Report: Finding the Higgs Boson with Deep Learning

ASTR598: Topics in Theoretical Astrophysics

Winter 2019

Stephan Hagel
Student Number: 1870562
sthagel@uw.edu

Contents

1 Introduction **2**
1.1 The project 2
1.2 The data 2
1.2.1 Missing data 2

2 The Implementation **4**
2.1 Software prerequisites 4
2.2 Setting the options 4
2.3 Reading the data 5
2.4 Preprocessing the data 5
2.5 Training the network 7

3 Results **7**

4 Closing remarks and outlook **8**

1 Introduction

1.1 The project

The motivation behind this project was to find a way, to apply deep learning methods to high energy particle physics. The first thing I thought of was: ”Can I train a computer to recognize a Higgs boson from plain experimental data from the LHC?” Luckily, there has already been a challenge by ATLAS in 2014 to improve their data research with machine learning methods, called the Higgs boson machine learning challenge [1].

What my project is about after all is to take simulated ATLAS data from the CERN Open Data Portal to train a neural network. The network is used for binary classification. It classifies, whether a collision event contains the production of a Higgs boson (signal, s) or not (background, b).

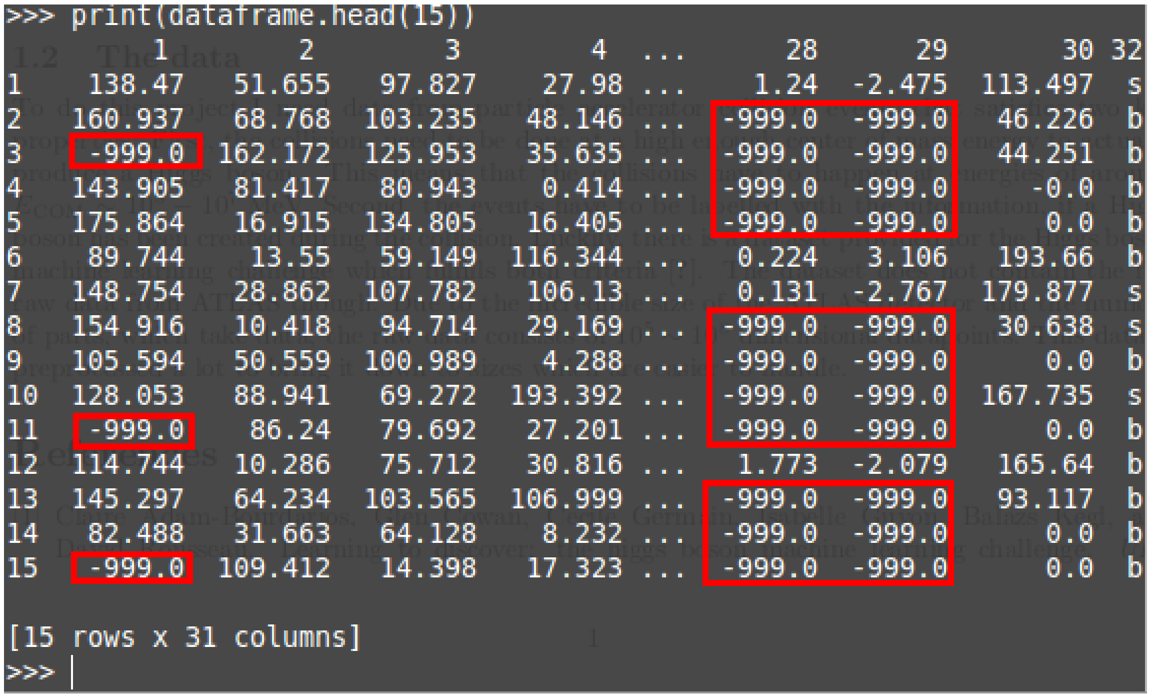
1.2 The data

To do this project I need data from particle accelerator collision events that satisfies two key properties: First, the collisions need to be done at a high enough center of mass energy to actually produce a Higgs boson. This means that the collisions have to happen at energies of around $E_{\mathrm{COM}} \sim 10^5 - 10^7$ MeV. Second, the events have to be labelled with the information, if a Higgs boson has been created during the collision. Luckily, there is a dataset provided for the Higgs boson machine learning challenge which fulfills both criteria [3]. The dataset does not contain the full raw data from ATLAS though. Due to the incredible size of the ATLAS detector and the number of parts, which take data, the raw data consists of $10^5 \sim 10^8$ dimensional datapoints. This data is preprocessed a lot to bring it down to sizes which are easier to handle. The actual dataset used for this project contains consists of 818238 events with 30 dimensions.[1]

1.2.1 Missing data

Figure 1 shows an excerpt from the dataset. Note, that there are some points of data, which have a value of -999.0 in some columns, eventhough other datapoints are far off from that value. Taking a closer look at the documentation of the data [1] we see that some columns contain data, which is related to particle jets produced during the collision. Events, in which no jets, or only one jet respectively, were produced,

[1]The full dataset contains even more dimensions, which are related to the challenge, but are not used in this project.



```
>>> print(dataframe.head(15))
         1        2        3        4  ...      28      29       30 32
1   138.47   51.655   97.827    27.98  ...    1.24  -2.475  113.497  s
2  160.937   68.768  103.235   48.146  ...  -999.0  -999.0   46.226  b
3   -999.0  162.172  125.953   35.635  ...  -999.0  -999.0   44.251  b
4  143.905   81.417   80.943    0.414  ...  -999.0  -999.0     -0.0  b
5  175.864   16.915  134.805   16.405  ...  -999.0  -999.0      0.0  b
6   89.744    13.55   59.149  116.344  ...   0.224   3.106   193.66  b
7  148.754   28.862  107.782   106.13  ...   0.131  -2.767  179.877  s
8  154.916   10.418   94.714   29.169  ...  -999.0  -999.0   30.638  s
9  105.594   50.559  100.989    4.288  ...  -999.0  -999.0      0.0  b
10 128.053   88.941   69.272  193.392  ...  -999.0  -999.0  167.735  s
11  -999.0    86.24   79.692   27.201  ...  -999.0  -999.0      0.0  b
12 114.744   10.286   75.712   30.816  ...   1.773  -2.079   165.64  b
13 145.297   64.234  103.565  106.999  ...  -999.0  -999.0   93.117  b
14  82.488   31.663   64.128    8.232  ...  -999.0  -999.0      0.0  b
15  -999.0  109.412   14.398   17.323  ...  -999.0  -999.0      0.0  b

[15 rows x 31 columns]
>>> |
```

Figure 1: The first 15 datapoints from the dataset used. The red boxes mark missing data.

have missing data in some columns. There were 568698 events, in which one or less jets were produced. This results in those events having missing data in columns 5, 6, 7, 13, 27, 28 and 29. During 320850 events there were no jets at all. Those events have additional missing data in columns 24, 25 and 26. To deal with this missing data, we will have to separate the events by the number of jets produced.

Unfortunately there is more missing data. Column 1 contains data for the mass of a possible candidate for a Higgs boson. The documentation for this columns says:

"The estimated mass m_H of the Higgs boson candidate, obtained through a probabilistic phase space integration (may be undefined if the topology of the event is too far from the expected topology)"

Therefore the missing data in this column is not directly related to the number of jets produced. It is also not related to the label of the event (signal or background). There are 122143 events with missing data in this column. Thus this column will need to be treated separately.

2 The Implementation

Next we want to take a look at how the problem was actually approached.

2.1 Software prerequisites

The code has been written in python 3 and has been tested in python 3.5 on the Hyak HPC cluster at the University of Washington. To read and manipulate the data the numpy and pandas packages are needed. The data has been further preprocessed with functions from the sklearn package. The neural network has been created and trained with keras using the tensorflow backend.

2.2 Setting the options

There are multiple ways to deal with the missing data in the dataset. The user can choose which way should be done by setting the corresponding options when running the code via the terminal. The usage looks like:

```
python3 src/main.py <options>
```

The options can be one or more of the following:

- `IGNORE_MASS_DATA`: If this option is set, the column containing the Higgs candidate mass will be ignored completely when training the neural network. This option prevents any events from being deleted but decreases the input dimension.

- `REMOVE_HIGGS_NAN`: If this option is set, those events with missing data for the Higgs candidate mass will be deleted. This option decreases the number of total events but keeps the input dimension as it is.

- `SIMPLE_IMPUTE`: This option will impute the missing data in the Higgs candidate mass with the physical value of the Higgs mass of 125.18 GeV [4]. This keeps the number of events and the input dimension unchanged, but approximates data.

- `IGNORE_MULTIJET_DATA`: If this option is set, all columns containing data, which is only defined if the number of jets is greater than one, will be ignored. Events with no jets produced will be deleted. This option reduces the number of events and the input dimension.

- IGNORE_JET_DATA: If this option is set, all columns containing jet data will be ignored. This keeps the number of events constant but reduces the input dimension even further than the previous option.

2.3 Reading the data

After the options have been set, the dataset has to be read by the program. To do this, the read_csv function from pandas is used.[2]

```
datafile = "data/data.csv"
dataframe = pd.read_csv(datafile, header=None)
```

Next the irrelevant data from the original challenge as well as the header of the data need to be removed.

```
del dataframe[KAGGLE_WEIGHT_INDEX]
del dataframe[KAGGLE_SET_INDEX]
del dataframe[WEIGHT_INDEX]
del dataframe[EVENTID_INDEX]
dataframe = dataframe.iloc[1:]
```

To further handle the data, we need to convert it from string to float and replace the -999.0 values by NaN, so it does not get confused with actually relevant data points. The labels also have to be converted to numerical values in order to be given to the neural network. I chose 1 to be the value for the signal label and 0 the value for the background label.

```
dataframe = dataframe.apply(pd.to_numeric, errors='ignore')
dataframe = dataframe.replace({-999.0:  np.NaN})
dataframe[LABEL_INDEX] = 1 - pd.factorize(dataframe[LABEL_INDEX])[0]
```

2.4 Preprocessing the data

Now the data is ready for actual preprocessing. First we will take care of the missing data. Depending on what options are set, the missing data for the jet productions are handled accordingly.

[2]In this report I will be skipping parts of the code which are not very instructive, such as error handling and garbage collection.

```
if IGNORE_JET_DATA:
   for i in SOLOJET_INDEX:
      del dataframe[i]
   for j in MULTIJET_INDEX:
      del dataframe[j]
elif IGNORE_MULTIJET_DATA:
   for i in MULTIJET_INDEX:
      del dataframe[i]
   dataframe = dataframe[dataframe[JETNUMBER_INDEX] > 0]
else:
   dataframe = dataframe[dataframe[JETNUMBER_INDEX] > 1]
```

Afterwards we can take care of the missing data in the Higgs candidate mass.

```
if IGNORE_MASS_DATA:
   del dataframe[DER_MASS_INDEX]
elif REMOVE_HIGGS_NAN:
   dataframe.dropna(inplace=True)
elif SIMPLE_IMPUTE:
   dataframe.fillna(PHYSICAL_HIGGS_MASS, inplace=True)
```

Next we can convert the pandas dataframe into a numpy matrix and use the MinMaxScaler from the sklearn package to normalize the data.

```
data_matrix = dataframe.as_matrix().astype(np.float)
scaler = MinMaxScaler()
scaler.fit(data_matrix)
print("Normalizing data.")
data_matrix_norm = scaler.transform(data_matrix)
```

For the split between test and training data, good results could be achieved with a 85/15 split.

```
target = data_matrix_norm[:, -1]
train = data_matrix_norm[:, :-1]
x_train, x_test, y_train, y_test = train_test_split(train, target,
   test_size=0.15, random_state=1)
```

With that out of the way, the data is ready to be used to train the neural network.

2.5 Training the network

The next question to ask ourselves is how to actually design the neural network. How many input neurons should we use? How many hidden layers should we use? How many neurons per layer should we use?
After some experimenting with those parameters and doing some reading I found the best results with the following blueprint: The number of input neurons should be equal to the dimension of the data. Their output type should be a rectified linear unit (ReLU). After that there should be one hidden layer, with $(\text{dim} + 1)/2$ neurons in it. Their output type should be a ReLU as well. Finally there is one output neuron of the sigmoid type, since it is a binary classification problem. The model is trained using the binary crossentropy as loss function and the `rmsprop` optimizer. The model is trained on the `x_train` and `y_train` data for 20 epochs and then evaluated on the `x_test` and `y_test` data.
The implementation of this procedure with keras is rather simple and given below.

```
model = Sequential()
mean = (input_dim_ + 1) // 2
model.add(Dense(input_dim_, input_dim=input_dim_, activation='relu'))
model.add(Dense(mean, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='rmsprop',
   metrics=['accuracy'])
model.fit(x_train, y_train, epochs=20, batch_size=128)
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

3 Results

The resulting score on the test data depends on the design of the neural network, as well as on the way, the missing data is handled. The first few times running the code I used a way bigger network (multiple hidden layers with 64+ neurons per layer). Interestingly this gave worse results than the final design. The results produced by the bigger network while ignoring all jet and Higgs candidate mass data were in the 70-75% range. Improving the network's design and using different ways to deal with missing data brought the accuracy of the model up significantly. The table below shows the score on the test data as well as the value of the loss function for all different ways of dealing with missing data.

Jet data option	Mass data option	Loss function	Accuracy
IGNORE_JET_DATA	IGNORE_MASS_DATA	0.3851	0.8246
IGNORE_JET_DATA	REMOVE_HIGGS_NAN	0.4122	0.8107
IGNORE JET_DATA	SIMPLE_IMPUTE	0.3765	0.8303
IGNORE_MULTIJET_DATA	IGNORE_MASS_DATA	0.4023	0.8149
IGNORE_MULTIJET_DATA	REMOVE_HIGGS_NAN	0.4082	0.8171
IGNORE_MULTIJET_DATA	SIMPLE_IMPUTE	0.3982	0.8187
None	IGNORE_MASS_DATA	0.3758	0.8357
None	REMOVE_HIGGS_NAN	0.3648	0.8393
None	SIMPLE_IMPUTE	0.3500	0.8501

Table 1: Results of the final model.

These results show us, that the model can give more accurate predictions, if we use higher dimensional data, eventhough we lose a lot of events that way. Using only the data of those events, in which two or more jets have been produced, together with the physical Higgs mass imputed for the missing data in that column brings the accuracy up to about 85%, which is a significant impovement from the beginning 70-75%.

4 Closing remarks and outlook

We have seen that even a simple model, which has been trained without any knowledge about high energy particle physics or relativistic kinematics, can perform quite well in classifying collision events. We have also seen that only a few steps of optimization can bring up the performance of the neural network significantly. Eventhough we get some nice results, the model is far from perfect and does not compare to the ones actually used for the challenge or research done at CERN. But even within the scope of this project there are still some ways to improve on the project. One idea, which I did not manage to implement yet is to improve the imputer for the missing Higgs candidate mass data by training a separate regression model on the events with valid data to predict a value for the missing data. Another way to improve is to implement some of the statistical tweaks mentioned by Balázs Kégl in his talk, in which he introduces the challenge.[3]

[3]http://opendata.cern.ch/record/330

References

[1] Claire Adam-Bourdarios, Glen Cowan, Cecile Germain, Isabelle Guyon, Balazs Kegl, and David Rousseau. Learning to discover: the higgs boson machine learning challenge. *CERN Open Data Portal*, 2014.

[2] S. Binet, Balazs Kegl, and David Rousseau. Software for the atlas higgs machine learning challenge 2014. *CERN Open Data Portal*, 2014.

[3] ATLAS Collaboration. Dataset from the atlas higgs machine learning challenge 2014. *CERN Open Data Portal*, 2014.

[4] Masaharu Tanabashi, K Hagiwara, K Hikasa, K Nakamura, Y Sumino, F Takahashi, J Tanaka, K Agashe, G Aielli, C Amsler, et al. Review of particle physics. *Physical Review D*, 98(3):030001, 2018.