Very Brief Introduction to Machine Learning for AI

The topics summarized here are covered in these slides.

Intelligence

The notion of *intelligence* can be defined in many ways. Here we define it as the ability to take the *right decisions*, according to some criterion (e.g. survival and reproduction, for most animals). To take better decisions requires *knowledge*, in a form that is *operational*, i.e., can be used to interpret sensory data and use that information to take decisions.

Artificial Intelligence

Computers already possess some intelligence thanks to all the programs that humans have crafted and which allow them to "do things" that we consider useful (and that is basically what we mean for a computer to take the right decisions). But there are many tasks which animals and humans are able to do rather easily but remain out of reach of computers, at the beginning of the 21st century. Many of these tasks fall under the label of *Artificial Intelligence*, and include many perception and control tasks. Why is it that we have failed to write programs for these tasks? I believe that it is mostly because we do not know explicitly (formally) how to do these tasks, even though our brain (coupled with a body) can do them. Doing those tasks involve knowledge that is currently implicit, but we have information about those tasks through data and examples (e.g. observations of what a human would do given a particular request or input). How do we get machines to acquire that kind of intelligence? Using data and examples to build operational knowledge is what learning is about.

Machine Learning

Machine learning has a long history and numerous textbooks have been written that do a good job of covering its main principles. Among the recent ones I suggest:

- Chris Bishop, "Pattern Recognition and Machine Learning", 2007
- Simon Haykin, "Neural Networks: a Comprehensive Foundation", 2009 (3rd edition)
- Richard O. Duda, Peter E. Hart and David G. Stork, "Pattern Classification", 2001 (2nd edition)

Here we focus on a few concepts that are most relevant to this course.

Formalization of Learning

First, let us formalize the most common mathematical framework for learning. We are given training examples

$$\mathcal{D} = \{z_1, z_2, \ldots, z_n\}$$

with the z_i being examples sampled from an **unknown** process $P(Z)$. We are also given a loss functional L which takes as argument a decision function f and an example z, and returns a real-valued scalar. We want to minimize the expected value of $L(f, Z)$ under the unknown generating process $P(Z)$.

Supervised Learning

In supervised learning, each examples is an (input,target) pair: $Z = (X, Y)$ and f takes an X as argument. The most common examples are

- regression: Y is a real-valued scalar or vector, the output of f is in the same set of values as Y, and we often take as loss functional the squared error

$$L(f, (X, Y)) = ||f(X) - Y||^2$$

- classification: Y is a finite integer (e.g. a symbol) corresponding to a class index, and we often take as loss function the negative conditional log-likelihood, with the interpretation that $f_i(X)$ estimates $P(Y = i|X)$:

$$L(f, (X, Y)) = -\log f_Y(X)$$

  where we have the constraints

$$f_Y(X) \geq 0 \ , \ \sum_i f_i(X) = 1$$

Unsupervised Learning

In unsupervised learning we are learning a function f which helps to characterize the unknown distribution $P(Z)$. Sometimes f is directly an estimator of $P(Z)$ itself (this is called density estimation). In many other cases f is an attempt to characterize where the density concentrates. Clustering algorithms divide up the input space in regions (often centered around a prototype example or centroid). Some clustering algorithms create a hard partition (e.g. the k-means algorithm) while others construct a soft partition (e.g. a Gaussian mixture model) which assign to each Z a probability of belonging to each cluster. Another kind of unsupervised learning algorithms are those that construct a new representation for Z. Many deep learning algorithms fall in this category, and so does Principal Components Analysis.

Local Generalization

The vast majority of learning algorithms exploit a single principle for achieving generalization: local generalization. It assumes that if input example x_i is close to input example x_j, then the corresponding outputs $f(x_i)$ and $f(x_j)$ should also be close. This is basically the principle used to perform local interpolation. This principle is very powerful, but it has limitations: what if we have to extrapolate? or equivalently, what if the target unknown function has many more variations than the number of training examples? in that case there is no way that local generalization will work, because we need at least as many examples as there are ups and downs of the target function, in order to cover those variations and be able to generalize by this principle. This issue is deeply connected to the so-called **curse of dimensionality** for the following reason. When the input space is high-dimensional, it is easy for it to have a number of variations of interest that is exponential in the number of input dimensions. For example, imagine that we want to distinguish between 10 different values of each input variable (each element of the input vector), and that we care about about all the 10^n configurations of these n variables. Using only local generalization,

we need to see at least one example of each of these 10^n configurations in order to be able to generalize to all of them.

Distributed versus Local Representation and Non-Local Generalization

A simple-minded binary local representation of integer N is a sequence of B bits such that $N < B$, and all bits are 0 except the N-th one. A simple-minded binary distributed representation of integer N is a sequence of $log_2 B$ bits with the usual binary encoding for N. In this example we see that distributed representations can be exponentially more efficient than local ones. In general, for learning algorithms, distributed representations have the potential to capture exponentially more variations than local ones for the same number of free parameters. They hence offer the potential for better generalization because learning theory shows that the number of examples needed (to achieve a desired degree of generalization performance) to tune $O(B)$ effective degrees of freedom is $O(B)$.

Another illustration of the difference between distributed and local representation (and corresponding local and non-local generalization) is with (traditional) clustering versus Principal Component Analysis (PCA) or Restricted Boltzmann Machines (RBMs). The former is local while the latter is distributed. With k-means clustering we maintain a vector of parameters for each prototype, i.e., one for each of the regions distinguishable by the learner. With PCA we represent the distribution by keeping track of its major directions of variations. Now imagine a simplified interpretation of PCA in which we care mostly, for each direction of variation, whether the projection of the data in that direction is above or below a threshold. With d directions, we can thus distinguish between 2^d regions. RBMs are similar in that they define d hyper-planes and associate a bit to an indicator of being on one side or the other of each hyper-plane. An RBM therefore associates one input region to each configuration of the representation bits (these bits are called the hidden units, in neural network parlance). The number of parameters of the RBM is roughly equal to the number these bits times the input dimension. Again, we see that the number of regions representable by an RBM or a PCA (distributed representation) can grow exponentially in the number of parameters, whereas the number of regions representable by traditional clustering (e.g. k-means or Gaussian mixture, local representation) grows only linearly with the

number of parameters. Another way to look at this is to realize that an RBM can generalize to a new region corresponding to a configuration of its hidden unit bits for which no example was seen, something not possible for clustering algorithms (except in the trivial sense of locally generalizing to that new regions what has been learned for the nearby regions for which examples have been seen).