Higgs boson challenge

Martin Hanzel
martin.hanzel@epfl.ch
313710

Rastislav Kováč
rastislav.kovac@epfl.ch
309532

Martin Kostelanský
martin.kostelansky@epfl.ch
312818

***Abstract*—We developed a machine learning algorithm from scratch for the Higgs Boson Challenge. We implemented several common learning methods from scratch, and constructed a model using evolutionary feature selection and regularization taking into account incomplete data. While the performance of our model does not meet our expectations, it nonetheless yields between 72% and 82% classification accuracy, depending on the training data.**

I. Introduction

In this report, we describe our approach to build a learning model for the Higgs Boson Machine Learning Challenge. We implement feature engineering and machine learning methods from scratch in Python.

II. Feature engineering

A. Number of jets

From the dataset, and from the feature descriptions provided in Adam-Bourdarios *et. al.* [1], we observed that one feature, `PRI_jet_num`, has only four distinct values. Therefore, we treated it as a categorical feature, and split the training set into four different subsets on `PRI_jet_num`. We trained a model on each subset individually, and evaluated using the appropriate model during testing and validation.

We assumed that all other features are real-valued.

B. Data cleaning

The training data includes some indeterminate values, marked by value `-999`. We replaced these values with the feature mean.

After splitting the data into subsets by `PRI_jet_num`, we eliminated columns in each subset that were all `-999` (i.e. indeterminate) or all `0`, normalized each features to the normal distribution ($\mu = 0, \sigma = 1$), and added an all-`1` constant feature. Since we eliminated useless features, not all subsets end up with the same number of columns.

C. Evolutionary feature selection

We performed incremental feature augmentation by training a model in multiple epochs, starting from the cleaned original dataset. In each epoch, we augmented the feature matrix, trained a model using GD, and retained the 50% most significant non-constant features, i.e. the ones with the highest weights. Thus, the feature matrix "evolves" by dropping the columns that contribute the least to the prediction, reducing the runtime of the model and allowing us to consider more augmentations and make hyper-parameter search more efficient.

We augmented the features with different powers of the original data. Specifically, we raised data to powers 2, 3, and 4. We wanted to consider roots, inverses, and logarithms of features as well, but this was impossible to do globally because some samples included negative values. Given more time, we would have liked to analyze each feature individually and perform tailored feature augmentation per-feature, as appropriate.

D. Hyper-parameter selection

Hyper-parameters were λ (regularization constant) and γ (learning rate). We performed a binary search to find best value for λ. We have selected $\gamma = 1m$, where m is the GD iteration number.

III. Models and Methods

A. Regularization

We use the following term for regularization in the loss function, which is a variant of $L2$ regularization:

$$HKK(\vec{w}) = \left(\frac{\lambda}{1-\vec{\rho}}\right) \cdot \|\vec{w}\|^2$$

Where $\vec{w}$ is the weight vector, λ is the regularization constant, and $\vec{\rho}$ is a vector containing the fraction of invalid values for each feature. Specifically, feature i has fraction ρ_i of values being invalid. This approach penalizes models that rely heavily on poor features. We found that certain columns are very sparse with valid data, which may unfairly skew the model if those columns are considered in training. If there are no invalid data (i.e. $\forall i, p_i = 0$), this equation reduces to $L2$ regularization weighted by λ.

Let us call this method, *HKK regularization* (after the authors, for lack of a more succinct name).

B. Loss

We measure loss using logistic regression with mean-squared error regularized with HKK:

$$\mathcal{L}(\vec{y}, \mathbf{X}, \vec{w}) = \frac{1}{N} \sum_i (y_i - \sigma(\mathbf{X}_i \cdot \vec{w}))^2 + HKK(\vec{w})$$

where $\vec{y}$ is the vector of true labels, $\mathbf{X}$ is the data set with rows representing samples, $\vec{w}$ is the weight vector, and N is the number of samples.

C. Training

We trained four different models, one on each value of `PRI_jet_num`, using GD with the aforementioned evolutionary feature selection and hyper-parameter selection. 10-fold cross validation was performed at every training step, including each epoch of feature and hyper-parameter selection. Our model involves a very large parameter space that includes λ and the evolutionary selection of features and augments over several epochs, and it is time-consuming to search this space exhaustively. Thus, we used the following approach to select λ:

1) Using binary search, find a reasonable λ using only first-order features (no augmentation). We searched for $\lambda \in [0, 1]$.
2) Apply the evolutionary feature selection algorithm to select the features to use in the final model.
3) Using these features, perform a binary search again to find λ_2, which is used as the regularization constant in the final model.

Steps 2 and 3 may be repeated to presumably approach a configuration of features and λ that yields the highest precision.

It is not guaranteed that this method produces the optimal combination of features and λ_2, but we believe that it yields a suitable approximation for the scope of this project.

We chose to use classic GD instead of SGD because we observed a shorter time to convergence. This saved time outweighs the computational complexity on our machines. In fact, when we ran our model with SGD, we obtained worse accuracy for greater running time.

IV. Results

A. Control

Our "control model" consists of basic MSE logistic regression with $L2$ regularization and 10-fold cross-validation, trained on the whole dataset without grouping on `PRI_jet_num`. This model achieves a classification accuracy of about 69.2% on random validation sets.

B. Test

Our proposed model consists of MSE logistic regression with HKK regularization and 10-fold cross-validation, all on approximately 30 features selected by the evolutionary algorithm described above. We initially achieved 74.0% classification accuracy against the test set.

Interestingly, the `PRI_jet_num` $= 0$ subgroup performed marvelously, with 82.0% accuracy against validation sets, but all other subgroups achieved around 72% accuracy. It is clear that the `PRI_jet_num` $= 0$ subgroup is distinct and deserves special treatment.

Our algorithm initially selected $\lambda = 0.02$ and we selected $\gamma = \frac{1}{i}$ as hyper-parameters. We followed up the initial experiment with a grid-search on hyperparameter λ (regularization constant) with exponentially-increasing steps around the initial value. However, we failed to find a value for which the accuracy was significantly higher. Further, we found that the regularization term does not have a profound effect on the accuracy of the model. In two experiments, we fixed $\lambda = 0$ and $\rho = 0$, respectively, and found that classification accuracy was not affected very much.

The following table lists classification accuracy of the control model and our models. We show accuracy of the non-regularized, $L2$-regularized, and HKK-regularized evolutionary model.

Method	Accuracy
Control	69.2%
Evo. feature selection	77.07%
+ $L2$ regularization	77.06%
+ HKK regularization	77.06%

C. Selected model

From all models described above, we selected the Logistic model with HKK regularization. This model achieved 77% accuracy and F1-score $= 0.638$ on the testing data.

V. Summary and Future Directions

Our approach appears significantly better than the control model, however, it falls short of many other competing submissions, which achieve a mean accuracy of over 80%.

We have identified several improvements we'd like to make to our model, as well as concepts we did not have a chance to implement:

- We should multi-thread our model. We treat each of the four subsets independently, and this is a perfect candidate for parallelization.
- Initially, we tried to consider many different feature expansions: powers, roots, logs, and inverses. However, we later realized that not all data are positive, and so we restricted ourselves to powers only. We would like to do a more in-depth analysis of each feature to determine which expansions can be performed per-feature.
- We could consider retaining more than 50% of the most significant features. Currently, the final model retains about 30 features. We may also consider retaining all features as a "comprehensive control", but running the model on all features would be very time-consuming.

References

[1] Claire Adam-Bourdarios et al. “The Higgs boson machine learning challenge”. In: *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*. Ed. by Glen Cowan et al. Vol. 42. Proceedings of Machine Learning Research. Montreal, Canada: PMLR, Dec. 2015, pp. 19–55. URL: http://proceedings.mlr.press/v42/cowa14.html.