MARCH 19, 2018

Introduction to Machine Learning

Source: Dilbert

Machine Learning is about making predictions. This post will give an introduction to Machine Learning through a problem that most businesses face: **predicting customer churn**.

ML can help predict which of your customers are at risk for leaving in advance, and give you an edge by pre-empting with action.

Introduction

Machine Learning can best be understood through four progressive lenses.

1. *The Broad*: Machine Learning is the process of predicting things, usually based on what they've done in the past.
2. *The Practical*: Machine Learning tries to find relationships in your data that can help you predict what will happen next.
3. *The Technical*: Machine Learning uses statistical methods to predict the value of a target variable using a set of input data.
4. *The Mathematical*: Machine Learning attempts to predict the value of a variable Y given an input of feature set X.

Machine Learning allows us to accurately predict things using simple statistical methods, algorithms, and modern computing power.

Churn Example: Machine Learning will help us understand why customers churn and when. Our input values might include:

- How frequently a user engages with the product
- What their engagements are
- How interspersed their activity is

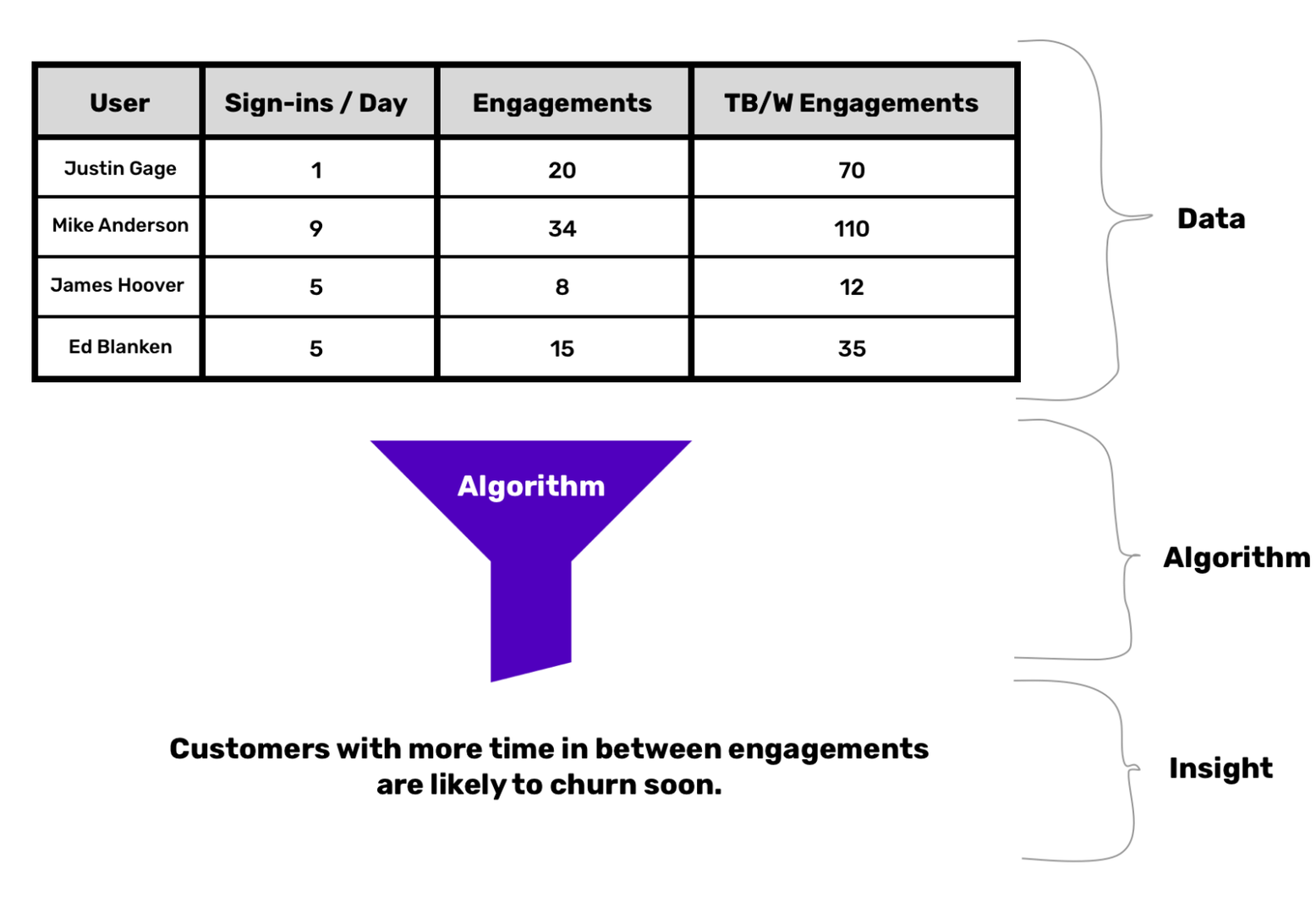
Our assumption is that these data points can reveal something fundamental about the customer.

What It Is – The Data/Algorithm Framework

Any type of Machine Learning can be broken down into 2 major parts:

1. *The data*
2. *The algorithm*

Any other complexities that you might hear thrown around—deep learning, gradient descent, reinforcement learning—are just variations on these two fundamental pieces. If you ever get confused or lost by all the terms floating, just ask yourself whether it has to do with your data or your algorithm. Everything else is commentary.



User	Sign-ins / Day	Engagements	TB/W Engagements
Justin Gage	1	20	70
Mike Anderson	9	34	110
James Hoover	5	8	12
Ed Blanken	5	15	35

Your Data

The data part of Machine Learning is the simplest – it's whatever you're trying to predict, and how you plan on teaching the computer about it.

Churn Example: the **data** might be past user interactions. Data is typically organized as rows, with columns representing features of our data (see diagram above).

What Machine Learning methods will do is try to find *patterns* in this data. We might find a few different kinds of relationships:

- Users with long times between engagements are likely to churn soon
- Users with a high number of engagements are unlikely to churn

These are made up, but any number of real relationships can actually exists within our data. Our machines need this data to sift through and *find* them.

Your Algorithm

In Machine Learning, the *algorithm* is just the method that you'll use to find those relationships within your data. Algorithms can be complex or simple, big or small, or any permutation of things: but at the core, they're just ways of figuring out what, if anything, drives the changes you're trying to predict. Heard of Deep Learning? That's (basically) a type of algorithm. Much like the MergeSort algorithm is efficient at sorting arrays, Machine Learning algorithms are efficient at surfacing relationships and associations.

Different types of algorithms can help you achieve different goals. If you want to be able to explain the relationships that you find in human speak, a simple algorithm like Linear Regression is probably a good bet. If you care the most about accuracy (and explainability isn't too important) neural nets can achieve higher accuracy rates. This is often called the accuracy-explainability tradeoff, and it's an important product decision that many data scientists need to make.

Anything else you encounter in the world of ML has to do with one of these two things. Feature scaling? Modifying your data. Deep learning? A type of algorithm. Cross validation? A way to improve your algorithm. Take this framework to the bank.

How it Works

Believe it or not, the concepts behind *how* Machine Learning works are actually very simple, even if the underlying algorithms can get complex. The majority of ML algorithms use one method to find those relationships we've been talking about, and it's literally groping around in the dark. It's technically called Gradient Descent, but the method is simple – we start at some random point, and try to improve our predictions.

We start out with a Cost Function – a way to measure how well we're doing in finding these relationships. In other words, it helps us quantify how far off our predictions are so we can work on improving them. The cost function differs depending on what algorithm you're using. For Linear Regression, it's the following:

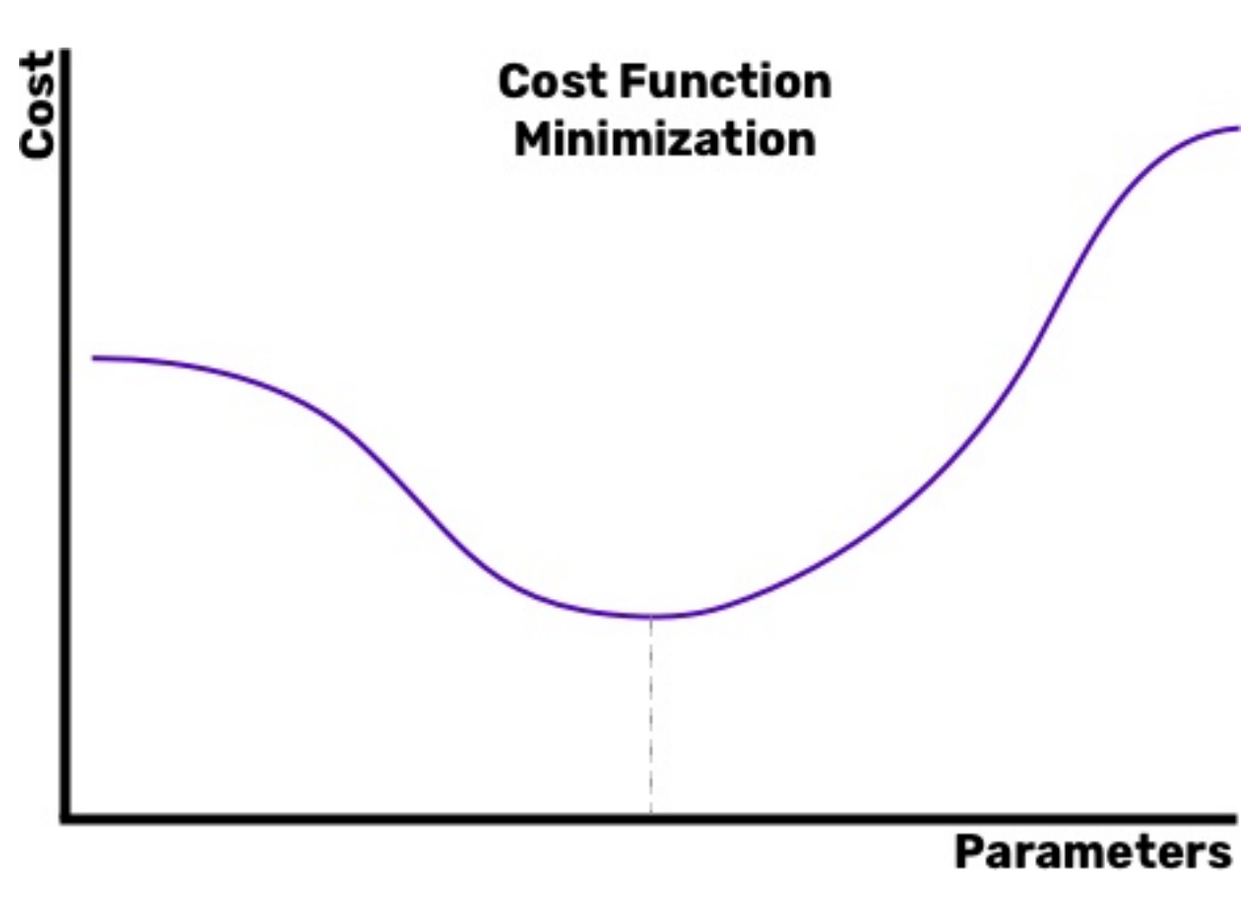
Cost Function

$$\text{Cost} = \frac{1}{2m}\sum^{m}\left(y_{\text{Predicted}} - y_{\text{Actual}}\right)^2$$

m	number of examples
$y_{\text{Predicted}} - y_{\text{Actual}}$	the difference between the real value and our predicted value

This might look complicated at first, but just focus on the cost function. It's quantifying the difference between what our algorithm is predicting – $h_\Theta(x^{(i)})$ – and what the actual value of our target variable is, or $y^{(i)}$. Our goal should be to minimize this cost function, because we want our predicted values to be as close as possible to the actual target variable values.

So how do we *actually do this*? Well, we start with a random set of predictions and try to improve from there. If our predictions come in too high, we'll adjust. If they're too low we'll adjust too. The algorithm basically moves around in the dark slowly until it finds what it's looking for. The funny thing is that Gradient Descent is actually an algorithm in of itself – we're using an algorithm to improve our algorithm. Pretty cool! For more details on how the Gradient Descent algorithm is implemented (in addition to the other solutions for optimizing our cost function), check out our Introduction to Optimizers.



To reiterate how all of these pieces fit into the puzzle: we use Gradient Descent to optimize our cost function, which is how our algorithm finds the underlying relationship in our data. We do that by starting with some random predictions, and slowly calibrating our approach until we get as close as possible to the actual values we're looking for.

Supervised and Unsupervised Learning

There are two major types of Machine Learning in practice that use different data-algorithm relationships: supervised learning and unsupervised learning. The distinction between these two is subtle but important. The major difference lies in *how* you present your data to the algorithm.

In *supervised learning*, you define the possible outcomes. In our weather example, we may say that a given day can be hot, cold, or moderate. We then pass our data into the algorithm, and it will figure out what (if anything) leads to hot days, what leads to cold days, and what leads to moderate days. Since we defined what we think the options are – hot, cold, and moderate – the algorithm will spit out predictions within that framework.

But sometimes, we don't really know what the options are or what we want them to be. *Unsupervised learning* takes the data we have and *tries to figure out itself* what the different potential groupings are. An unsupervised learning algorithm might find that the groups with the most distinct features are days that are very hot, moderate, and cloudy. Instead of deciding the groups of outcomes in advance and trying to map relationships to them, we let the algorithm find the ones it feels are the most natural.

Essentially, the difference between these two types of Machine Learning is the input output model. In supervised learning, you provide the algorithm with X and Y, and try to figure out the relationships between the two. In unsupervised learning, there is no Y – you're just trying to understand the underlying organization of the data in of itself.

Machine Learning in Practice

Practically, it's pretty easy to get some rudimentary Machine Learning done on a dataset that you're interested in exploring. There are toolsets across the spectrum – for people that have never written a line of code before, all the way to Machine Learning Engineers seeking to optimize every parameter perfectly.

Software Tools

Some tools let you run Machine Learning on a dataset without writing any code. If you're a subject matter expert without any programming experience, these tools can extremely helpful in bringing ML into whatever you're looking to build.

- Dataiku DSS
- Pienso
- Google AutoML
- Amazon Sagemaker

Popular Frameworks / Packages

Frameworks aggregate core functions in a programming language into something that's easier to use and more powerful. Practically, instead of having to write out the math and statistics of your chosen algorithm manually (some people prefer this!), you'll be able to implement it in your favorite programming language pretty simply.

In fact, using Python's popular scikit-learn framework, we can train a Support Vector Machine (type of algorithm) on a set of data pretty quickly.

```
#Import the support vector machine module from the sklearn framework

from sklearn import svm

#Label x and y variables from our dataset

x = ourData.features

y = ourData.labels

#Initialize our algorithm

classifier = svm.SVC()

#Fit model to our data

classifier.fit(x,y)
```

Scikit-learn is one of many packages that developers use to make Machine Learning easier and more powerful. Some other popular ones include:

- TensorFlow
- Caffe (for deep learning)
- MLlib (for big data)
- Torch (GPU first)

Online Courses and Videos

Thankfully, in 2018 there are a number of excellent online resources that can help you get up and running with Machine Learning in no time.

- Andrew Ng's Coursera course is the standard here, and does an excellent job of explaining the math and theory behind traditional Machine Learning
- For a more practical and code based approach, try Machine Learning Mastery's guide
- For Deep Learning, the fast.ai course

Books

Hands-on Machine Learning with Scikit-Learn and Tensorflow (O'Reilly) – "*Through a series of recent breakthroughs, deep learning has boosted the entire field of machine learning. Now, even programmers who know close to nothing about this technology can use simple, efficient tools to implement programs capable of learning from data. This practical book shows you how.*"

Real World Machine Learning (Manning) – "*Real-World Machine Learning is a practical guide designed to teach working developers the art of ML project execution. Without overdosing you on academic theory and complex mathematics, it introduces the day-to-day practice of machine learning, preparing you to successfully build and deploy powerful ML systems*"

Programming Collective Intelligence (O'Reilly) – "*Want to tap the power behind search rankings, product recommendations, social bookmarking, and online matchmaking? This fascinating book demonstrates how you can build Web 2.0 applications to mine the enormous amount of data created by people on the Internet. With the sophisticated algorithms in this book, you can write smart programs to access interesting datasets from other web sites, collect data from users of your own applications, and analyze and understand the data once you've found it.* "

An Introduction to Statistical Learning (Springer Texts in Statistics) – "*An Introduction to Statistical Learning provides an accessible overview of the field of statistical learning, an essential toolset for making sense of the vast and complex data sets that have emerged in fields ranging from biology to finance to marketing to astrophysics in the past twenty years. This book presents some of the most important modeling and prediction techniques, along with relevant applications.*"

Tutorials

Your First Machine Learning Project in Python Step-By-Step (Jason Brownlee) – "*Do you want to do machine learning using Python, but you're having trouble getting started? In this post, you will complete your first machine learning project using Python. If you are a machine learning beginner and looking to finally get started using Python, this tutorial was designed for you.*"

Python Machine Learning: Scikit-Learn Tutorial (Datacamp) – "*Machine learning is a branch in computer science that studies the design of algorithms that can learn. Typical tasks are concept learning, function learning or "predictive modeling", clustering and finding predictive patterns. These tasks are learned through available data that were observed through experiences or instructions, for example.*"

Machine Learning in Python: A Tutorial (Dataquest) – "*In this tutorial, we'll guide you through the basic principles of machine learning, and how to get started with machine learning with Python. Luckily for us, Python has an amazing ecosystem of libraries that make machine learning easy to get started with. We'll be using the excellent Scikit-learn, Pandas, and Matplotlib libraries in this tutorial.*"

Machine Learning in R for Beginners (Datacamp) – "*This small tutorial is meant to introduce you to the basics of machine learning in R: more specifically, it will show you how to use R to work with the well-known machine learning algorithm called "KNN" or k-nearest neighbors.*"

Machine Learning on Algorithmia

- Parsey McParseface – parse sentences with ease
- Text Similarity – find the most similar text files within a collection of documents
- Geographic Spectral Clustering
- Nudity Detection – detect nudity in pictures
- Illustration Tagger – automatically tag your images

Algorithmia
More Posts - Website
Follow Me: