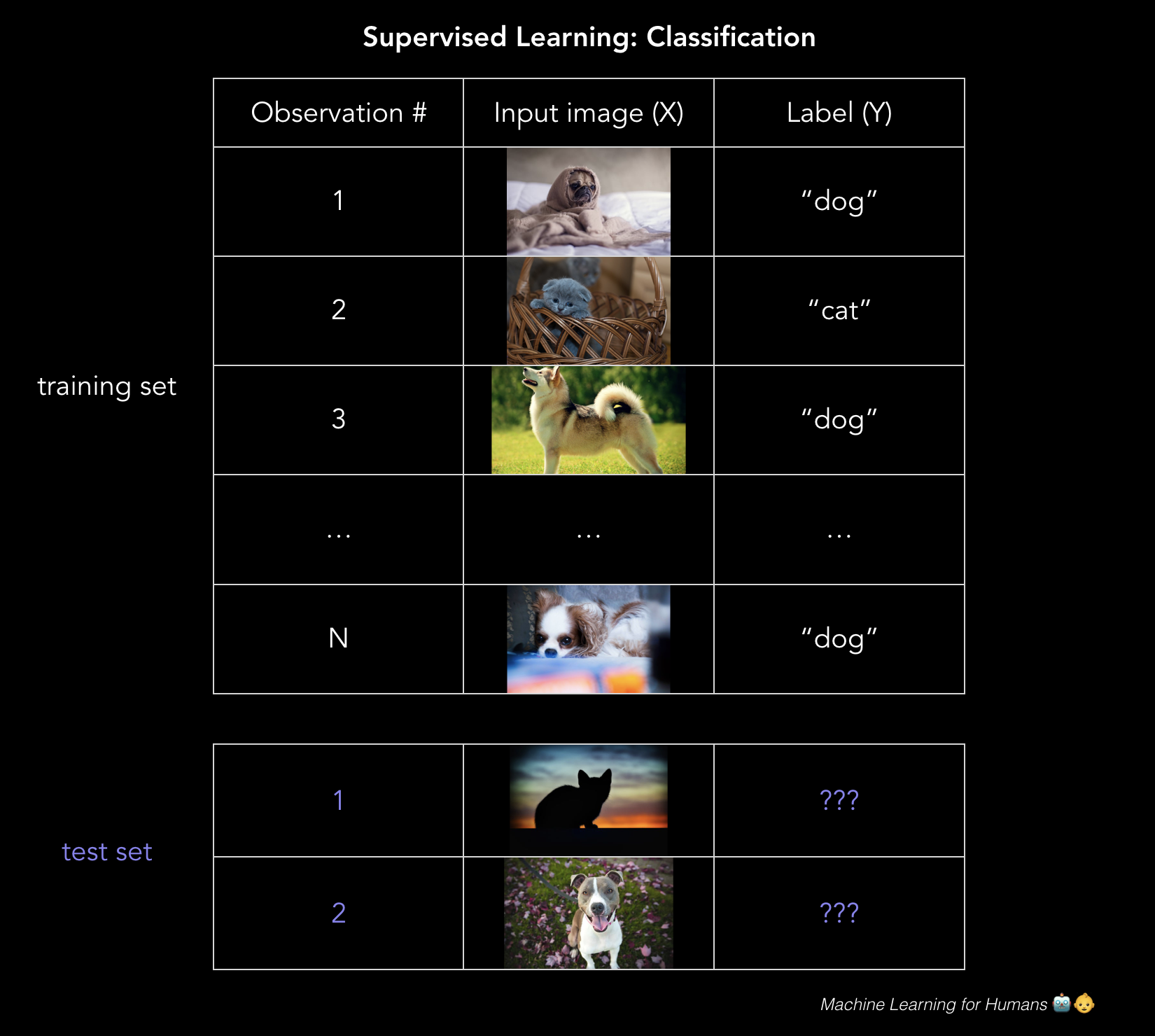
# **--------------------**

# **Classification is all about predicting a label.**

*If spam is a label… The question is… Is this email spam or not? If good borrower or bad borrower is the label then the question becomes, Is that borrower going to repay their loan? If I show you a picture… can you tell me if the picture has a cat… or a dog… or a horse… or a person… or a person that looks like a horse or whatever. If I am looking to sell ad space … think Social Media companies… Will those users click on the ad or not? Who is that person in your Facebook picture?*

Classification **is all about** predicting a **discrete target label Y. It is about trying to determine WHAT bucket a thing belongs to.** Classification is the problem of assigning new observations to the **class** to which they most likely belong to. They do this based on a machine learning classification model that is built from labeled training data. In other words… will a borrower repay the loan is one class and borrowers that will not repay a loan is another class.

The accuracy of your classifications will depend exclusively on the effectiveness of the algorithm you choose, how you apply the algorithm, and how much useful training data you have.



# **---------------------**

# **Logistic regression is about making a call of 0 or 1? Or 0 or 1 or 2 etc.**

Logistic regression is a method of **classification**. Instead of predicting a real value, the model predicts the probability of a **categorical** target variable Y belonging to a certain class. In other words, I am 95% certain that the image contains a dog. I am 80 percent certain the customer will repay the loan.

Most people are familiar with Equifax, Experian and Transunion being the providers of credit scores. Your credit score is a function of the probability that you will be a good steward of credit. This is a good example of classification ... determining whether a loan applicant is credit worthy.

Ultimately, the lender wants to know whether they should give you… the borrower a loan or not, and they have some tolerance for risk that the applicant is in fact credit worthy or not based on credit history and some other features. In this case, the goal of logistic regression is to calculate the probability (between 0% and 100%) that the application is credit worthy. We can use logistic regression to determine whether to even make the loan. We can say that we will move forward with making the loan if the bank is more than 90% certain that the borrower will repay the loan with interest. With these probabilities, we can set some threshold above which we’re willing to lend to the borrower, and below which we deny their loan application or flag the application for further review. In my case - I said 80% we offer the loan… 70% we flag for further review… and 69 percent and below we decline. Something like that...

Though logistic regression is often used for **binary classification** where there are two classes, keep in mind that classification can be performed with any number of categories (Think about when assigning handwritten digits a label between 0 and 9, or using facial recognition to detect which friends are in a Facebook picture).

**Remember OLS? Can you just use ordinary least squares?**

Not really. If you trained a linear regression model on a bunch of examples where the prediction Y = 0 or 1, you might end up predicting some probabilities that are less than 0 or greater than 1, which doesn’t make sense. Instead, we’ll use a logistic regression model which was designed for assigning a probability between 0% and 100% that Y belongs to a certain class.

**\*\*How does the math work?**

*Side Note: the math in this section is interesting but might be on the more on the overkill side. Feel free to skip it if you are more interested in the high-level concepts.*

The logistic regression model is a modification of linear regression that makes sure to output a probability between 0 and 1 by applying the **sigmoid function,** which, when graphed, looks like the characteristic S-shaped curve that you’ll see a bit later.



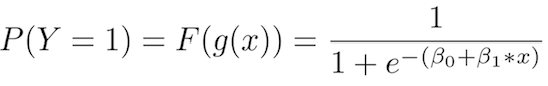
Here is the Sigmoid function, which squashes values between 0 and 1 so we can have a probability of a classification.

Recall the original form of our simple linear regression model, which we’ll now call g(x) since we’re going to use it within a compound function:



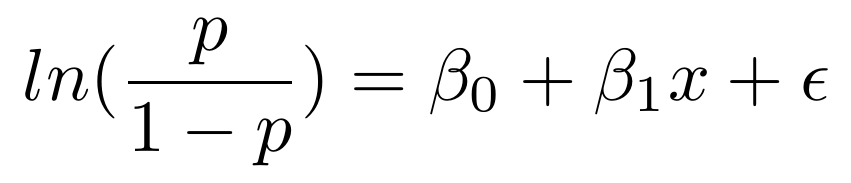
So the first problem that arises is that we may end up with model outputs that are greater than one or less than zero which makes no sense if were looking for percentages between 0 and 1. So, we are going to define a *new* function F(g(x)) that transforms g(x) - which you will recall is the linear regression algo… by SQUASHING THE OUTPUT OF LINEAR REGRESSION in to a value in the [0,1] range. I just named the function that does this… The Sigmoid function.

So we plug g(x) - the linear regression function into the sigmoid function on your screen which results in a function of our original function that outputs a probability between 0 and 1:



In other words, we’re calculating the probability that the training example belongs to a certain class: P(Y=1).

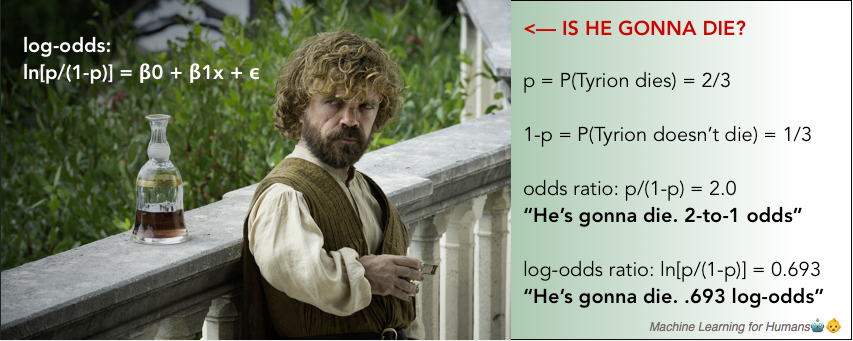
Here we have isolated *p,* the probability that Y=1, on the left side of the equation. If we want to solve for a nice clean β0 + β1x + ϵpsilon - on the right side so we can simply and straightforwardly interpret the beta coefficients we’re going to learn, we’d instead end up with the **log-odds ratio**, or **logit**, on the left side — hence the name “logit model”:



The log-odds ratio is simply the natural log of the **odds ratio, p/(1-p),** which crops up in everyday conversations:

*“Yo, what do you think are the* ***odds*** *that Ernesto goes to the Virginia Tech Hokies / Miami Hurricanes game in the 2020 NCAAF Season?*

*“Hmm. It’s definitely at least twice as likely to happen than not.* ***2-to-1 odds.*** *Especially since he didn’t go in 2019!”*

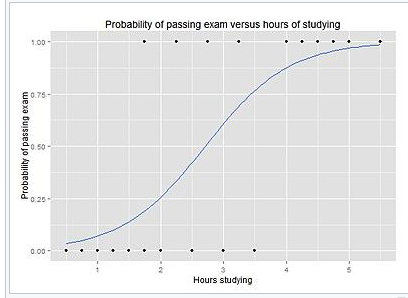
**

Note that in the logit model, β1 now represents the **rate of change in the log-odds ratio** as X changes. In other words, it’s the “slope of log-odds”, not the “slope of the probability”.

Log-odds are defnitely not intuitive but it is still worth understanding since it will come up again when you’re interpreting the output of neural networks performing classification tasks.

**\*\*\*Let’s Use the output of a logistic regression model to make decisions.**

The output of the logistic regression model that you see on your screen clearly looks like an S-curve. It is showing P(Y=1) or the probability that the phenomenon is in classification 1 based on the value of X:



Source: [Wikipedia](https://en.wikipedia.org/wiki/Logistic_regression)

To predict the Y label — spam/not spam, cancer/not cancer, fraud/not fraud, etc. — you have to set a probability cutoff, or a **threshold,** for a positive result. For example: *“If our model thinks the probability of this email being spam is higher than 70%, label it spam. Otherwise, don’t.”*

The threshold depends on your tolerance for **false positives** vs. **false negatives**. As you can see from wikipedia - In binary classification, a false positive is an error in data reporting in which a test result improperly indicates presence of a condition, such as a disease, when in reality it is not present. A false negative is an error in which a test result improperly indicates no presence of a condition, when in reality it is present. Imagine how important this is if you are using healthcare for life or death decisions. Or if you are creating an auto-pilot. If you’re diagnosing cancer, you would want to have a very very low tolerance for false negatives (or errors in which your test results improperly indicate the presence of a condition that does not exist when in reality… it is present). The truth is even if there’s a very small chance the patient has cancer, you’d want to run further tests to make sure. So you’d set a very low threshold for a positive result.

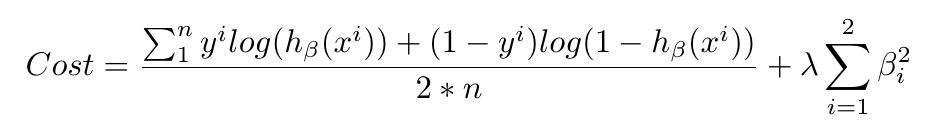
If you are in the financial sector and you are using AI to assist in the processing of loan application then your tolerance for false positives might be higher- especially for smaller loans, since further vetting is costly and a small loan might not be worth the additional operational costs and friction for non-fraudulent applicants who are flagged for further processing.

-------------------------------

## **Minimizing loss with logistic regression**

Just like with linear regression, we use gradient descent to learn the beta parameters that minimize loss. What are the optimal values for the coefficients or weights?

In logistic regression, the cost function is basically a measure of how often you predicted 1 when the true answer was 0, or vice versa. On your screen is a **regularized cost function** just like the one we went over for linear regression.



Don’t panic when you see a long equation like this! Break it into chunks and think about what’s going on in each part conceptually. Then the specifics will start to make sense.

The first side is a measure of the **data loss**. In other words, how much discrepancy is there between what the model predicted and reality. The side is the **regularization loss** - or how much we penalize the model for having large parameters that heavily weight certain features (remember, this prevents overfitting).

We’ll minimize this cost function with gradient descent, as above, and *boom*! we’ve built ourselves a logistic regression model to make class predictions as accurately as possible.

--------------------------

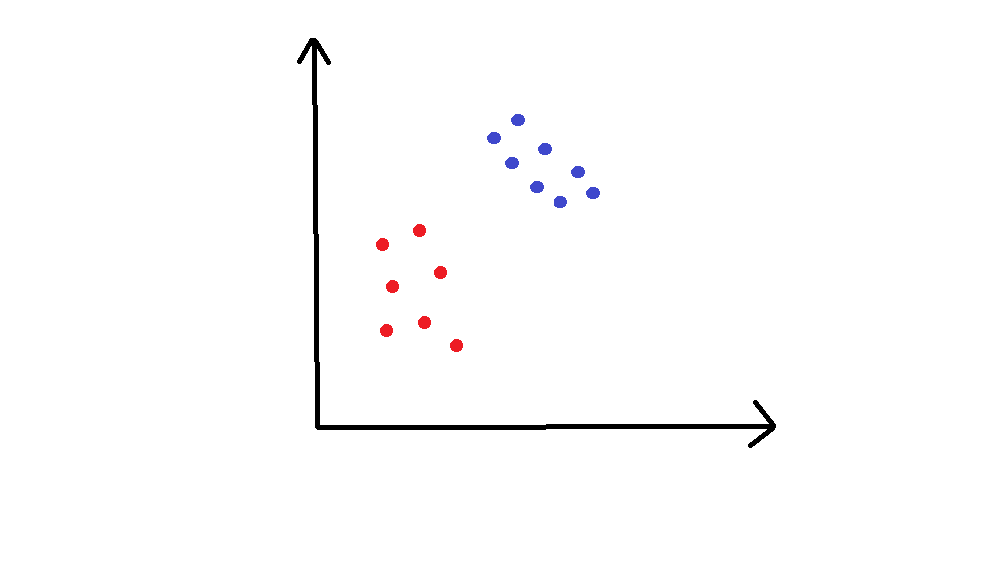
# **Support vector machines (SVMs)**

SVM is the last parametric model we’ll cover. It typically solves the same problem as logistic regression — classification with two classes — and yields similar performance. It’s worth understanding because the algorithm is geometrically motivated in nature, rather than being driven by probabilistic thinking.

A few examples of the problems SVMs can solve:

* Is this an image of a cat or a dog?
* Is this review positive or negative?
* Are the dots in the 2D plane red or blue?

We’ll use the third example to illustrate how SVMs work. Problems like these are called **toy problems** because they’re not real — but if you have seen the matrix then you already know that nothing is real… so cool.

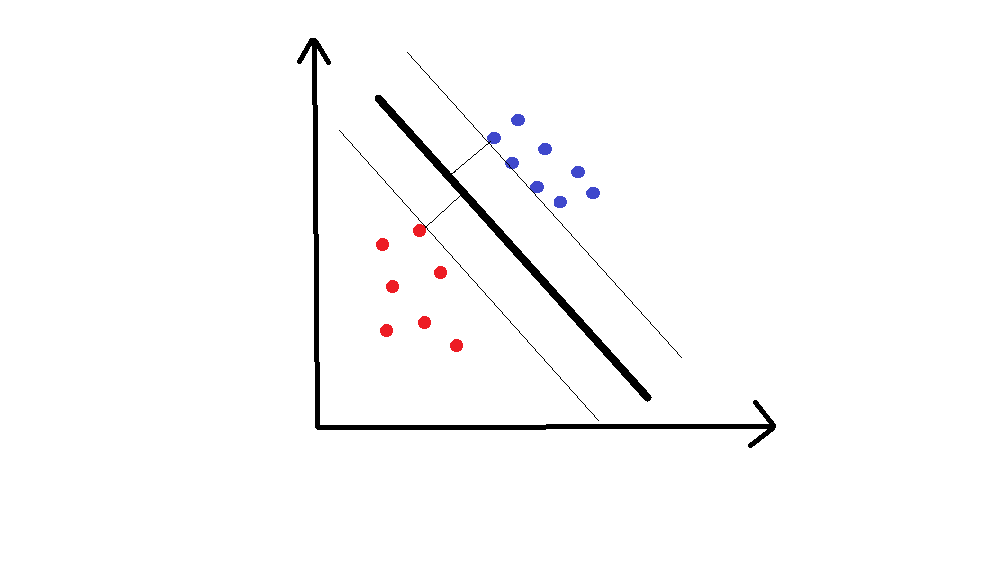


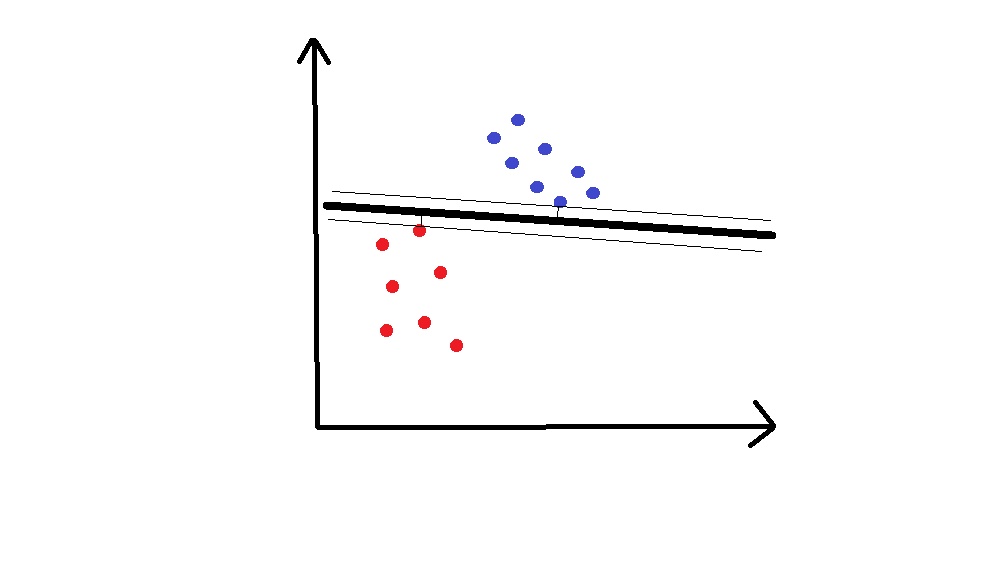
In this example, we have points in a 2D space that are either orange or grey, and we’d like to cleanly separate the two.

The training set is plotted in the graph above. We would like to classify new, unclassified points in this plane. To do this, SVMs use a separating line (or, in more than two dimensions, a multi-dimensional **hyperplane**) to split the space into a red zone and a blue zone. You can already imagine how a separating line might look in the graph above.

How, specifically, do we choose where to draw the line?

On the screen are two examples of such a line:





Hopefully, you share the opinion that the first line is superior. The distance to the nearest point on either side of the line is called the **margin**, and SVM tries to **maximize the margin**. You can think about it like a safety space: the bigger that space, the less likely that noisy points get misclassified.

Based on this short explanation, a few big questions come up.

**1. How does the math behind this work?**

We want to find the optimal hyperplane (a line, in our 2D example). This hyperplane needs to (1) separate the data cleanly, with orange points on one side of the line and grey points on the other side, and (2) the hyperplane needs to maximize the margin. This makes it an **optimization** problem. The solution has to respect constraint (1) which is to cleanly separate the data while maximizing the margin as is required in (2).

The human version of solving this problem would be to take a ruler and keep trying different lines separating all the points until you get the one that maximizes the margin.

It turns out there’s a clean mathematical way to do this maximization, but the specifics of the math are beyond our scope in this episode.

The solution hyperplane you end up with is defined in relation to its position with respect to certain x\_i’s, which are called the **support vectors**, and they’re usually the ones closest to the hyperplane.

**2. What happens if you can’t separate the data cleanly?**

There are two methods for dealing with this problem.

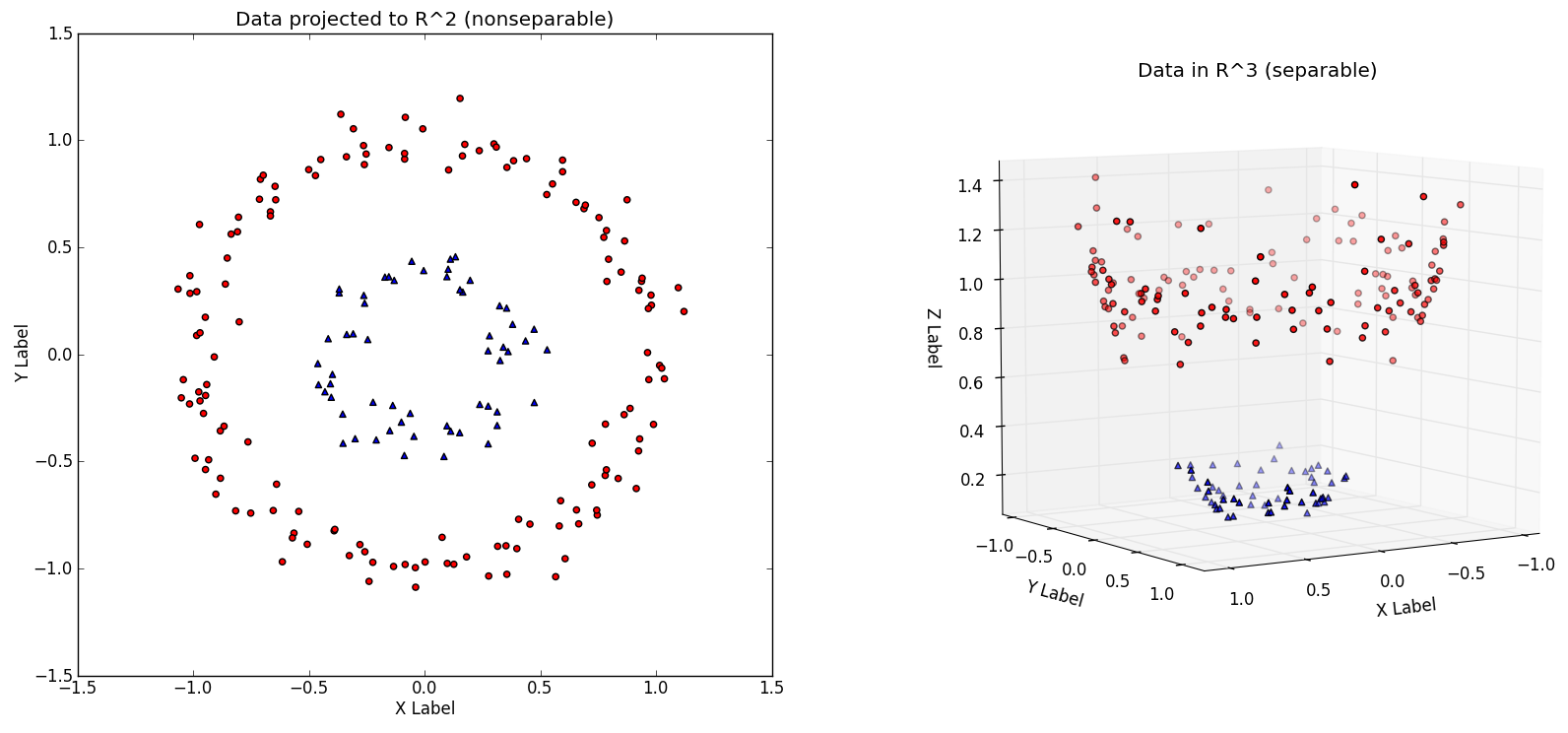
*2.1. Soften the definition of “separate”.*

We allow a few mistakes, meaning we allow some orange points in the grey zone or some gray points in the orange zone. We do that by adding a cost C for misclassified examples in our loss function. Basically, we say it’s acceptable but costly to misclassify a point.

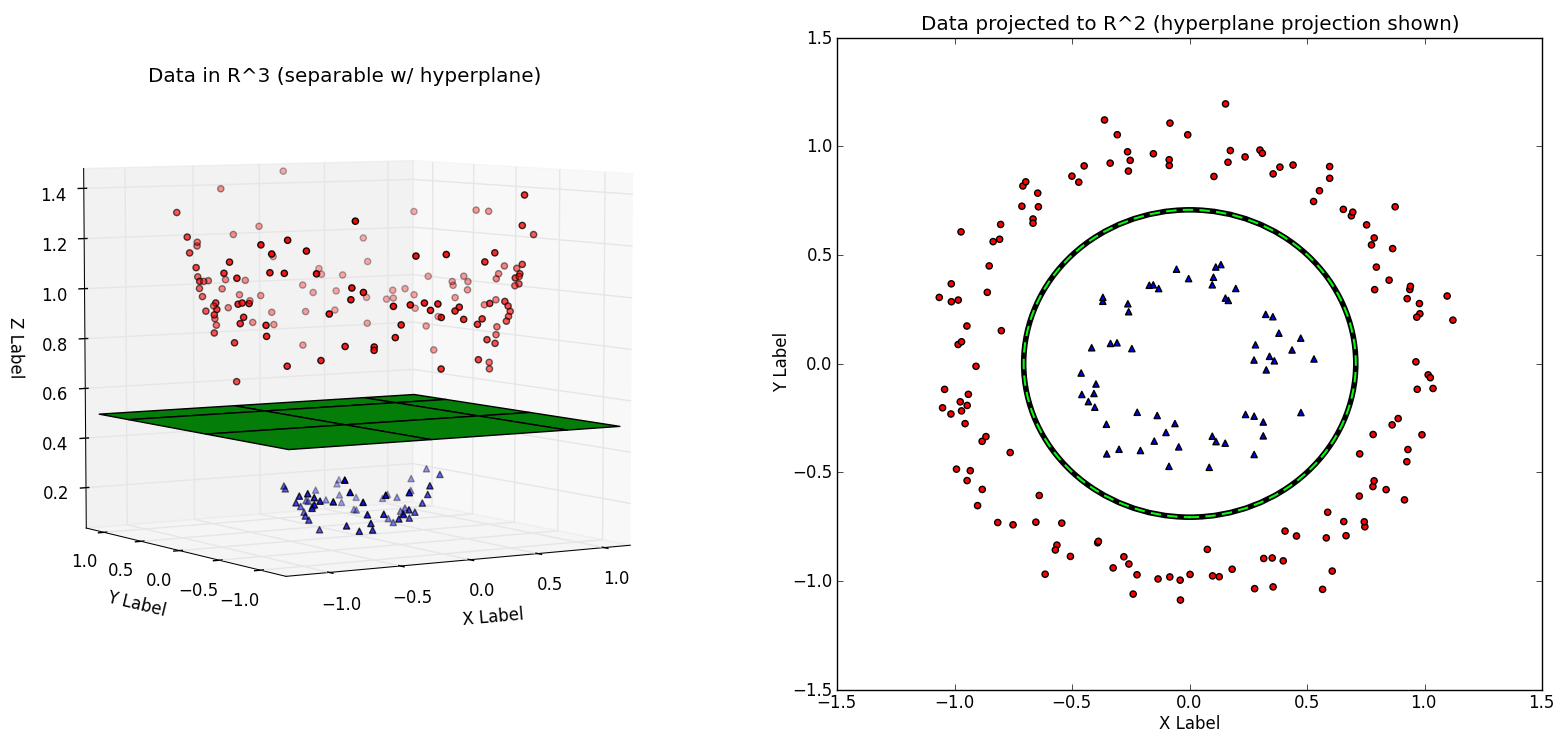
*\*\*Here is where the power of SVMs come to light… we can process data in higher dimensions.*

We can create nonlinear classifiers by increasing the number of dimensions or features, which is to say that we can include x², x³, even cos(x), etc. Suddenly, you have boundaries that can look more wiggly and squiggly when we bring them back to the lower dimensional representation.

Here is the best way to think about Support Vector Machines, imagine that you have orange and grey marbles all laying on the ground. They appear to be mixed in a way that they can not be cleanly separated by a line — but if you could make all the orange marbles levitate off the ground by projecting them into a higher dimension, then you could draw a plane separating them. Then you let them fall back to the ground knowing exactly where the grey marbles stop and orange marbles begin.



On the screen, you see A nonseparable dataset in a two-dimensional space that we’re calling R squared. And then the exact same dataset mapped onto three dimensions with the third dimension being x²+y²



The decision boundary is shown in green, first in the three-dimensional space (left), then back in the two-dimensional space (right). Same source as previous image.

In summary, SVMs are used for classification with two classes. They attempt to find a plane that separates the two classes cleanly. When this isn’t possible, we either soften the definition of “separate,” or we throw the data into higher dimensions so that we *can* cleanly separate the data.

--------------------------------

# **Success!**

In this section we covered:

* The **classification** task of supervised learning
* Two foundational classification methods: **logistic regression** and **support vector machines (SVMs)**
* Recurring concepts: the **sigmoid** function, **log-odds (“logit”)**, and **false positives** vs. **false negatives**,

In [Part 2.3: Supervised Learning III](https://medium.com/@v_maini/supervised-learning-3-b1551b9c4930), we’ll go into **non-parametric** supervised learning, where the ideas behind the algorithms are very intuitive and performance is excellent for certain kinds of problems, but the models can be harder to interpret.

# **Practice materials & further reading**

## **2.2a — Logistic regression**

*Data School has an excellent* [*in-depth guide to logistic regression*](http://www.dataschool.io/guide-to-logistic-regression/)*. We’ll also continue to refer you to* [*An Introduction to Statistical Learning*](http://www-bcf.usc.edu/~gareth/ISL/)*. See Chapter 4 on logistic regression, and Chapter 9 on support vector machines.*

*To implement logistic regression, we recommend working on* [*this problem set*](https://datahack.analyticsvidhya.com/contest/practice-problem-1/)*. You have to register on the site to work through it, unfortunately. C’est la vie.*

## **2.2b—Down the SVM rabbit hole**

*To dig into the math behind SVMs, watch Prof. Patrick Winston’s* [*lecture*](https://www.youtube.com/watch?v=_PwhiWxHK8o) *from MIT 6.034: Artificial Intelligence. And check out* [*this tutorial*](https://pythonprogramming.net/svm-in-python-machine-learning-tutorial/) *to work through a Python implementation.*