A Gentle Guide to Machine Learning

Machine Learning is a subfield within Artificial Intelligence that builds algorithms that allow computers to learn to perform tasks from data instead of being explicitly programmed.

Got it? We can make machines learn to do things! The first time I heard that, it blew my mind. That means that we can program computers to learn things by themselves!

Recent Posts

- A practical explanation of a Naive Bayes classifier
- Analyzing 10 years of startup news with Machine Learning
- Creating machine learning models to analyze startup news
- Filtering startup news with Machine Learning
- Introducing Google Sheets add-on for MonkeyLearn

Categories

- Applications
- Guides
- How To
- News
- Text Classification

The ability of learning is one of the most important aspects of intelligence. Translating that power to machines, sounds like a huge step towards making them more intelligent. And in fact, Machine Learning is the area that is making most of the progress in Artificial Intelligence today; being a trendy topic right now and pushing the possibility to have more intelligent machines.

This post will try to give the novice reader a brief introduction to Machine Learning. I'll give a general overview of important concepts, applications and challenges when working with Machine Learning. It's not the goal of this post to give a formal or exhaustive description about the topic, but to give some initial concepts to invite the reader to continue investigating.

The real thing about Machine Learning

Alright, not all is as beautiful as it sounds, Machine Learning has its limits. Still, we can't build intelligent machines like *Data* from *Star Trek* or *Hal 9000* from *2001 a Space Odyssey*. However, we have plenty of examples of **real world applications** where Machine Learning works like a charm. The following are some of the most common categories of practical Machine Learning applications:

Image Processing

Image processing problems basically have to analyze images to get data or do some transformations. Some examples are:

- **Image tagging**, like in Facebook, when the algorithm automatically detects that your face or the face of your friends appear in a photo. Basically a Machine Learning algorithm learns from the photos you manually tag.
- **Optical Character Recognition** (OCR), when algorithms learn to transform a manuscript or scanned text document

into a digital version. The algorithm has to learn to transform an image of a written character into the corresponding digital letter.
- **Self-driving cars**: part of the mechanisms that allow cars to drive by themselves use image processing. A Machine Learning algorithm learns where's the edge of the road, if there's a stop sign or a car is approaching by looking at each frame taken by a video camera.

Text Analysis

Text analysis are processes where we extract or classify information from text, like tweets, emails, chats, documents, etc. Some popular examples are:

- **Spam filtering**, one of the most known and used text classification applications (assign a category to a text). Spam filters learn to classify an email as spam or ham depending on the content and the subject.
- **Sentiment Analysis**, another application of text classification where an algorithm must learn to classify an opinion as positive, neutral or negative depending on the mood expressed by the writer.
- **Information Extraction**, from a text, learn to extract a particular piece of information or data, for example, extracting addresses, entities, keywords, etc.

Data Mining

Data mining is the process of discovering patterns or making predictions from data. The definition is a bit generic, but think of it as mining useful information from a huge table in a database. Each row would be our training instances, and each column a feature. We may be interested in predicting a new column in that table based on the rest of the columns, or discover patterns to group the rows. For example:

- **Anomaly detection**: detect outliers, for example for credit card fraud detection, you could detect which transactions are outliers from the usual purchasing pattern of a user.
- **Association rules**: for example, in a supermarket or e-commerce site, you can discover customer purchasing habits by looking at which products are bought together. This information can be used for marketing purposes.
- **Grouping**: for example, in a SaaS platform, group users by their behaviour or their profile.
- **Predictions**: predict a variable (column in a database) from the rest of the variables. For example, you could predict the credit score of new customers in a bank based by learning from the profiles and credit score of current customers.

Video Games & Robotics

Video games and robotics have been a huge field where Machine Learning has been applied. In general we have an agent (game character or robot) that has to move within an environment (a virtual environment in a video game or physical environment in the case of robots). Machine Learning then can be used to allow the agent to perform tasks, like moving in an environment while avoiding obstacles or enemies. A very popular Machine Learning technique used in these cases is Reinforcement Learning, where the agent learns to perform a task by learning from the reinforcement of the environment (the reinforcement is negative if it hits an obstacle or positive if it gets to the goal).

Alright, I got the value of Machine Learning, but how does it work?

One of the first books I read about Machine Learning about ten years ago was *Machine Learning* by Tom Mitchell. This book was written in 1997, but the general concepts remain valid today.

From that book, I like the following formal definition about Machine Learning:

*A computer program is said to learn to perform a **task T** from **experience E**, if its performance at task T, as measured by a **performance metric P**, improves with experience E over time.*

For example, an artificial game player that has to play chess (task T), could learn by looking at previous chess matches or playing against a tutor (experience E). It's performance P can be measured by counting the percentage of games won against a human.

Let's picture that in a couple of more examples:

Barack Obama's face appears in that image (the generalization would be like Facebook's auto tagging).

- **Example 2**: A system that given a tweet, tells if it talks with a positive or negative mood.
- **Example 3**: A system that given some person's profile, assigns a score representing the probability of that person paying a credit loan.

In example 1, the task is to detect when Barack Obama's face appears in an image. The experience could be a set of images where Barack Obama's face appears and others where it doesn't appear. The performance could be measured as the percentage of new images that our system correctly tags.

In example 2, the task is to assign the sentiment of a tweet. The experience could be a set of tweets and their corresponding sentiment. The performance could be measured as the percentage of new tweets that our system correctly classifies.

In example 3, the task is to assign a credit score. The experience could be a set of user profiles with their

corresponding credit scores. The performance could be measured as the squared error (the difference between the predicted and expected score).

In order to allow the algorithm to learn to transform the input to the desired output, you have to provide what is called training instances or **training examples**, which in Mitchell's definition are defined as experience E. A training set is a set of instances that will work as examples from which the Machine Learning algorithm will learn to perform the desired task. Pretty intuitive, isn't it? It's like when you show a little baby how to throw a ball, you throw the ball a couple of times to show him how to do it, and by looking at those examples, he starts to learn to do it

attributes or **features**. The features are the way to characterize each instance. For instance, in example 1, an image could be characterized by the gray level of each of its pixels. In example 2, a tweet could be characterized by the words that appear in the tweet. In example 3, a credit record could be represented by the age of the person, the salary, the profession, etc.

Calculating and selecting the proper features to represent an instance is one of the most important tasks when working with Machine Learning, we'll take a look at this point later in this post.

Categories of Machine Learning algorithms

At this point we have to talk about two general categories of Machine Learning algorithms: **Supervised Learning** or **Unsupervised Learning** algorithms. The main difference between both approaches resides in the way we feed training examples to our algorithm, how the algorithm uses them and the type of problems they solve.

Supervised Learning

In the case of supervised learning, the Machine Learning algorithm can be seen as a process that has to transform a particular input to a desired output.

The Machine Learning process then has to learn how to transform every possible input to the correct/desired output, so each training example has the particular input and the desired output.

In the example about the artificial chess player, our input would be a particular chess board state, and the output would be the best possible movement in that situation.

supervised learning:

Classification

When the output value belongs to a discrete and finite set, we're talking about a classification. Example 2 can be solved as a classification problem, the output is a finite set of options: positive, negative or neutral. In this case, our training examples will look like:

Text	Sentiment
"Don't stay here if you can avoid it. Everything smells like old cigarettes."	Negative
"Friendly service. Superior room! Loved the high ceiling."	Positive
...	...

Regression

When the output value is a continuous number, for example, a probability, then we're talking about a regression problem.

Example 3 is a regression as the result is a number between 0 and 1 that represents the probability that a person will pay his debts. In this case, our training examples will look like:

Occupation	Income	Age	Score
Engineer	> $60K	30	0.85
Unemployed	< $60K	20	0.3
...	...	...	...

Supervised learning is the most popular category of Machine Learning algorithms. The disadvantage of using this approach is that for every training example, we have to provide the correct output, and in many cases, this is quite expensive. For example,

correct sentiment (positive, negative or neutral). That would require a group of humans to read and tag each tweet (quite a time consuming and boring task). This is usually a very **common bottleneck for Machine Learning** algorithms: gather quality tagged training data.

Unsupervised Learning

There's a second category of Machine Learning algorithms called unsupervised learning. In this case, the training examples only need to be the input to the algorithm, but not the desired output. The typical use case is to discover the hidden structure and relations between the training examples. A typical example are clustering algorithms, where we learn to find similar instances or groups of instances (clusters). E.g.: we have a news article and we want to get similar ones to recommend. Some clustering algorithms like K-means "learn" to do that by only looking at the input.

Machine Learning algorithms

Ok, here's when the math and logic comes to action. In order to transform an input to a desired output we can use different models. Machine Learning is not a unique type of algorithm, perhaps you heard about Support Vector Machines, Naive Bayes, Decision Trees or Deep Learning. Those are different Machine Learning algorithms that try to solve the same problem: learn to transform every input to the correct output.

Those different Machine Learning algorithms use different paradigms or techniques to do the learning process and represent the knowledge of what they have learned.

But before we go ahead and talk about each algorithm, a common principle is that Machine Learning algorithms try to

razor. Every Machine Learning algorithm, regardless of the paradigm it uses, will try to create the simplest hypothesis (the one that makes fewest assumptions) that explains most of the training examples.

There are lot of Machine Learning algorithms, but let's briefly mention three of the most popular ones:

- **Support Vector Machines**: The model tries to build a set of hyperplanes in a high dimensional space that tries to separate instances of different classes by getting the largest separation between the nearest instances from different classes. The concept intuitively is simple, but the model can be very complex and powerful. In fact, for some domains it is one of the best Machine Learning algorithms you can use nowadays.
- **Probabilistic Models**: these models usually try to predict the correct response by modeling the problem with a probability distribution. Perhaps the most popular algorithms in this category are Naive Bayes classifiers, that

use the Bayes theorem alongside with strong independence assumptions between the features. One of their advantages besides being a simple yet powerful model, is that they return not only the prediction but also the degree of certainty, which can be very useful.

- **Deep Learning**: is a new trend in Machine Learning based on the very known Artificial Neural Network models. Neural networks have a connectionist approach, they try to emulate (in a very simplified way) the way the brain works. Basically they consist of a huge set of interconnected neurons (the basic unit of processing), organized in various layers. Deep learning has, in a few words, developed new structures with deeper layers and improved the learning automatically with higher levels of abstraction.

Important aspects when dealing with Machine Learning

Again, Machine Learning sounds like a beautiful concept, and it is, but there are some processes involved that are not so automatic. In fact, many times manual steps are required when designing the solution. Nevertheless, they are of **huge importance** in order to obtain decent results. Some of those aspects are:

Which type of Machine Learning algorithm should I use?

Supervised or unsupervised?

Do you have tagged data? That is, the input with its corresponding output? In that case you can apply supervised learning algorithms. If not, perhaps unsupervised algorithms can solve the problem.

Classification, regression or clustering?

That mainly depends on what you're trying to solve. If you want to tag data (assign a tag within a discrete set of options), classification may be the right option. If on the other hand, you need to assign a number, for example a score, regression may be the best option for you. Perhaps what you need is to recommend similar products to users depending on what they are currently looking at in an e-commerce site, in that case clustering could be the best option for you.

Deep Learning, SVM, Naive Bayes, Decision Trees… which one is the best?

most powerful and versatile in different applications. But take into account that depending on the particular application, some Machine Learning algorithms may be a better choice than others. Just see which are their individual strengths and try them!

Feature engineering

Feature engineering is the process by which we extract and select the most important features to be used to represent our training examples and instances to be processed by our Machine Learning algorithm. This process is one of the most important aspects of Machine Learning (and sometimes not given enough credit and importance).

Pay attention to that: if you don't provide quality features to your algorithm, the results will be bad, no matter if you use the best Machine Learning algorithm out there. It's like trying to learn to read with your eyes in complete darkness, you won't be able to do that, no matter how intelligent you are.

Feature extraction

In order to feed the Machine Learning algorithm with data, you usually have to transform the raw data into something that the algorithm can 'understand'. That process is known as **feature extraction**. Usually we're talking about transforming the raw data into a vector of features.

In example 1, how do we feed the Machine Learning algorithm with an image?

Well, a straightforward approach would be to transform the image into a vector where each component is the gray value of each pixel in the image. So, each component or feature, would

This approach may work, but perhaps it would work better if we provide higher level features:

- Does the image contain the face of a person?
- What's the skin color?
- What's the eye color?
- Does the face have hair?
- …

Those are higher level features, that feed the algorithm with more knowledge than just the gray level of each pixel (and their calculation may be done by other Machine Learning algorithms!). By providing higher level features we're 'helping' the Machine Learning algorithm with better information to learn to decide if my or someone else's face appears in an image.

If we provide better feature extraction:

- We'll have more chances that our algorithm will learn to perform the desired task.

- We may need less training examples to learn.
- As a result, we may reduce dramatically the time needed to train our model.

Feature selection

Sometimes (not to say mostly), the features that we selected to feed our algorithm may be useless. For example, when trying to tag the sentiment of a tweet, we could add the length of the tweet as a feature, the time of the day the tweet was written, etc. Those features may be useful or not, and there are automatic methods to figure out which of them are the most useful. Intuitively, **feature selection algorithms** use techniques to

Another important thing to keep in mind is: avoid using huge feature sets. One may be tempted to add all the possible features to the model and let the algorithm just learn. But that's not a good idea, as we add more features to represent our instances, the dimension of the space increases, making it more sparse. Intuitively, as we get more features, we have to get much many instances to represent a decent amount of the combinations. This is a very common problem known as the curse of dimensionality, as the complexity of the model grows, the number of training examples needed grows exponentially, and believe me, that's a problem.

Training examples

So, you have to feed the Machine Learning algorithm with training examples. Depending on the problem you want to solve, we may be talking in the order of hundreds, thousands, millions or billions of training examples. Besides, it's very important to keep the quality of the examples, if you feed the algorithm with wrong examples, the chances to obtain good results are low.

Collecting quality data in huge volumes to train Machine Learning algorithms is usually an expensive thing to do. Unless you already have tagged data, you may have to manually tag the data yourself or pay others to do that. Some tools to try to solve that are **crowdsourcing platforms** where you can find labor to do the job. Also **bootstrapping** is a way to try to make tagging more efficient by using your own Machine Learning model to help you.

The general rule regarding training examples is: the more quality training data you can gather, the better results you may get.

Testing examples and performance metrics

won't know if your model actually learned anything!

The concept is simple, we use a **testing set**, a set of instances not contained in the training set. Basically we'll input every testing example to our model and see if it performs as expected. In the case of supervised learning classification, we just input each testing instance and check if the outputs are what we expected. If our model returns the correct output on 95% of the testing examples, we say that our model has an accuracy of 95%.

It's important to remember that the training and testing sets of instances must be disjoint, this is the only way to test the generalization and prediction power of your model. You may have very good results by measuring the accuracy in your training data, but get poor results when measuring in a separate testing set. That problem is known as overfitting, that is, the algorithm overfits the training examples and has a poor predictive power. Usually the way to avoid that is to try to use a

simpler model with less features, simplify the model and use a bigger and more representative training set.

Accuracy is the most basic metric, you should also look at other metrics like Precision and Recall which will tell you how well the algorithm performs on each class (when working with supervised learning classification). Confusion matrices are a great tool to see where our classification algorithm is 'confusing' predictions.

For regression and clustering problems you have other sets of metrics that will allow to see if your algorithm is performing well.

Performance

solution. In the case of Machine Learning applications, this can be a complex task. First, you have to select the Machine Learning framework, which is not easy as not all programming languages have strong tools on that. Python and Scikit-learn are good examples of a programming language where a strong Machine Learning framework was built.

Having chosen your framework, performance issues may arise. Depending on the amount of data, complexity and algorithm designed, it may take large amounts of computing power and memory to run the training process. You'll probably have to run multiple trainings until you get decent results. Besides, usually you may be retraining your model to cover new instances and keep improving its accuracy.

In order to train huge models and get fast results when using it, we're usually talking of various GBs of RAM and multi-core machines to parallelize the processing.

These are mostly practical issues, but definitively important ones if you want to deploy a real Machine Learning solution in

production.

Final words

That's it, a brief overview of what Machine Learning is about. There are plenty of other real world applications not covered here, plenty of other Machine Learning algorithms and concepts to talk about, but we leave the reader to do his/her own research on that.

Machine Learning is powerful but hard, the difficulties and aspects to take into account described in this post are just the tip of the iceberg.

One can end very disappointed by the many difficulties to solve before getting on track.

This is why we created MonkeyLearn, to democratize the access to Machine Learning technologies applied to text analysis. To avoid reinventing the wheel and allow every software developer or entrepreneur to quickly get practical results. These are our main challenges, to abstract the end user of all of these problems, ranging from Machine Learning complexities to practical scalability issues to make just plug and play Machine Learning.

Hope you enjoyed this post! Just drop me a line if you have any questions or comments!

By Raúl Garreta | August 27th, 2015 | Guides | 23 Comments

About the Author: Raúl Garreta