the morning paper

an interesting/influential/important paper from the world of CS every weekday morning, as selected by Adrian Colyer

A few useful things to know about machine learning

OCTOBER 29, 2014

A few useful things to know about machine learning (http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf) – Domingos 2012

> *Developing successful machine learning applications requires a substantial amount of 'black art' that is hard to find in textbooks*

This paper looks at twelve key lessons including pitfalls to avoid, important issues to focus on, and answers to common questions. The paper was published in 2012, and since then the excellent '**Data Science for Business (http://shop.oreilly.com/product/0636920028918.do)**' book by Provost and Fawcett has been released by O'Reilly. Now much of the wisdom from this paper can indeed be found in a textbook! If you enjoy this paper, I highly recommend the book as well.

According to the paper, the key to not getting lost in the huge space of learning algorithms is to understand that they are composed of three elements: a representation model (e.g. k-nearest neighbour, naive bayes, decision trees); an evaluation function (scoring function) to tell good from bad; and an optimization technique to search for the highest scoring classifier.

> *Most textbooks are organized by representation, and it's easy to overlook the fact that the other components are equally important.*

You want your machine learning to work well (generalize) outside of the examples in the training set you gave it.

> *The most common mistake among machine learning beginners is to test on the training data and have the illusion of success… in the early days of machine learning, the need to keep training and test data separate was not widely appreciated.*

(Of course, if you started your journey as a 'machine learning beginner' by taking **Andrew Ng's online course (https://www.coursera.org/course/ml)** you won't be falling into this trap!).

Data alone is not enough, you also need to supply some domain knowledge to help guide the process.

> *Machine learning is not magic; it can't get something from nothing. What it does is get more from less. Programming, like all engineering, is a lot of work: we have to build everything from scratch. Learning is more like farming, which lets nature do most of the work. Farmers combine seeds with nutrients to grow crops. Learners combine knowledge with data to grow programs.*

Overfitting is a well-known problem in machine learning, and cross-validation can help to combat this.

> *Nevertheless, you should be skeptical of claims that a particular technique 'solves' the overfitting problem. It's easy to avoid overfitting (variance), by falling into the opposite problem of underfitting (bias). Simultaneously avoiding both requires learning a perfect classifier, and short of knowing it in advance there is no single technique that will always do best.*

The 'curse of dimensionality' refers to the fact that many algorithms that work in low dimensions become intractable when the input is high-dimensional.

> *…the similarity-based reasoning that machine learning algorithms depend on (explicitly or implicitly) breaks down in high dimensions*

Fortunately many problem domains are non-uniform giving an effectively lower dimension, or algorithms for explicitly reducing the dimensionality can be used.

The *most important factor in the success of a machine learning project is the features used*. If the raw data is not in a form that is amenable to learning, you may be able to construct features from it that are:

> *First-timers are often surprised by how little time in a machine learning project is spent actually doing machine learning. But it makes sense if you consider how time consuming it is to gather data, integrate it, clean it and pre-process it, and how much trial and error can go into feature design.*

What's better? Smart algorithms or lots of data:

> *As a rule of thumb, a dumb algorithm with lots and lots of data beats a clever one with modest amounts of it.*

This brings up the problem of scalability:

> *In most of computer science, the two main limited resources are time and memory. In machine learning there is a third one: training data. Which one is the bottleneck has changed from decade to decade.*

Why have just one learner though, when you can have many, *ensembles* of learners do best.

> *…researchers noticed that if instead of selecting the best variation found, we combine many variations, the results are better – often much better – and at little extra effort for the user*

The winner of the '**Netflix prize (http://netflixprize.com/)**' was a stacked ensemble of over 100 learners.

There's plenty more folk wisdom in the paper, so do check it out if this has piqued your interest.

from → Uncategorized

No comments yet