

Daniel Grollman, Aude Billard  
EPFL  
Lausanne  
Switzerland  
{daniel.grollman,aude.billard}@epfl.ch

## Synonyms

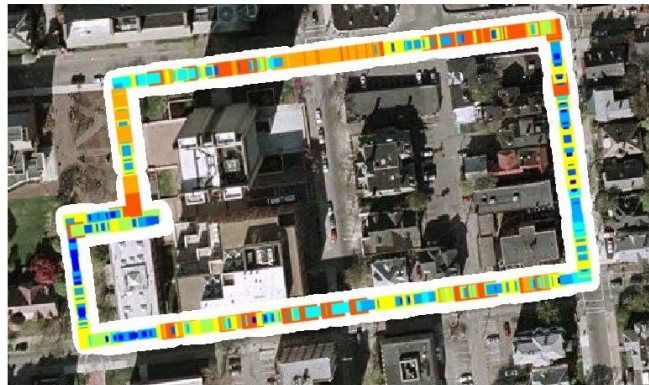
Machine Learning, Data Mining, Artificial Intelligence

## Definition

Learning algorithms are methods by which data is processed to extract patterns that can later be applied to novel situations. Generally, a system is said to learn if the performance of some task improves (with respect to a particular metric) after the analysis of data (experience). Familiar examples of learning systems are speech recognizers that adapt to individual users, automatic text translation services, and product recommenders. Underlying all of these techniques is a *model*, which defines what assumptions are made about the data and patterns that can be discovered in it. A learning algorithm is what is responsible for generating parameters for the model by processing collected data.



(a)



(b)

Figure 1: Example applications of learning algorithms. In (a), correspondences between languages have been learned from known translations (supervised learning). In (b), a robot has automatically grouped different locations on its path by similarity based only on sensor readings at those locations (unsupervised learning).

Images copyright MIT, Brown

## Theoretical Background

Learning algorithms are developed with the aim of enabling systems to adapt to new situations. To do so, they must process data, of which there are generally three types: Training data, Validation data, and Test data. Training data is assumed to contain all the information necessary with which to build the model, and validation data is a separate set that is used to confirm that the model works as expected. Test data is that which the system encounters after training is complete, and provides the actual measure of how successful learning was. To reduce the sensitivity of an algorithm to the particular data used in training, *cross validation* is a technique where all the available data is repeatedly randomly split into training and validation sets, and performance is measured as the average over the multiple trials.

There are many different types of learning algorithms, corresponding to the different types of patterns that can be discovered, the manner in which data is collected and analyzed, and how much information is put in by the user. Here we present some of the broad dichotomies in learning algorithms:

- Regression vs. Classification: In regression one is concerned with a particular value associated with a query (i.e: How much rain will fall tomorrow?), while in classification one only cares what group the query belongs to (i.e: Is this picture of a person or a chair?).
- Supervised vs. Unsupervised: In supervised learning, training data consists of queries and their correct answers, and the algorithm learns to generalize to new queries (Figure 1a). In unsupervised learning, the algorithm must itself discover how the data should be organized (Figure 1b). Regression is typically approached in a supervised manner, while classification can be either supervised or unsupervised (if the class labels are known). There also exist semi-supervised techniques, where only some of the data has associated answers.
- Batch vs Incremental Learning: In batch learning, all of the training data is presented at once, and processed as a whole. Incremental learning instead starts with some model and adjusts this model as more data is gathered.
- Prior Knowledge vs Tabula Rasa: Often, practitioners can steer an algorithm towards discovering a particular relationship by adding in additional information as to how the world works. Tabula Rasa (literally “blank slate”) indicates that no prior knowledge is assumed.
- Exploration vs Exploitation: There is a tension between exploiting what is already known and exploring unknown territory. Particularly, the potential gains (e.g prediction accuracy) of collecting more data must be weighed against any associated cost (e.g. computation time). Often the human users of an algorithm perform this tradeoff implicitly during training, but some algorithms that actively collect their own data do it explicitly.

## Important Scientific Research and Open Questions

Learning algorithm (and model) development follows multiple paths, and are often inspired by learning as observed in nature. Some are psychologically based, drawing on studies of learning and adaptation in humans and other animals, such as *reinforcement learning* (Sutton and Barto, 1998), which trains agents by rewarding good results and punishing bad ones. Other techniques go a level lower, and attempt to mimic the behavior of the brain or its constituent neurons directly, such as work in *neural networks* (Bishop, 2000). At a larger biological scale are *evolutionary algorithms*, which draw inspiration from species dynamics and ‘breed’ different possible solutions to find the best (Ashlock, 2006). Alternatively, mathematically based approaches abstract away the substrate of learning and seek to describe the learning process statistically (MacKay, 2003).

No matter what techniques is used, one common issue that must be dealt with is noise, which can arise from different sources such as the measurement process itself or sample bias. Removing noise, or *denoising*, is important as the goal of learning is to model only the signal, which is obscured by the noise. Some algorithms model the noise directly and consider it during learning, while others depend on a separate denoising preprocessor.

Improper treatment of noise can lead to over- or under-fitting. In overfitting, the learned model conforms to the training data extremely well, but fares poorly on validation or test data; one can say that it has learned the training data too well. These errors can be a result of the learning algorithm considering the noise to be important and modeling it. Alternatively, under-fitting is akin to over-generalization, where the learned model is not specific enough, and has not extracted all that there is to be learned from the training data. This situation can arise when some important parts of the training data are considered to be noise and discarded.

As learning algorithms can only discover patterns that exist in the data they are trained on, proper data collection is key. Often, data from many different regions of the dataspace are needed, so that the learnt model can be applicable all over. However, in many real world examples, the data space is extremely high dimensional. For example, a face recognition system may have a dimensionality equal to the number of pixels in an image. This has led to the so called *curse of dimensionality*, which states that the number of samples necessary for successful training grows exponentially with the dimensionality of the dataspace. For large dataspaces, this number quickly outstrips our current ability to collect, store, and process the necessary data.

*Dimensionality reduction* or *Feature Extraction* techniques seek to alleviate this curse by finding which portions of the data are actually necessary for learning, and removing extraneous ones. For instance, of 100 recorded dimensions, perhaps only 2 (or 2 particular combinations of the 100) are sufficient. Likewise, instead of operating in the raw pixel space of an image, a learning algorithm could instead work with ‘interesting’ points in the image, such as corners.

As having good data is so important, some current work in learning algorithms look to tie the learning system and the data collection process together more tightly. That is, rather than collecting all of the data before learning, systems can use partial learning results to steer the collection of more data. Most often seen in tandem with incremental approaches, *active learning* techniques

select the data to be added based on an estimate of where the current learned model is likely to fail. If the data must be evaluated by a human, we have a *tutelage* framework, where a human and a learning agent work together to improve the capabilities of the learner.

## Cross-References

- Learning in artificial neural networks
- Reinforcement Learning
- Supervised Learning

## References

- D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer, 2006.
- C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 2000.
- D. J. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge, 2003.
- R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

## Glossary