Probability Model Type Sufficiency

Leigh J. Fitzgibbon, Lloyd Allison and Joshua W. Comley

School of Computer Science and Software Engineering
Monash University, Victoria 3800, Australia
{leighf,lloyd,joshc}@bruce.csse.monash.edu.au

Abstract. We investigate the role of sufficient statistics in generalized probabilistic data mining and machine learning software frameworks. Some issues involved in the specification of a statistical model type are discussed and we show that it is beneficial to explicitly include a sufficient statistic and functions for its manipulation in the model type's specification. Instances of such types can then be used by generalized learning algorithms while maintaining optimal learning time complexity. Examples are given for problems such as incremental learning and data partitioning problems (e.g. change-point problems, decision trees and mixture models).

1 Introduction

The formal specification of a statistical model type is an important ingredient of machine learning software frameworks [1]. In the interests of software reuse, robustness, and applicability the model type should encompass a general notion of a statistical model, and allow generalized machine learning algorithms to operate using any valid model type while still maintaining optimal time complexity. The model type should encourage and facilitate the decoupling of learning algorithms from the statistical models that they operate over, allowing the two to be plugged together as required depending on the problem and data involved. An intuitive example is that of a generalized decision tree algorithm that can operate with any valid model type in the leaves.

Sufficient statistics play an important role in reducing the time complexity of many machine learning algorithms. In this paper we investigate how the explicit inclusion of sufficient statistics in a model type allows machine learning algorithms to be generalized while maintaining optimal time complexity.

Code examples are given in the Haskell [4] programming language using a fixed width font. Footnotes are used to explain some of the semantics of the Haskell code.

2 A Basic Model Type

In this section we define the basic requirements for a model type that can be used for inference or learning. To perform probabilistic learning we need either the log-probability density/distribution function[1]:

```
lpr :: theta -> x -> Double
```

or the log-likelihood function[2]:

```
ll :: [x] -> theta -> Double
```

Since these can be derived from one another, having either the log-likelihood or log-probability function would allow one to do likelihood based inference. However, we are also interested in the time complexity of the basic model type and the implications this model type has on a system that is built around such a type. Note the difference in the order that the parameters are given in the two (curried) functions. The log-likelihood function requires the data-set as its first parameter, whereas the log-probability function requires the model parameters as the first parameter. This is because the log-likelihood function by definition is used for calculating *the log-likelihood of the parameters* whereas the probability function is used to calculate *the log-probability of the data*. These are conceptually two very different things. The log-likelihood function will be evaluated for different values of theta (e.g. numerical integration of parameters; simulation; optimisation - maximum likelihood, maximum posterior, minimum message length [7], minimum description length [6]), whereas we will generally evaluate the log-probability function for varied values of the data (e.g. numerical integration; optimisation - mode; moments; numerical entropy).

The log-likelihood function is used for inference, and if time complexity is of any importance then it needs to evaluate efficiently since it will be called frequently. Therefore the order of the parameters is important, the data must come first, for which a [minimal] sufficient statistic can be computed and stored, then the log-likelihood can be evaluated in optimal time for each call. It follows that a basic model type that explicitly includes sufficient statistics requires at least the following functions:

```
data ModelType x theta s = Model {
  lpr :: theta -> x -> Double,
  ss :: [x] -> s,
  ll :: s -> theta -> Double
}
```

[1] I.e. a function that given an element from the parameter space, and an element from the dataspace, returns the log-probability of the data element given the parameter.

[2] I.e. a function that given a vector of elements from the dataspace, and an element from the parameter space, returns the log-probability of the data elements given the parameter.

where `ss` is a function for computing the sufficient statistic for a given data-set.

There are many other operations that could be built into the model type. There are also important functions that are clearly distinct from model types yet are related. For example an estimator function with type:

```
type Estimator theta s = s -> theta
```

In the following sections we will show that having sufficient statistics as an explicit part of the model type allows for efficient implementations of a large class of popular algorithms.

3 Incremental Learning

Assume we initially observe some data $\boldsymbol{x}_1$ and then estimate the model parameters according to some criterion (e.g. maximum likelihood). We might then observe more data, $\boldsymbol{x}_2$, which is assumed to come from the same distribution. We then wish to revise our parameter estimate in light of the new data. We therefore wish to update the sufficient statistic to take into account the new data. The type of such a function would be:

```
type AddSs x s = s -> [x] -> s
```

Once the sufficient statistic has been update we can evaluate the likelihood function in best case time for the given sufficient statistic. If minimal sufficient statistics are used then these will typically be in the order of the dimension of the number of parameters being estimated.

More generally, we observe M data-sets incrementally $\{\boldsymbol{x}_i : i = 1..M\}$ for which we make M estimates $\{\theta_i : i = 1..M\}$. Let the length of data-set $\boldsymbol{x}_i$ be denoted by N_i. We wish to estimate the parameters, θ_i, for each $\boldsymbol{x}_i$ using a known estimator function. Based on the data types defined so far we can describe the type of this incremental learning function (`incLearn`):

```
incLearn :: [[x]] -> (ModelType x theta s) ->
            (Estimator theta s) -> (AddSs x s) -> [theta]
```

which returns the vector of inferred parameters.

The `incLearn` function is able to perform incremental learning for any model type in optimal time[3].

4 Learning Models that Partition the Data

A large class of useful and popular inference algorithms partition the data-set into a pairwise disjoint set. Examples are change-point, decision tree, and mixture model algorithms. These algorithms can be efficiently implemented by storing a sufficient statistic for each subset and then updating the statistic as the subsets change (i.e. as data is added and removed). So as well as the `AddSs` function defined in the previous section, we also require a `RemoveSs` function:

[3] We ignore the function call overhead in the implementation language.

```
type RemoveSs x s = s -> [x] -> s
```

which allows for the removal of a set of data from a sufficient statistic.

Change-Point Algorithms Change-point algorithms are quite common and can be found as the building blocks of more complex algorithms (e.g. decision trees). There are many algorithms for change-point problems that benefit from sufficient statistics. One example is the approximate binary search algorithm where the optimal location for a single change-point is found and then this is repeated recursively on each segment until some stopping criterion is reached. Another example is the dynamic programming algorithm (see e.g. [3]) which can identify the optimal change-point locations.

Using the types defined so far, a multiple change-point algorithm could be implemented that operates with arbitrary models in the segments. The algorithm could achieve optimal time complexity using the `AddSs` and `RemoveSs` functions.

Mixture Modelling There is some research into versions of the EM algorithm that do not require all of the data to be visited. For these algorithms the use of sufficient statistics can improve the algorithm's time complexity. One example can be found in [5] where k-d trees and sufficient statistics are used to improve the time complexity of the algorithm for a specified class of distributions.

5 Revised Model Type

Given that incremental learning is a generic operation, and that data partitioning algorithms are extremely common one may consider including the update functions (`addSs` and `removeSs`) in the model type. There appears to be no downside to the inclusion of these functions in the model type especially since the data itself is a sufficient statistic and therefore a default implementation could simply manipulate the data. The Haskell code for the revised model type becomes:

```
data ModelType x theta s = Model {
  lpr :: theta -> x -> Double,
  ss :: [x] -> s,
  ll :: s -> theta -> Double,
  addSs :: s -> [x] -> s,
  removeSs :: s -> [x] -> s
}
```

For a large class of distributions (e.g. the Exponential Family [2]) the `addSs` and `removeSs` functions will be trivial, only involving addition and subtraction.

6 Parallelism

In probabilistic machine learning algorithms the likelihood function is the major (if not the only) link to the data. Through use of sufficient statistics we replace the high-dimensional data with a possibly low dimension sufficient statistic. If the algorithm is parallelised then the low dimensional sufficient statistic need only be transmitted between processors (instead of the data) thus reducing communication time as well as improving time complexity (over transmission and inference using the raw data). The revised model type defined in the previous section facilitates the use of sufficient statistics in generalized parallel algorithms.

7 Conclusion

We have investigated the explicit use of sufficient statistics in the definition of a statistical model type for machine learning software frameworks. A model type was defined and several examples explored in the functional language Haskell. The type explicitly includes a sufficient statistic which allows for the implementation of a class of generalized machine learning algorithms that have optimal time complexity.

References

1. L. Allison. Types and classes of machine learning and data mining. In M. Oudshoorn, editor, *Proceedings of the Twenty-Sixth Australasian Computer Science Conference*, volume 16, pages 207–215, February 2003.
2. J. M. Bernardo and A. F. M. Smith. *Bayesian Theory.* Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, Chichester, 1994.
3. L. J. Fitzgibbon, L. Allison, and D. L. Dowe. Minimum message length grouping of ordered data. In H. Arimura and S. Jain, editors, *Proceedings of the Eleventh International Conference on Algorithmic Learning Theory (ALT2000)*, volume 1968 of *Lecture Notes in Artificial Intelligence*, pages 56–70, Berlin, 2000. Springer-Verlag.
4. P. Hudak and J. H. Fasel. A gentle introduction to Haskell. *SIGPLAN Notices*, 27(5), May 1992.
5. A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, December 1998.
6. J. J. Rissanen. Hypothesis selection and testing by the MDL principle. *Computer Journal*, 42(4):260–269, 1999.
7. C. S. Wallace and D. L. Dowe. Minimum message length and Kolmogorov complexity. *The Computer Journal, Special Issue - Kolmogorov Complexity*, 42(4):270–283, 1999.