The space of machine learning applications

In real-world computing, we target applications that process complex real-world data to perform semantically significant operations at a level close to human cognition. This processing will be done by future extensions of machine learning algorithms. In this section, we classify both existing and future machine learning applications in a novel way that highlights the potential gains to be had from real-world computing. *Machine learning* (ML) is the discipline that studies how to program computers to evolve behaviors based on example data or past experience [ALP2010]. The results of machine learning are directly useful in many ways: to optimize performance, to improve intelligence, to predict the future, and to aid understanding. ML is the primary technique to solve problems for which we do not have explicit algorithms. Often it is not possible to write down an explicit algorithm because the problem is simply too large or too complex. This is usually the case for processing of complex data to extract useful high-level information, such as audio language translation mentioned before. With machine learning, programmers no longer have to laboriously write code to process data, but the computer itself writes the code according to a learning algorithm. This greatly increases programmer productivity as well as allowing programmers to tackle real-world computing problems.

The general approach is to suppose that the data has regularities that fit a given general model (the *inductive hypothesis*, also called the *inductive bias*) and to determine the best fit of the data to the model. Machine learning comprises both determining the right model and calculating the parameters for the best fit in this model. The model can define a simple classification (buy or sell a stock) or a complex behavior (how to play a game). The power of machine learning, like the power of the scientific method, depends crucially on the choice of the model. If it is properly chosen, the model is not a limitation. On the contrary, a good model can give essential insights on the phenomena.

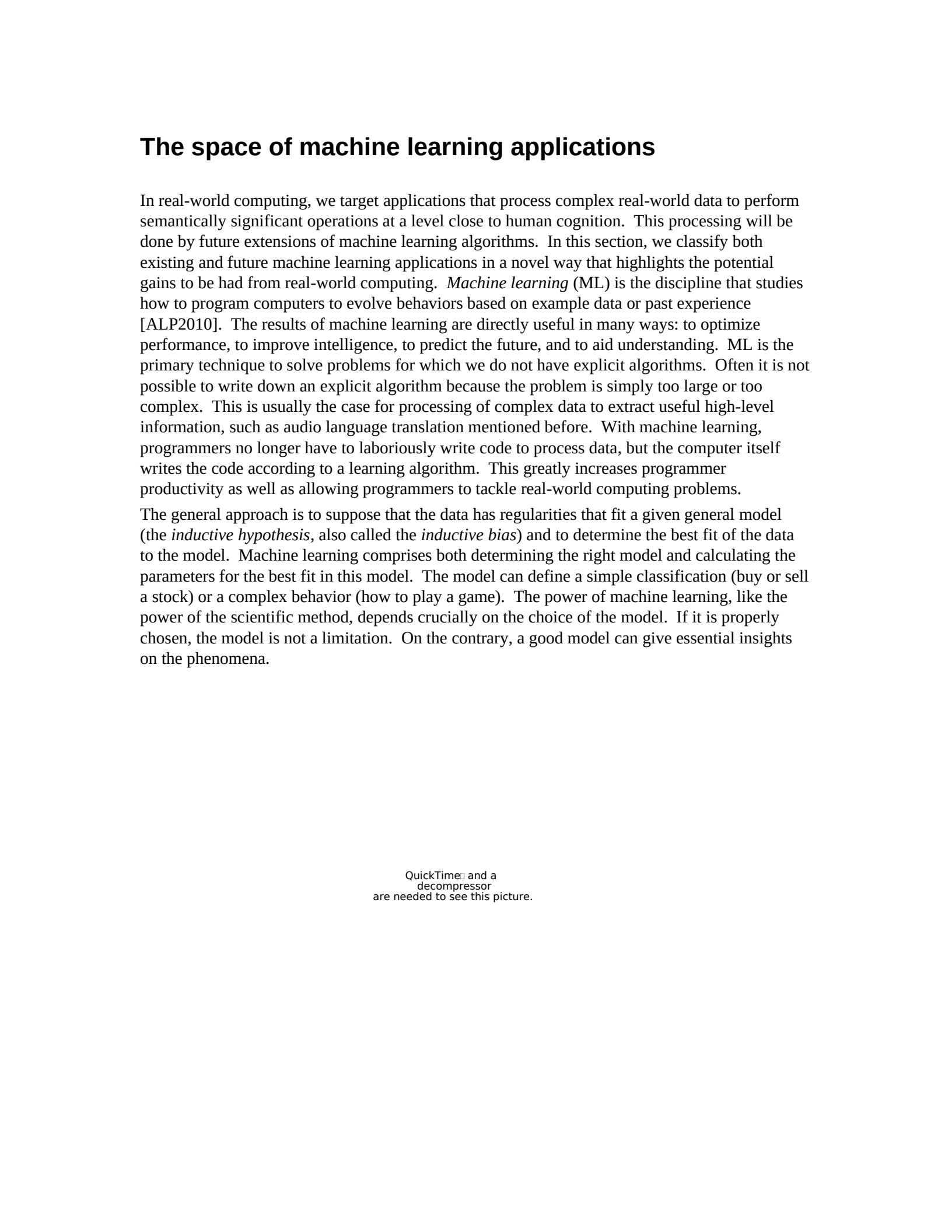

Figure 1: The space of machine learning algorithms

The space of machine learning algorithms is vast. It covers well-known kinds of applications such as Web search, corpus-based translation, recommendation systems, and computer games. But it is by far not limited to this. Most applications that require large computational and/or storage resources can be considered of this form. To achieve good results, these algorithms usually run in two phases: a *learning phase* whenever new information comes in, and a *query phase* when being used. We note that the learning phase is sometimes (partially) done by the manufacturer, such as in speech recognition systems. Figure 1 classifies algorithms in a three-dimensional space, depending on whether the learning and query phases have high elastic

degrees of interactivity: whether the algorithm is used for one-shot queries, with one-way streams, or in conversations (the darker or redder applications are more interactive). The figure shows some existing applications in this space. We are careful to separate the learning phase of an implemented algorithm from improvements in the algorithm itself. The latter can be considered as a form of "second order" learning. But it is much more unpredictable than a programmed learning phase: it is tantamount to predicting the future of the discipline of machine learning. Therefore Figure 1 gives only the learning phase of an existing algorithm without taking algorithmic improvements into account.

The highly elastic resource requirement combines two simpler requirements: it means that the application needs many resources for a short time. It is not enough for the application to need high resources if there is no constraint on time, or for the application to run in a short time if it does not need many resources. Both of these cases are easily handled by systems of modest size. The difficulty is when the two requirements are combined. For example, weather forecasting needs an enormous quantity of computing and storage resources, and it must give a result in time for users to profit from the prediction.

We extend the learn/query space with a third dimension related to time, namely the degree of interactivity. For our purposes, we distinguish three common degrees of interactivity that have a significant effect on the algorithms used for the applications. In a *one-shot query*, there is a

Fi

single learning phase (or several widely separated learning phases) and queries that interrogate the application in between learning phases. A typical example of a one-shot query is a data processing application with a deadline: for example, media data that needs lots of resources for a one-off format transformation. In a *one-way stream*, the learning phase is continuous (information coming in and being incorporated in a stream) and the query phase consists of a set of queries, each of which uses the information that has been assimilated up to that point. A typical example of a one-way stream is Google Search, if we consider that the search database is principally affected by changes in the Web and not by the queries themselves. In a *conversation*, there is a two-way interaction between the application and the user such that the user's queries can strongly affect the future behavior of the application and vice versa. A typical example of a conversation is a *game*. This includes two-player games like chess and also multi-user role-playing games such (where the conversation is now between *n* players and the application).

References

[ALP2010] Ethem Alpaydın. *Introduction to Machine Learning*. Second Edition. MIT Press, 2010.