Detection of Genomic Island: Deep Learning Approach

Arnab Mukherjee

M.STAT 2nd Year, ISI Kolkata

Theory

Horizontal gene transfer is an important mechanism for the evolution of microbial genomes. In 1990, it was first observed that large blocks of horizontally acquired foreign sequences occur in chromosomes of pathogenic bacteria, and those regions are highly correlated with pathogenicity. Here I will discuss how to use deep learning methods to identify GI's. {Foreign gene blocks are GI's}

Methodology

We have a train set of size ~213k and test set of size ~53k, each of the sets have sequences of length 100 and each sequences are classified as 0 or 1. Previously we were splicing data by creating 401 sequences each of length 100 from a sequence of length 500. That approach resulted in overfitting, because we were populating train and validation set with sequences which are very similar, and the test set had sequences which are very different. This was due to the fact that if we have a 100 length sequences then expected number of sequences in the validation set which has 95 or more match (i.e. identical run of length 95+) with that sequence is ~ 1. So, I changed the data creation method. This time we took disjoin subsequences from the parent sequence. i.e. If a sequence is of length 500, we took 5 sequences of length 100 from it. I have tested 23 different models on this dataset. **Below I will list the models which performed above 90% level.**

Performance

Classical Method:

Among the classical methods only SVM with gaussian kernel attained more than 85% test accuracy (~88.9% to be accurate)

Deep Learning Methods:

At first, I trained different models for 5 epochs and compared their cross validated accuracy level. {I used k=5 fold cross validation here}

Description	CV Accuracy	CV Loss
One Layer LSTM with dropout = 0	91.718%	0.2573
One Layer LSTM with dropout = 0.2	91.742%	0.2562
One Layer LSTM with dropout = 0.4	91.662%	0.2603
One Layer LSTM with dropout = 0.6	91.718%	0.2574
One Layer LSTM with dropout = 0.8	91.621%	0.2687
Embedding Layer and one Dense Layer of size 32	91.618%	0.2879
Embedding Layer and one dense Layer of size 16	91.756%	0.2854
Two dense layer one with size 32 and another with size 16	91.741%	0.2821
Two Layer LSTM first layer of size 128, second layer of size 128 both with dropout = 0.2	91.785%	0.2549
One Embedding Layer and 4 Dense Layer each of size 16	91.716%	0.2857
4 Layer LSTM + 2 deep Layers	91.702%	0.2848
CNN+ one Dense Layer	91.904%	0.2424

After the first round of training it was evident that the performances of all the models are very similar (which is not surprising as they are very close to each other). Two models which stood out among the rest were CNN + one Dense Layer and 2 Layer LSTM, so next I joined these two models and created a new hybrid model with the following structure:

```
model = Sequential()
model.add(Embedding(input_dim=INPUT_DIM, output_dim=output_dim,
input_length=input_length, name='embedding_layer'))
model.add(Bidirectional(LSTM(rnn_hidden_dim, return_sequences=True)))
model.add(Dropout(dropout))
model.add(Bidirectional(LSTM(rnn_hidden_dim, return_sequences=True)))
model.add(Dropout(dropout))
model.add(Conv1D(128, 5, activation='relu'))
model.add(GlobalMaxPooling1D())

model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile('adam', 'binary_crossentropy', metrics=['accuracy'])
```

After 5 Epochs it has CV accuracy = 92.01% and CV Loss = 0.206

Performance on Test Data

I ran the hybrid model for 100 epochs and after 20 epochs the model starts over-fitting.

So, I have used the weights from the 20-epoch level and tested the model on the test set.

Test Set Accuracy: 92.41675%

Test Set Loss: 0.2517