PROCEEDINGS OF SCIENCE

Support Vector Machines and Generalisation in HEP

J. Hays*
Queen Mary University of London
E-mail: j.hays@qmul.ac.uk

A.J. Bevan,
Queen Mary University of London
E-mail: a.j.bevan@qmul.ac.uk

T.J. Stevenson
Queen Mary University of London
E-mail: t.j.stevenson@qmul.ac.uk

The concept of Support Vector Machines is briefly reviewed and their potential use in high energy physics scenarios is discussed. They have the potential to be less susceptible to over-fitting than artificial neural networks and boosted decisions trees - whose use is common across high energy particle physics. Cross validation is discussed in the context of improving generalisation of machine learning algorithms and preliminary work on bringing more rigor to understanding generalisation in HEP analysis is presented.

38th International Conference on High Energy Physics
3-10 August 2016
Chicago, USA

*Speaker.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).

PoS(ICHEP2016)976

1. Introduction

Modern high energy physics (HEP) data analysis makes heavy use of machine learning to optimise the sensitivity of measurements. The current fashion is for artificial neural networks and boosted decision trees, though other approaches involving deep learning are starting to be investigated. Another algorithm that has seen widespread use outside of HEP but only modest use [1] within the field is that of support vector machines (SVM) [2]. These were originally developed in the 1960s with the current standard approach being proposed in 1995. One aspect that is currently under investigation is whether SVMs provide for better performance with respect to generalisation when working with relatively small training data sets.

2. Support Vector Machines

Support vector machines attempt to define an optimal hyperplane decision boundary for some set of training data in a higher dimensional feature space. In the simplest case, where the training set $\mathbf{x}$ is linearly separable, a hyperplane can be defined of the form: $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ where the weight vector $\mathbf{w}$ is perpendicular to the plane and b the bias determines the distance of the plane from the origin. A decision function can then be defined as $y_i = \text{sign}(\langle \mathbf{w} \cdot \mathbf{x} \rangle)$ and classifies the points x_i in the training set according to the side of the hyperplane they are on. The weight vector $\mathbf{w}$ and the bias b represent the tunable parameters of this linear discriminant. Without any further constraints there will in general exist a continuum of solutions for $\mathbf{w}$ and b for a given linearly seperable training set. Defining a geometric margin a distance γ either side of the hyperplane, an optimal solution can be defined by finding the values of $\mathbf{w}$ and b that maximise the value of γ where no data points lie within the margin. It can be shown that only the data points that lie exactly on the margin in the optimal case are needed to define the hyperplane - these are the so-called support vectors. However, where the training data may be noisy this form of classifier is very susceptible to over-fitting. Furthermore, for many problems the training data are not linearly separable and there is no guarantee on the convergence of an algorithm to find the optimal solution. The use of linear discriminants as above is therefore very limited in the real world. This can be dealt with by sacrificing the simplicity and useful properties of linear models or by building multiple layers of linear models with thresholds such as in artificial neural networks. An alternative to this is to instead to map the training space X into a kernel-induced feature space F: $\mathbf{x} = (x_1, x_2, \ldots, x_n) \mapsto \phi(\mathbf{x}) = \phi(x_1), \phi(x_2), \ldots, \phi(x_n)$. Where the kernel is defined as: $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \rangle, \forall \mathbf{x}, \mathbf{y} \in X$. The basic idea is that with the appropriate choice of kernel the problem can be expressed in a way that is tractible to the linear approach described above. This does not necessarily lead to a linearly separable problem so a further extension is needed. The introduction of slack parameters and an associated cost function that allow for points to be misclassified provides a means to define an optimal solution that can be found with a convergent algorithm.

One recurring problem that is seen in several data analyses at the LHC experiments arises from a limitation on the size of samples of simulated data available to train machine learning algorithms. It would be of huge benefit to discover new (to HEP at least) techniques that can do more with less training data. Work is currently underway in our group to investigate the relative performance

PoS(ICHEP2016)976

SVMs, see for example [3], with other established techniques in HEP to understand if they offer advantages when dealing with small samples of training data.

Our group has provided updates to TMVA [4] to improve the implementation of SVMs, adding additional kernel functions and aids to hyperparameter tuning.

3. Generalisation

A generalised algorithm is one whose performance in correctly categorising unseen data is compatible with its performance on the training samples. Increasing the model complexity - for example increasing the tree depth in boosted decision trees - is a relatively easy way to improve the performance as measured on the training sample. Ultimately, though, it leads to the problem of *over-fitting*, where the model becomes very good at fitting features of the training set where these may have arisen through fluctuations and noise rather than something fundamental about the data. The algorithm perfomance in correctly describing or categorising unseen data can then be degraded.

A very common approach in widespread use in HEP is a variant of the *hold-out* method [5], where the data are split into two samples (that need not be equal in size). One of these samples is used as the training set, and the other for testing the performance. Typically the algorithms employed have as their output some continuous or quasi-continuous value whose distribution can be compared between the training set and the testing set. If the output distributions are compatible then the algorithm is declared as generalised.

A first look at how a better understanding of how generalised the particular application of an algorithm is to a specific problem has been carried out and is described briefly here. A simple toy model has been constructed of a new particle resonance search with one dominant background and a number of simple kinematic observables that distinguish between signal and background. The average performance of a boosted decision tree under different sets of training parameters was estimated on a training sample of 2000 events and a statistically independent testing sample, also with 2000 events. The metric of performance was the statistical significance of the observed signal. The average performance together with its variance was estimated for both the training and test samples using a stratified jackknife resampling with 50 strata. A total of nine different sets of training parameters, numbered sequentially from 0 to 8, with the maximum tree depth increasing incrementally from 2 for data set 0 to 10 for data set 8. This was done with the intention of forcing a case of severe over-fitting.

The results of this can be seen in Figure 1. The black points show the average performance of the algorithm for each of the 9 training points and the band shows the estimated standard deviation when applied to the test sample. For the red points, showing the performance on the training set - only the average is shown. The difference in the performance on the training and test samples shows the over-fitting very clearly. The extremely high performance on the training samples relative to the test sample arises as a result of the BDT becoming overly good at describing features of the training set that are not reflected in the test data set. The key feature that has been introduced here beyond the basic hold-out method often used in HEP analysis is the estimate of the training variance. This gives context to the difference between the training and test results, providing a way to judge the significance of the difference. Future work will look at alternative measures of performance and

PoS(ICHEP2016)976

estimates of variance and how the size of the training set and choice of algorithm impact on the process.

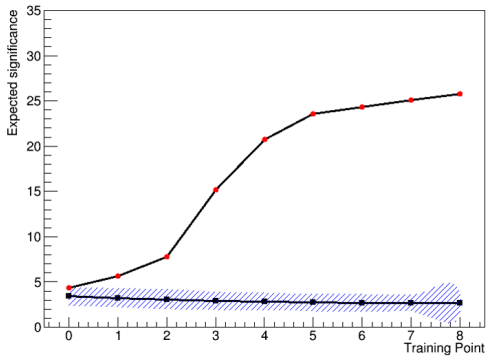

Figure 1: The points show the performance of a BDT in terms of signal signifance for a toy new-physics model across nine different training points. The red and black points show the performance on the training and test samples respectively. The band around the black points shows an estimate of the training variation.

4. *k*-fold Cross Validation

One problem of using the hold-out method is that it requires the sacrifice of a significant fraction of the available data that will not be used in training. When training data are not very abundant this can lead to problems. Another approach to validating how a given analysis approach will generalise to an unseen data set is *k*-fold cross validation [5, 6].

In this approach the data are split into k distinct equal-sized sub-samples. Then the chosen algorithm is trained k times, removing (with replacement) in each training a different sub-sample - or fold - from the whole data set. The fold that has been removed is then used for validation. The set of k results can then be averaged to provide an estimate of the average expected performance of the algorithm. The choice of k is problem specific but $k = 10$ is commonly used. The choice k=2 recovers the hold-out method.

As part of our work in this area we have introduced *k*-fold cross validation to TMVA.

References

[1] A. Vossen, *Support vector machines in high energy physics*, arXiv:0803.2345.

[2] C. Cortes and V. Vapnik, *Support-vector networks*, *Machine Learning* **20** (1995) 273–297.

[3] A. Bethani, A. J. Bevan, J. Hays and T. J. Stevenson, *Support vector machines and generalisation in hep*, arXiv:1610.09932.

[4] A. Hoecker et al., *Tmva, the toolkit for multivariate data analysis with root*, *PoS ACAT* **040** (2007) .

[5] L. Devroye and T. J. Wagner, *Distribution-free performance bounds for potential function rules*, *IEEE Transactions in Information Theory* **25** (1979) .

[6] M. Stone, *Cross-validatory choice and assessment of statistical predictions*, *J. Roy. Statist. Soc. Ser. B* **36** (1974) 111147.

PoS(ICHEP2016)976