Amerio Aurelio, PML week 3 assignment

Summary

In this analysis, I have used several MLP models, applied to the Kaggle Higgs dataset, in order to distinguish signal from noise.

Data pre-processing

After importing the training and test set, I scaled the training features to have 0 mean and unit variance. Than I applied the same transformation to the test set (note: the test set will not have 0 mean and unit variance).

There are two datasets: the small dataset (10k entries) and the full dataset (250k), both of them have more than 30 features. At the beginning, only some of the features have been used.

I am going to further split the training set into a training and validation set (10% of the data), so that I can optimize my model having a reference dataset, without the risk of leaking information from the test set.

Model1 and model2, small dataset

The first model I've built is a one-layer model. I used the small dataset and only a subset of the features ("DER_mass_MMC", "DER_mass_transverse_met_lep", "DER_mass_vis", "DER_pt_tot", "DER_sum_pt","PRI_tau_pt", "PRI_lep_pt", "PRI_met").

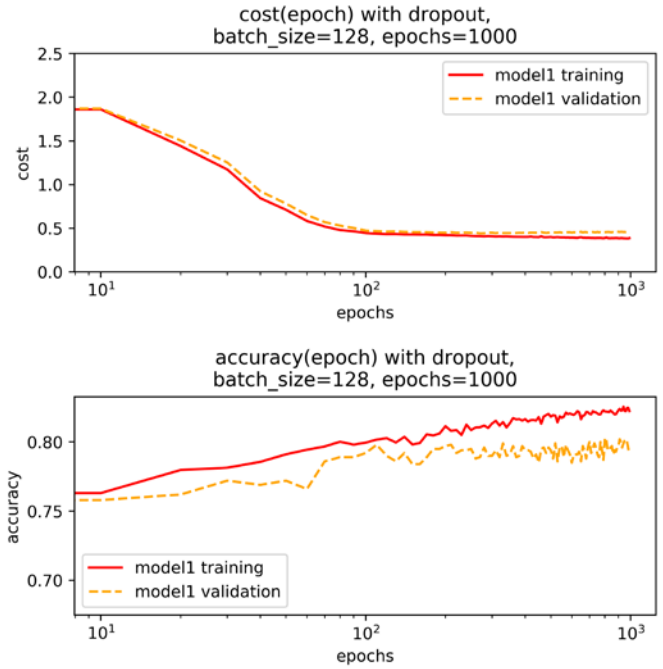
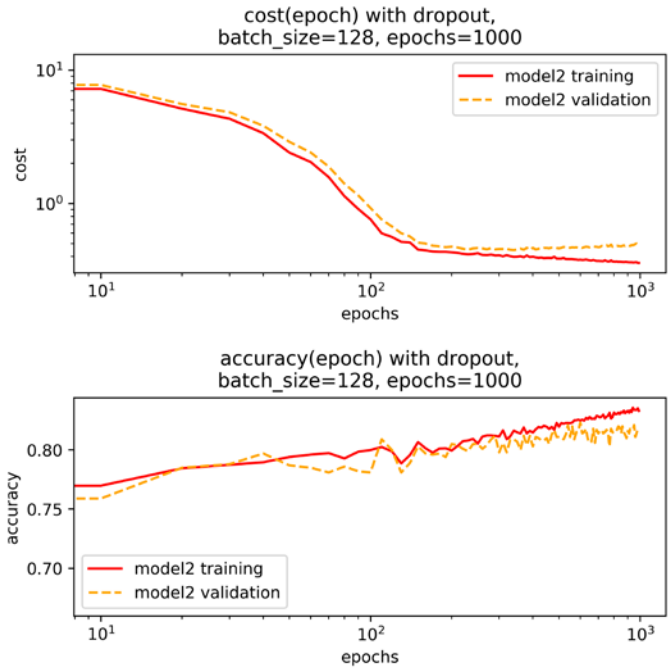


Fig. 1 model1 with 1 layer, learning rate=0.001, 256 nodes, batch size=128, dropout probability=0.2

The model shown in Fig. 1 is already overfitting and its performance is not that good (79.9% accuracy on the test set).

In order to avoid overfitting, and looking for a better accuracy, I have created another model (Fig. 2) with one layer and 1024 nodes, though this time with a bigger dropout probability (50%).

In this case I get a better performance on the validation set



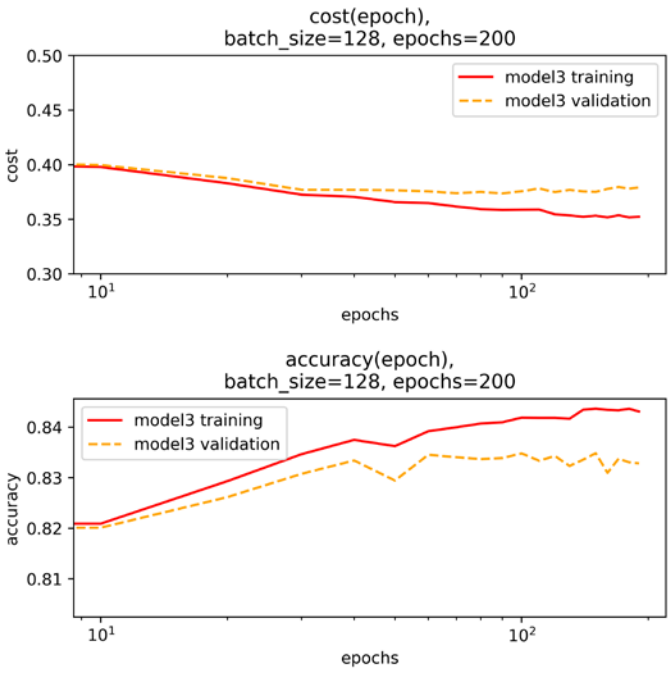
(81.5% accuracy), but the accuracy on the test set is lower (79%). Decreasing the number of training epochs may lead to better results on the test set and may let the model overfit less.

I have decided now to move to the full dataset with all the features (except for the EventId and Weights), as a model built on more data will eventually lead to a better performance.

Fig. 2 model2 with 1 layer, learning rate=0.001, 1024 nodes, batch size=128, dropout probability=0.5

Model3, full dataset

Model3 is the same model as model2, but it was trained on all the samples available (see Fig. 3). It achieved a much better performance on the training and test, in fact I achieved an accuracy of 83.4% on the test set. Now that we are using the full dataset, we can compute the AMS score, which is a metric useful to compare the performance of different models. For model3 I get AMS=0.9823.



The model is probably overfitting, as the accuracy on the validation set is decreasing. In the next model we will introduce a new kind of regularization, the weight L2 regularization.

Fig. 3 model3 with 1 layer, learning rate=0.001, 1024 nodes, batch size=128, dropout probability=0.5

Model4

A model with more layers should allow us to make finer adjustments on the parameters of the model and possibly lead to a better accuracy on the test set. Model4 has 4 layers, it has 600 nodes in each layer and there is a term added to the loss function in order to regularize the squared sum of the weights and biases. Thus, the new loss function becomes:

$$loss_{new} = loss_{old} + \beta_{\mathrm{w}} \sum w_i + \beta_{bias} \sum b_j$$

$\beta_{\mathrm{w}} = 0.0001$ and $\beta_{\mathrm{bias}} = 0.0001$

This model achieves an accuracy of 83.4% on the test set and has an AMS=0.9615<0.9823, thus it performs worse than model3 with just one layer.

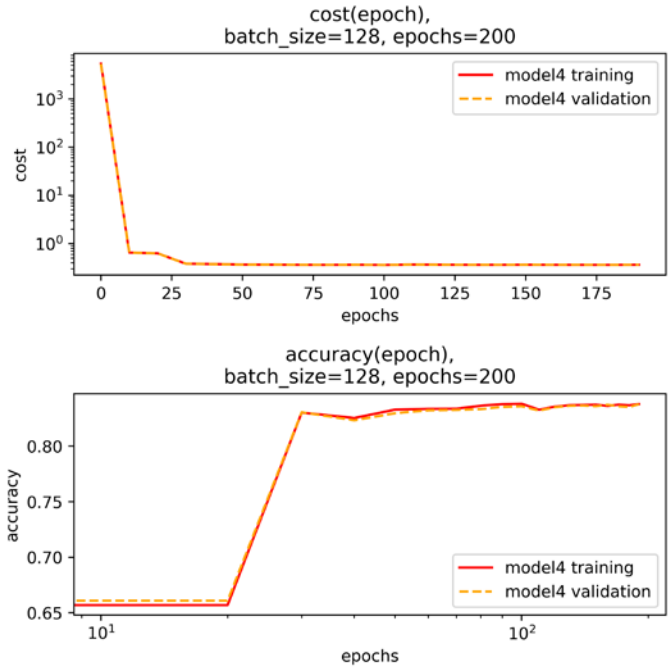


Fig. 4 model4 with 4 layers, learning rate=0.001, 600 nodes per layer, batch size=128, dropout probability=0.2, 0.5, 0.5, 0.5

Model5

I have tried several kinds of approaches to increase the accuracy on the test set, but without much success. They either resulted in overfitting or to an accuracy comparable to that of the previous models. A model worth noting is the one made using a technique called “exponential decay of the learning rate”, in order to allow finer adjustment of the weights as the training epoch increases.

This model is more prone to overfitting though, as increasing the accuracy on the training set doesn’t necessary lead to a better performance on the test set.

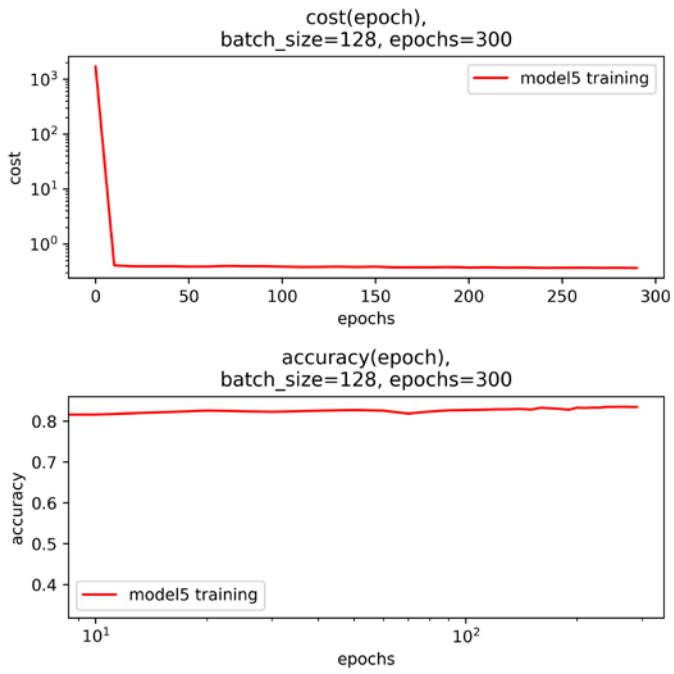


Fig. 5 model5 with 1 layer, learning rate=0.01 with exponential decay, 1024 nodes per layer, batch size=128, dropout probability=0.2, beta_w=beta_b=0.00005

In Fig. 5 the confrontation with the validation set is absent as I've trained the model on the full training dataset. The exponential decay of the learning rate is implemented according to this formula

$$\eta = \eta_0 d \cdot e^{\frac{globalStep}{decaySteps}}$$

Where η_0 is the initial learning rate (0.01 in this case), d is the decay rate (0.96) and decay steps is a constant (10000) which regulates how fast the decay happens. I have chosen d in a way that after 30% of the training epochs the learning rate would become 1/10 of the initial one.

Adopting this technique, I achieved an accuracy on the test set of 83.3% and an AMS= 0.9494, which is worse than the one achieved before.

Model6

The last model I have built (Fig. 6) is a model with 4 layers. It uses the leaky relu activation function (to avoid dead nodes during the last part of the optimization) and the learning rate decay technique. This time a further pre-processing step was adopted: I have run a principal component analysis on the training data (as implemented in the scikit-learn package) and kept only the first 22 projected features, out of the previous 30. This way, since some of the features were derived from the primary features, it is possible to consider the cross-correlation between the features and drop the projected features which contain lesser information (I've decided to drop the projected features which were contributing for less than 0.1%). As there were less features involved in this model, I have decided to reduce the number of nodes down to (600, 400, 400, 400), in order to avoid overfitting.

The new model which I have built has now an accuracy on the test set = 83.8% and AMS= 0.9993. This is the best model so far and it is a "big" improvement over model4 (the first model with 4 layers) which had AMS=0.9615, meaning that the number of signals being classified correctly has increased.

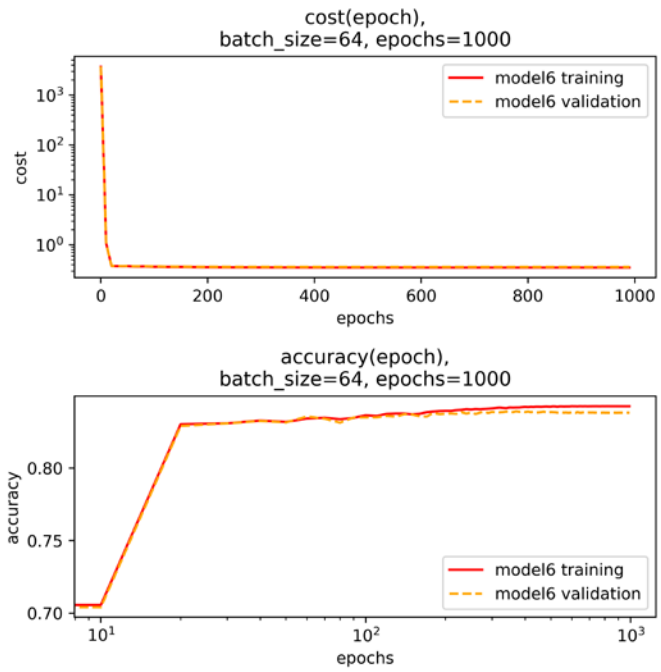


Fig. 6 model6 with 3 layers, initial learning rate=0.001 with exponential decay, decay_rate=0.96, decay_steps=10000, n_nodes=(600,400,400,400), batch size=64, dropout probability=0.2, 0.5, 0.5, 0.5, beta_w=0.0005, beta_b=0.00005. The L2 regularization for the biases is weaker, as it showed from my tests that it leads to smaller training time.

Possibly this model could perform a little better if the number of nodes per layer and regularization parameters were tweaked even more. In addition, given more training epochs, model6 would probably reach a better performance on the test set as the accuracy on training and validation set is slowly increasing over time.

Conclusions

To further improve the performance of our models, it would be probably necessary to resort to different activation functions (according to literature) which are more efficient than the relu (or leaky relu), for example the maxout activation function (which is the generalization of the relu function, though its implementation is not currently available in tensorflow v1.2) or the more advanced max-channel activation function, which is currently not implemented in tensorflow.

Some of the models I have proposed (especially model6), given more training epochs would possibly achieve a better performance, but I didn't have enough time and computational power to explore this possibility.