**Machine Learning**

Alternative title: “I, For One, Welcome Our New Computer Overlords”

1. **Introduction** *(5 minutes)* 
   1. Purpose of this talk
      1. To demystify some of the fundamentals of machine learning
      2. To give a bird’s eye overview of the topic
      3. To think outside the virtual box: show dozens of practical applications of machine learning in everyday life (that might be taken for granted) and how TWDC could be using those services
   2. Assumptions about basic math for this lecture: None! I’ll try to explain the underlying concepts of machine learning with as little mathematics as possible
      1. This is not hard because: learning is something we do every day. Conceptually, these ideas are very relatable.
   3. Format of the lecture:
      1. The Learning Problem (a bird’s eye view)
      2. The Building Blocks of Machine Learning (Generalization)
      3. Machines aren’t that Scary: What you can learn (specific examples)
      4. Conclusion
   4. To start:
      1. The classic example: Spam filters
         1. How do computers block certain types of emails as spam, but they leave others perfectly intact?
         2. Assumption: the computer looks for words that normally are spam, and it just puts those emails wholesale in the spam filter
         3. But wouldn’t that be really inaccurate for different users? (Answer: yes, very inaccurate)
         4. Furthermore: how did the computer know what’s spam and what isn’t spam in the first place? (it’s not like anyone hard-coded it to know)
         5. Answer: The machine is taught to **learn** constantly what users think is spam/not spam, and it
2. **The Learning Problem**: what is machine learning, anyway? *(10 minutes)*
   1. **The problem** with defining machine learning:
      1. Machine learning combines **dozens** of fields of study **(word-cloud)**. Set theory, regression analysis, vector analysis, probability theory, data mining, information theory, computer science, A.I., database management, Natural Language Processing, distributed computing, decision theory, Latent Semantic Analysis, Game Theory, neuroscience, buzzwords
      2. Machine learning requires **three large tasks**: **Conceptualization, Theory, and Application** (programming)
      3. Machine learning is a ***new field***
      4. Machine learning has **MANY tradeoffs**.
   2. Definitions of learning:
      1. “Field of study that gives computers the ability to learn without being explicitly programmed” –**Arthur Samuel**, 1959
         1. Samuel: wrote the first self-learning checkers program, the **Samuel Checkers-playing Program**,
         2. He just measured the chance of winning based on any position of the board.
         3. This is before we solved checkers— where Game Theory intersects Machine Learning
      2. “*Given a function m, where x is the input and y is the output, each time an experience E happens, for every time function m runs successively it performs at a rate P higher than before.*” – adaptation of Tom M. Mitchell’s definition
      3. Intuitively, there are some steps to the learning process
         1. You *observe* an event happening a number of times
         2. You *classify* that event with a bunch of other events
         3. You *trust* that every time you come across something similar to your other event, you know what to do with it.
      4. **Dispense the philosophy:** The underlying process isn’t “Can machines think?” Rather, it’s “Can machines do what we can do as well as we can do it?” –The Great Alan Turing
   3. When to use machine learning: (Prof. Yaser S. Abu-Mostafa, Caltech)
      1. A pattern exists.
      2. We cannot pin it down mathematically.
      3. We have data on it.
      4. We don’t want to (or can’t ) find the pattern ourselves. (my own)
3. **The Building Blocks of Learning** (*10 minutes*)
   1. Types of learning (the basic ones):
      1. **Supervised:** You’re given a data set that we know the answer for a small subset of the data. Think spam filters. We
      2. **Unsupervised:** You’re given a data set in which we have plenty of properties, but NO answers. Furthermore, we don’t even tell the computer what patterns to find! We just ask it to find patterns
      3. Why would you use one or the other?
         1. Supervised: Large data sets, more accurate, you know what you want to find, you have the manpower to get the answers
         2. Unsupervised: You have NO way of feasibly giving the computer a subset of answers.
   2. The flowchart: its elements (a *little* bit of math)
      1. Unknown function ***F: X -> Y***: There’s some way of categorizing the data out there. There’s a pattern (a function is just a generalization of a pattern). The rule in machine learning is we *don’t* know the pattern… if we knew the pattern, we wouldn’t have to learn anything!
         1. **Example**: Say we’re trying to categorize tweets on Twitter into positive or negative results. Maybe the function is “All people whose names start with J HATED Monsters University.” Maybe it’s “Everyone hated Monsters University before it came out, but now they all love it.” These are *functions.* We just don’t know which one to pick.
         2. We can define the input and output, though. In our example, the input is the data we have in our database. Input: name of the person, what they said online, the metadata, etc. Output: whether they liked the movie or not.
      2. Training Example: ***(x1, y1), (x2, y2), … , (xn, yn)***: We have a small set of data. We might know the answers or not at this stage, but we might not. All we know is that we
         1. **Example:** We might show the computer a list of tweets by different users and *give it the answers*: tell it whether certain tweets are positive or negative.
         2. Dimension: The number of different parameters you have in **xi**. As you increase the number of parameters, finding patterns becomes MUCH more difficult!
      3. The Hypothesis Set: ***H.*** THIS is the difference between machine learning and pattern recognition. In machine learning, we have a set of possible hypotheses, or models **h1, h2, … , hn** that all are ways to solve the problem.
         1. **Example:** The model we choose determines what hypotheses we can choose from. Maybe our model only finds relations between positive/negativity and the length of characters in the tweet. A specific hypothesis would be: “Most twitterers who wrote more than 80 characters liked the movie.”
      4. The Algorithm ***A.***This picks from the set of hypotheses H and decides which one it wants to use.
         1. **Example:** Our algorithm picks a hypothesis and sees how well it works for 5 example tweets. Then it compares it to how well another one works for 5 different tweets.
      5. The final hypothesis **g.** Obviously, we want this to be as close to our function **F** as possible… but we don’t know **F!** So how do we estimate how close we are to the function? How do we know that we truly learned, and we’re not just memorizing the examples?
         1. Answer: we compare the error of our small training examples with the error we get when we use other data. We can approximate this other error by adding the bias and the variance. (Okay, maybe this is getting too math-heavy).
         2. Example: Maybe our **g** works REALLY well on the 10 tweets that we got as examples, but it does HORRIBLY on the other 10,000. This is called **overfitting.** This is VERY bad for learning. It’s up to you to pick the right Hypothesis set (and algorithm) to prevent this.
   3. What does this all mean?
      1. This concept what’s called the **Theory of Generalization**: We want to minimize the error we get, but more importantly we want to minimize the difference in error between our small set and all other examples, so we can learn!
      2. To do this, we can do one of the following:
         1. Increase the number of examples we give the computer
         2. Increase the number of dimensions (but ALSO increase the number of training examples)
         3. Decrease the number of dimensions for a simpler problem
         4. Decrease the number of ways that we can come up with patterns
4. **Machines aren’t that Scary: What you can learn** *(15 minutes)*
   1. How I’m going to organize this section
      1. Rather than giving you specific types of algorithms, I’m going to classify machine learning tasks into different types, and give examples of each
      2. Biggest researcher in machine learning for business right now: **Google!** (by far and away). Link: <http://research.google.com/pubs/ArtificialIntelligenceandMachineLearning.html>
      3. The catch: All my practical examples will be based on Google services
   2. **Decision Trees** (making choices)
      1. Definition: you already KNOW that there is a finite set of options of your data, AND you know that they’re iteratively structured. This is true for things like species trees, genetics, things that happen in a time series (aka ways a user can interact on a website…) etc.
      2. Terminology
         1. The training set: is a series of objects that have already been classified by the decisions they made and are in a hierarchical order.
         2. The hypothesis set: a set of classifiers
         3. The output: the final “tree” and the classifiers that go with it.
      3. Examples
         1. YouTube: the single best example of decision trees. The algorithm goes uses the intersection of ideas to come to conclusions. Such as: “does the word ‘soar’ appear AND we’re not talking about airplanes/flights? Okay, put it in the “finance” category.”
   3. **Clustering** (grouping things together)
      1. Definition: an unsupervised learning technique that attempts to group a myriad of different components into a single group, category, or concept
      2. Terminology
         1. The training set: objects that all have the same property (or just objects! Can be unsupervised)
         2. The hypothesis set: a function that separates objects by category
         3. Output: the similarity of clusters to the “ideal” cluster
      3. Examples:
         1. Google News: does this all the time. Groups news stories that have similar words in them together. If the probability of a word occurring in one news story is high, it is grouped as a new class
         2. Spam filters: find emails that are closest to each other on a list of features. This can be supervised by introducing values that are predicted.
         3. GREAT for unsupervised learning as well as supervised, and can be iterated over and over again. Clusters on clusters on clusters. Examples: finding the genre of a film/piece of music, grouping movies together that have the same properties. Great for research.
   4. **Similarity Learning** (“One of these things is not like the other…”)
      1. Definition:
         1. Not so much “clustering” or “classifying” different categories, but finding the “closeness” of one object to another. Baysian regression possible as well: finding the probability of one thing being “close” to another thing.
      2. Terminology
         1. The training set: Pairs of similar objects and not-so-similar objects
         2. The hypothesis set: a section of functions that determines whether two objects are similar or not
         3. Output: how “close” (metrically) an object is in attributes to another object
      3. Applications
         1. Facial Recognition: Google’s Picasa (and Google Plus) are a subset of similarity learning, where the result set is constantly being reused as the training set. (This is a form of **bootstrapping**).
         2. More recently, G+ compares the similarity of the composition of photo albums and makes the content more evenly distributed (more pictures of landmarks, less pictures of your husband/wife)
         3. Recommendation Systems: finding movies/apps/music in Google Play based on other films that you like. For each set, it’s comparing the features of a movie/game/book to features of nearby neighbors and making a guess. (a variant of K-nearest neighbors)
         4. Crimson Hexagon uses a variant of similarity learning to calculate whether tweets are grouped into different categories.
   5. **Representation Learning**
      1. Definition: transforming or representing existing data as another type of data.
         1. A way of pre-processing data so it can be used by clustering algorithms or other things. Make it more useful
         2. Creating multiple dimensions of data from single-dimension data. Finding layers in your data that you didn’t know existed.
         3. Linear regression is probably the simplest of this type.
      2. Terminology
         1. The training set: Objects that have been transformed to one thing or another
         2. The hypothesis set: Possible transformations
         3. Output: ????????????????????
      3. Examples
         1. Audio/Video processing, semantic analysis
         2. Google photos can now assign a hashtag to a photo before you even upload it: representing it so it’s easier to read/analyse by clustering algorithms
5. **Conclusion** *(2 minutes)*
   1. Thanks!