Anatomy of a Learning Problem

Mark D. Reid, James Montgomery, Mindika Premachandra

Research School of Computer Science
Australian National University
Canberra ACT 0200, Australia

Abstract — In order to relate machine learning problems we argue that we need to be able to articulate what is meant by a single machine learning problem. By attempting to name the various aspects of a learning problem we hope to clarify ways in which learning problems might be related to each other. We tentatively put forward a proposal for an *anatomy* of learning problems that will serve as scaffolding for posing questions about relations. After surveying the way learning problems are discussed in a range of repositories and services. We then argue that the terms used to describe problems to better understand a range of viewpoints within machine learning ranging from the theoretical to the practical.

1 Introduction

As machine learning researchers and practitioners, we define and solve problems every day. However, the sheer diversity of modern machine learning can make it difficult for, say, a learning theorist who focuses on bandit problems to appreciate what kind of problem a deep-belief network researcher is trying to solve and *vice versa*. As the FAQ from the MLComp project[1] puts it:

> Anyone who has worked with machine learning knows that it's a zoo. There are a dazzling array of methods published at conferences each year, leaving the practitioner, who just wants to choose the best method for his/her task, baffled.

We argue that an unavoidable part of the problem is that different research aims necessarily focus on different aspects of a learning problem and therefore use and emphasise different terms. One could also reverse cause and effect by invoking the Sapir-Whorf hypothesis. This linguistic theory states that language is not just a medium for articulating our thoughts but is a medium that affects the way in which we conceptualise the world. In his *Notation as a tool of thought*[2] Ken Iverson makes a similar point concerning programming, rather than human, languages, quoting George Boole: "That language is an instrument of human reason, and not merely a medium for the expression of thought, is a truth generally admitted."

Whatever the direction of causality, there is strong case for the correlation between the diversity of terminology and the machine learning "zoo" we find ourselves in. In an attempt to find some structure or common ground this paper presents some tentative steps towards surveying and discussing terminology used to describe machine learning problems.

1.1 What is a Learning Problem?

For this preliminary work, we will take a fairly narrow view of machine learning and begin by focusing on describing properties of the most common varieties of classification, regression, clustering, and some others. We start by trying to pin down some high-level terminology and, in particular, what we mean by "machine learning problem".

1.1.1 Background Knowledge

There are relatively few books, papers, or projects that attempt to categorise learning problems. Carbonell, Michalski, and Mitchell's textbook[3] from 1983 sketches a taxonomy of machine learning research at the time along three dimensions: *underlying learning strategies*, *representation of knowledge*, and *application domain*. We find some discussion of classification and regression problems under "learning strategies" that learn "from examples". A distinction is made under this heading between "one-trial" and "incremental" learning. Problems such as clustering fall under those strategies that learn "from observation and discovery". Along the "representation of knowledge" dimension is a list of what is required of a learning algorithm to solve a problem. These include "decision trees", "grammars", "parameters", "logical expressions", and more.

In his later book, Mitchell[4] introduces the notion of a *Well-Posed Learning Problem* by defining what it means for a computer to learn:

> A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, measured by P, improves with experience E.

Mitchell uses the term "task" here to denote broad categories such as *reinforcement learning* and *concept learning* ("Inferring a boolean-valued function from training examples of its input and output").

1.1.2 Generalising From Examples

Both those takes on describing machine learning arguably focus on what learning is rather than what learning acts upon. Our view, which we articulate further below, attempts to turn around Mitchell's definition of learning by saying a **learning problem** consists of *data*, *models*, and an *assessment procedure* and defines learning to be any process that returns models assessed to be of higher quality when presented with larger amounts of data.

To expand on this more problem-focused approach, we turn to repositories and services for machine learning on the web for examples of taxonomies "in the wild". These projects necessarily have to grapple with the question of how to classify a variety of data sets, methods, performance measures, etc. in order to present them in a unified manner. We survey some of the terminology used by these projects in §3 below.

1.2 Caveats and Disclaimers

The space of machine learning problems is already rich and rapidly expanding so any attempt for comprehensiveness is doomed. We do not claim we are particularly systematic in our survey of projects either. Projects were chosen by the authors familiarity with the projects or their perceived visibility within the machine learning literature. This means there is very likely a bias in the terminology towards describing *predictive* machine learning problems such as classification and regression (i.e., those solved by minimising some loss on previously unseen instances).

We should also note what we are **not** trying to do with this line of work. We are not proposing a single vocabulary that we hope everyone adopt. Our intention is merely to survey commonly used terms in part of machine learning to better understand why those terms are used.

2 Anatomy of a Learning Problem

With the *data*, *model*, *assessment* taxonomy of learning problems in place, we now propose an tentative refinement to build an *anatomy* of learning problems. Many of these terms were chosen to echo existing usage but are deliberately vague or under-specified to allow for broad applicability. The hope is that these place-holders will take on more well-defined meaning over time. The aim is not to prescribe a new vocabulary but rather to offer a starting point for discussing how we talk about machine learning problems.

2.1 Data

The data available for a learning problem is of central importance for applied machine learning. We will speak of **data** as a collection of **instances** each with an associated **training signal** (typically in the form of labels) represented in a common **format**.

Formats describe how data is represented. Generally speaking, many learning problems use **features** to describe instances. These can be thought of as projections of the instances down to simple values such as numbers or categories. Another common format is the **relational matrix**. This can be thought of as describing features of pairs of instances of the same kind (e.g., kernel matrix) or between different types of instances (e.g., viewers and movies in collaborative filtering). Both feature vectors and matrices can be either **sparse** or **dense** depending on whether every entry is explicitly described or not.

2.2 Models

A **model** is used to describe the class of possible results obtained by applying a learning algorithm to data. It can be thought of as a set of constraints on the class of hypotheses a learner can make about the data (e.g., linear models).

An important feature of a model is the form of its **predictions**. Typically, these are the same form as the training signal in the data but may not be. For example, in class probability

estimation, a data set contains a categorical training signal but predictions are probability distributions over those categories.

2.3 Assessment

Once a model is created it must be **assessed** in some way. This is commonly achieved by applying the model to some new data and evaluating its predictions against the novel training signal using a **loss** – a function that assigns a penalty to prediction-signal pairs.

2.4 The Learning Process

With the main components of a learning problem sketched, we can turn to describing common patterns or procedures for solving a learning problem.

We first introduce some general terminology to talk about the objects or *resources* of a learning problem and how they interact with each other in various *phases* of a solution.

The resource that *solves* a learning problem by constructing a model given some data is called a **learner**. The application of a learner to data to create or update a model is called the **training** phase. A **prediction** phase is when a model is applied to an instance to make a prediction. The assessment of a model is performed in the **evaluation** phase.

During the training phase, the access a learner has to the training signal can be described as **supervised** (a training signal for every instance), **semi-supervised** (signal for some instances but not all), or **unsupervised** (no training signal). In **active** access to the training signal is “pull” rather than “push” in that the learner can request it for specific instances rather than being given them all.

In addition to the training signal, the learner may have some access to the way its models will be assessed. This will be called **feedback**. In **full feedback** problems the learner can request an assessment of a model’s predictions during training. In some problems (e.g., bandit problems) the learner only has **partial feedback**. In these cases, the loss for certain predictions is not available.

2.5 Modes

The various phases of interaction between resources can take place in several different ways. **Modes** are used to described the relationship between phases. Modes can be used as modifiers when describing learning phases.

Batch training, prediction, and evaluation phases occur independently of each other. In **online** learning training, prediction, and possibly evaluation are interleaved in blocks of one or more instances. The **inductive** and **transductive** modifiers apply to the training phase of learning and describe whether the learner has access to the data its models will be assessed upon (transductive) or not (inductive).

3 A Survey of Some Existing Terminology

We briefly survey two repositories and two services for their use of terminology for describing various aspects machine learning problems and their solutions. Our focus will be on the language used by each to describe and distinguish between different classes of learning problem (e.g., classification, regression, etc.). Of particular importance are the terms used to describe the different dimensions learning problems can be placed upon.

3.1 The UCI Repository

This venerable machine learning repository has been available in various forms since 1987[5]. Currently, it holds 189 data sets for a variety of problem types, including classification, regression and clustering problems. In the default table view of the repository, entries are described by their *Name*, *Data Type*, *Default Task*, *Attribute Type*, *Number of Instances*, *Number of Attributes*, and *Year*. Additionally, the search criteria also add *Area* and *Format Type*. The term *Area* denotes an application domain (e.g., "Life Sciences", "Business", "Game", etc.) while *Format* is either "Matrix" or "Non-Matrix". Of these, the most pertinent to classifying problems are *Data Type*, *Default Task*, and *Attribute Type*.

The listed categories in the *Default Task* field are "Classification" (126 data sets), "Regression" (16), "Clustering" (8), and "Other" (44). Under "Other" are tasks described as "Causal-Discovery", "Recommender-Systems", "Function-Learning", and "N/A".

Attribute Types describe the values in instances' features as well as their output type. These include "Integer", "Categorical", "Real", and combinations thereof. A total of 24 out of the 189 data sets do not have an entry for *Attribute Types*. This is occasionally due to lack of documentation but also occurs for data sets with instances that are not easily described in attribute-value format, such as relational problems such as finding domain theories.

Data Types can be loosely described as denoting a class of problem. Entries include "Data-Generator", "Domain-Theory", "Image", "Multivariate", "Relational", "Sequential", "Time-Series", "Spatial", "Text", "Univariate", "Transactional" and combinations of some of those terms (e.g., "Text, Sequential"). The *Format Type* is one of either "Matrix" or "Non-Matrix".

3.2 MLData

The MLData service at `http://mldata.org` describes itself as "a repository for your machine learning data". It allows its users to create four entities that are stored in the repository: *data* ("Raw data as a collection of similarly structured objects"), *methods* ("Descriptions of the computational pipeline"), *tasks* (a data set plus some performance measure to optimise on the data), and *challenges* ("Collections of tasks which have a particular theme").

At the time of writing there are 27 publicly available tasks at MLData which fall into three *task types*: "binary classification", "multiclass", and "regression". A task specifies an input and

output format (e.g., “real-valued matrix” to “+1/–1”), a *performance measure* (e.g., “accuracy”), a data set, and a split of the data set’s variables into input/output and instances into train/validation/test. Of the 21 binary classification tasks, 20 use “accuracy” as the performance measure and one uses “ROC curve”; both multiclass problems use “accuracy”; and three of the four regression tasks use “root mean square error” with the other using “mean absolute error”.

Under *methods* there are currently 9 entries, all of which describe (to varying degrees of completeness) the application of a particular software package to one or more tasks using a particular configuration of learning parameters.

3.3 Google Prediction API

This project[6] offers a web service API to a number of (undisclosed) learning algorithms. Users can upload data, use it to train a predictors, then make predictions on new data.

In the Google Prediction API several problem-related terms are used with a specific meaning. *Data* refers to a tabular representation of instances in CSV format that is uploaded to Google’s data storage service. Each column of the data table is a *feature* that can take values that are exclusively either numerical or strings. The first column of the data is the *target value* and can be either numerical (defining a *regression* problem) or a string (for *classification* problems). Data for classification problems can specify an optional *utility* for each example. The training service will aim to predict better on higher utility examples.

After the data is uploaded, a *model* can be trained by sending a URL for the data to the service. Depending on the type of the target value in the data set, either a *regression* or *categorical* model is returned along with an evaluation of the model’s classification accuracy (for categorical models) or mean square error (for regression models). The evaluation is generated via cross-fold validation.

Once trained, categorical models (but not regression models) can be updated with additional training examples. This is referred to as *streaming training*.

3.4 MLComp

The MLComp service[7] describes itself as “a free website for objectively comparing machine learning programs across various datasets for multiple problem domains.” At the time of writing the service hosts 382 *data sets*, 325 *programs*, 10 *domains*, 3 *domain types*, and has performed 9774 *runs*.

Unlike the Google Prediction service, MLComp admits a range of problem types than just classification and regression. In particular, they categorise problems into *domains* which can be solved via conforming *programs* during a *run*. Domains are classes of problems “equipped with the following”: “a particular *dataset format*”, “a particular *program* interface”, and “standard *evaluation metrics*”. A user of the MLComp service can define their own domain by specifying these properties via a structured markup language (YAML).

Domains are further categorised into three broad *domain types*. In *supervised learning* a run consists of a *learn* and *test* phase. In a *performing* domain (e.g., clustering, optimisation), a run just applies a program to a single data set. In an *interactive learning* domain (e.g., online learning and bandit problems), the training and evaluation phases are interleaved so that during a run the program repeatedly receives an unlabelled instance, makes a prediction, then the label is revealed.

The MLComp framework also recently added support for *reductions* between classes of learning problems. As described by Beygelzimer et al.[8], these are techniques for transforming one type of problem (e.g., mutliclass classification) into others (e.g., binary classification) in a way that provides some guarantees regarding generalisation performance.

4 Summary and Conclusions

How useful are the introduced terms and questions in emulating the surveyed projects' systems of categorising machine learning problems? Obviously since the terminology we have introduced is generally at a higher level than that used in particular projects, any re-expression of those projects' terms will not capture the same detail.

What the UCI repository refers to as a *data type* does not easily fit into our ontology. For example, we note that since the repository mainly holds data sets, much of our process-oriented terminology — that is, phases and modes — is not applicable.

What the MLComp project calls a *domain* is what we would described by a combination of *format*, *mode* and *feedback* specification. For example, in `BinaryClassification` data in a *sparse*, *feature vector* format is presented with a *binary* label *supervised* signal in *batch* mode assessed using a *misclassification* loss. `WordSegmentation` is an *unsupervised batch* trained problem with English sentences as instances presented in *text* (UTF8) format assessed using *precision*, *recall*, and *F1* losses.

4.1 Anatomical Viewpoints

Just as medical specialists can be dermatologists, neurosurgeons, and heart specialists, so too can machine learning researchers and practitioners be placed into a number of categories, each with their own jargon. A big difference, however, between medicine and machine learning is that medical students are trained broadly so that specialists have many common points of reference.

Can the type of work various people do under the auspices of machine learning be describe with reference to the parts of machine learning problems they most emphasise? The following caricatures are intended to highlight the ways in which researchers differ in their focus.

Applied research tends to ask "What can be learnt from this *data*?" and will use whatever models and assessment procedure is most appropriate. Research into new machine learning **techniques** is arguably *model*-focused, developing efficient algorithms that perform

well on particular data sets for a narrow range of assessments. Learning **theory** is arguably *assessment*-focused, typically ignoring the finer details of data representation and aiming to show when classes of problems are learnable or not independent of particular techniques.

4.2 Future Work

This proposal barely scratches the surface of the rich and complex range of learning problems that are encountered in the literature. A more comprehensive study would include an analysis of the APIs of various machine learning toolkits in an attempt to understand the implicit ontologies are used by the algorithms they implement. An extension of the current approach to a broader range of problems such as reinforcement learning, dimensionality reduction, and other classes is also planned.

References

---

1. *MLComp*, http://mlcomp.org ↩
2. Iverson, K.E., *Notation as a tool of thought*, Communications of the ACM, vol. 23, 1980. ↩
3. Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, 1983 ↩
4. Mitchell, T., *Machine Learning*, McGraw-Hill, 1997 ↩
5. Asuncion, A. & Newman, D.J. UCI Machine Learning Repository http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science. 2007 ↩
6. The *Google Prediction API*, http://code.google.com/apis/predict/ ↩
7. *MLComp*, http://mlcomp.org ↩
8. Beygelzimer, A. and Langford, J. and Zadrozny, B., *Machine learning techniques — Reductions between prediction quality metrics*, In Performance Modeling and Engineering, Springer, 2008 ↩