Higgs Boson Machine Learning Challenge - Notes

João Alexandre Mateus Luna António (joaoantonio@ua.pt)

December 2018

1 Models from Classification Learner App

In this section we are going to explore various types of ML algorithms, and compare their values of Accuracy, percentage of background predicted as signal (100% would mean that all the background was predicted as signal in the training data), percentage of signal predicted as signal and the AMS score (calculated like is represented in section **??**). After presenting a table with value for every type of classifier we also present a quick overview of what the classifier does based on the approaches of [6], [7] and [8] after that we present an even simpler version, a kindergarten version for every type of classifier, here we lose some of the veracity of the words to make it even simpler to understand.

All the models were tested with 10% of the total data set called training, and the percentages of B predicted as S, S predicted as S and AMS are calculated on this 10% (250000 events in total and 90% were used to do the training. Choosing smaller percentages, could lead to "over-fitting", this property happens when the model corresponds too closely to the data, obtaining a complete "memory" and loosing robustness against possible errors in the training data.

As we didn't use 100% of the data to calculate the AMS, we need to correct the s_e and b_e that we obtain equations 1 and 2.

$$s_e = \sum_{i=1}^{s_{predicted}} w_i \frac{w_{s\ in\ 100\%}}{w_{s\ in\ 10\%}} \tag{1}$$

$$b_e = \sum_{i=1}^{b_{predicted}} w_i \frac{w_{b\ in\ 100\%}}{w_{b\ in\ 10\%}} \tag{2}$$

This Algorithms were using the Classification Learner App from Matlab [6], and every tweak made to make it differ from the default ones, is documented. By the fact of being models already created, they minimise the false positives but also the false negatives.

1.1 Discriminant Analysis

Type Discriminat	Accuracy (%)	B Predicted as S (%)	S Predicted as S (%)	AMS
Linear	74.2	13.37	51.26	1.8730
Quadratic	73.0	15.84	52.62	1.7011

Table 1: Models of Discriminant Analysis, values of Accuracy, B predicted as S, S Accuracy and AMS

The representation of Discriminant Analysis consists of statistical properties of your data, calculated for each class. For a single input variable this includes:

- The mean value for each class.

- The variance calculated across all classes.

Predictions are made by calculating a discriminate value for each class and making a prediction for the class with the largest value.

The technique assumes that the data has a Gaussian distribution (bell curve). The result of this type of classifier is in table 1.

One kindergarten version of thinking the problem, if we image that the data only has two dimensions, we want to create a separating line (usually called decision line), all the events that are in one side of the line are classified as one class, all the others, the other class.

The main issue with this view, is that, this version also explains the "Naive Bayes classifier" and the "Logistic Regression Classifier", the difference between these three types of classifiers is the functions that they use to create the decision line.

1.2 Logistic Regression Classifier

Type Logistic	Accuracy (%)	B Predicted as S (%)	S Predicted as S (%)	AMS
Regression	74.9	13.93	53.75	2.0162

Table 2: Model of Logistic Regression, values of Accuracy, B predicted as S, S Accuracy and AMS

Logistic regression is like linear regression in that the goal is to find the values for the coefficients that weight each input variable.

Unlike linear regression, the prediction for the output is transformed using a non-linear function called the logistic function.

The logistic function looks like a big S and will transform any value into the range 0 to 1. This is useful because we can apply a rule to the output of the logistic function to snap values to 0 and 1 (e.g. IF less than 0.5 then output 1) and predict a class value.

Because of the way that the model is learned, the predictions made by logistic regression can also be used as the probability of a given data instance belonging to class 0 or class 1. This can be useful on problems where you need to give more rationale for a prediction, or where we want to use various classifiers together (ensemble). The result for this classifier is in the table 2.

1.3 Nearest Neighbour Classifier

The model representation for KNN is the entire training dataset. Predictions are made for a new data point by searching through the entire training set for the K most similar instances (the neighbours) and summarising the output variable for those K instances. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value.

Type KNN	Accuracy (%)	B Predicted as S (%)	S Predicted as S (%)	AMS
Euclidian 150 Neig.	79.6	13.56	67.09	2.4984
Euclidian 100 Neig.	79.9	13.34	67.08	2.5059
Euclidian 10 Neig.	79.3	11.46	61.92	2.4048
Euclidian 3 Neig.	77.1	16.89	65.54	2.0713
Euclidian 1 Neig.	74.6	21.27	63.07	1.6929
Cosine 150 Neig.	79.7	12.85	65.98	2.5426
Cosine 100 Neig.	80.1	12.45	66.00	2.5528
Cosine 10 Neig.	79.4	10.90	61.66	2.4337
Cosine 3 Neig.	77.4	16.47	65.30	2.0844
Cosine 1 Neig.	74.8	20.75	62.67	1.6876

Table 3: Models of Nearest Neighbour Classifier, values of Accuracy, B predicted as S, S Accuracy and AMS

The trick is in how to determine similarity between the data instances. The simplest technique if your attributes are all the same scale is to use the Euclidean distance (3), a number you can calculate directly based on the differences between each input variable, other also very simple metric to use to use is the Cosine (4).

The Euclidean methods use a Euclidean distance that can be defined by:

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3}$$

The cosine uses a cosine metric that has the similarity (equivalent to distance) be defined by:

$$\frac{x \cdot y}{\sqrt{x \cdot x}\sqrt{y \cdot y}} \tag{4}$$

KNN can require a lot of memory or space to store all the data, but only performs a calculation (or learn) when a prediction is needed, just in time. You can also update and curate your training instances over time to keep predictions accurate. The results are in table 3.

The kindergarten version of the KNN classifier, once again suppose a two-dimensional data, first we place a few events in their correct coordinates. The new events, go to their coordinates and upon arriving "communicate" with their neighbours, and join the same class that most of their neighbours has.

1.4 Decision Trees

Type Tree	Accuracy (%)	B Predicted as S (%)	S Predicted as S (%)	AMS
4 Nodes	77.9	15.64	67.74	2.2365
20 Nodes	80.0	13.59	68.77	2.3600
100 Nodes	81.1	14.26	72.22	2.3723

Table 4: Models of Decision Trees, values of Accuracy, B predicted as S, S Accuracy and AMS

The representation for the decision tree model is a binary tree. Each node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric).

The leaf nodes of the tree contain an output variable (y) which is used to make a prediction. Predictions are made by walking the splits of the tree until arriving at a leaf node and output the class value at that leaf node. The results are in table 4.

Decision trees usually yield more accurate predictions when used in an ensemble.

For the kindergarten version of the decision tree imagine once more the two-dimensional data, a map where some events are already classified. To place a new event in the map, we first must discover what is his first coordinate, and then the second, and only after we know both things, we can place a new event on the map. The catch of this classifier is that instead of asking what is the coordinate, we can only ask yes or no questions.

1.5 Ensemble Classifiers

Type Ensemble	Accuracy (%)	B Predicted as S (%)	S Predicted as S (%)	AMS
AdaBoost Tree	81.7	11.29	68.90	2.5571
Subspace Discriminat	71.8	8.80	38.26	1.5152

Table 5: Models of Ensemble Classifiers, values of Accuracy, B predicted as S, S Accuracy and AMS

AdaBoost is used with short decision trees. After the first tree is created, the performance of the tree on each training instance is used to weight how much attention the next tree that is created should pay attention to each training instance. Training data that is hard to predict is given more weight, whereas easy to predict instances are given less weight.

Models are created sequentially one after the other, each updating the weights on the training instances that affect the learning performed by the next tree in the sequence.

After all the trees are built, predictions are made for new data, and the performance of each tree is weighted by how accurate it was on the training data.

We also do the calculations for a discriminant ensemble, where the method for creating the AdaBoost tree is replicated but instead of using tree, now uses discriminant methods. The results of the Ensemble Classifiers can be seen in Table 5.

The last kindergarten version is for the ensemble classifier, here we use various classifiers to make the decision, imagine someone stuborn, that only agrees if we do various classifiers, and the value of every classification depends on the value of the previous ones and the hardness of that classification.

1.6 Conclusions

As expected, they don't get values of AMS very close to the winners, still the best ones have between 2.4 and 2.6 of AMS, that suggest that an ensemble using KNN methods would be a very good solution, or even with the Ada Boost methods, and it's what happens since most of the winner solutions involve at least one of these methods.

One interesting properties of the KNN methods is that for both metrics, the best results were for near one hundred neighbours, that means that with less neighbours we would classify poorly the events, this is not common in this type of problem, usually with more neighbours, the result get worst.

References

[1] Adam-Bourdarios, Claire, et al. "Learning to discover: the higgs boson machine learning challenge." (updated 21st July 2014) `http://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf`

[2] Smola, Alex and S.V.N. Vishwanathan. Introduction to Machine Learning

[3] Radovic, Alexander, et al. "Machine learning at the energy and intensity frontiers of particle physics." Nature 560.7716 (2018): 41.

[4] Wikipedia article for the Higgs boson `https://en.wikipedia.org/wiki/Higgs_boson`

[5] Johnston, Hamish. The Physics World Article named "Higgs boson seen decaying to two bottom quarks" of 9 Jul 2018
https://physicsworld.com/a/higgs-boson-seen-decaying-to-two-bottom-quarks/

[6] Mathworks Team, Documentation from matlab in how to choose and program a classifier
https://www.mathworks.com/help/stats/choose-a-classifier.html

[7] Brownlee, Jason . The Site Machine Learning mastery
https://machinelearningmastery.com/

[8] Gama, João, et al. "Extração de conhecimento de dados: data mining." (2015).

[9] Eliahou, Shalom. The $3x+1$ problem: New lower bounds on nontrivial cycle lengths.*Discrete Mathematics*, 118(1–3):45–56, 1993.