SIGNAL PEPTIDE CLASSIFIERS: A 2nd REPORT

Authors: Joshua Koopmans, Michelle de Groot

Manually identifying signal peptides in peptide sequences is laborious and prone to human errors. Automating this process can provide ease of mind for researchers and can make room for other more important tasks. As a continuation to our previous work (SIGNAL PEPTIDE CLASSIFIERS: A REPORT), where three different classifiers were tested using three different encoders, we are now testing a convolutional neural network (CNN) on this same classification problem.

Convolutional Neural Networks

The aim of using a CNN is to extract information about the residues and their respective locations and to apply this gained knowledge to a new set of sequences. For the neurons to learn something about the residues in a peptide sequence, we first look at the individual residues and then a number of residues as a group.

In this report a simple CNN model containing four convolutional layers, with ReLu, BatchNorm and MaxPool steps in between, and a sigmoid function was used. The ReLu function looks at two features and keeps the one with the highest value. The BatchNorm function improves training speeds by first subtracting the batch mean and then dividing the batch by its standard deviation. Finally, the MaxPool function reduces the dimension of your input which results in more information being stored in a specific number of features.

The model can be found by following this link: model.py.

Radepathy Encoder

A new encoder used in this report is the radepathy encoder. The radepathy encoder, created by Daniel Rademaker and Gert Vriend, contains generated features by an autoencoder trained on 7 million columns of HSSP multiple sequence alignments. This encoder has 21 rows, one for each amino acid, with two value columns for each.

Results

Table 1: Area under the curve scores for every dataset with a selection of encoders.

DATASET V.S. ENCODING	**ONE-HOT**	**NLF**	**BLOSUM62**	**RADEPATHY**
VALIDATION SET	**0.999**	0.989	**0.999**	0.989
TEST SET (UNSEEN BY CLASSIFIER)	0.976	**0.988**	0.987	0.985

Using a simple model four encoders were tested on classification accuracy (**Table 1**). While training on train data the one-hot and BLOSUM62 encoders outperformed the NLF and radepathy encoders (0.999 v.s. 0.989) (**Figure 1**).

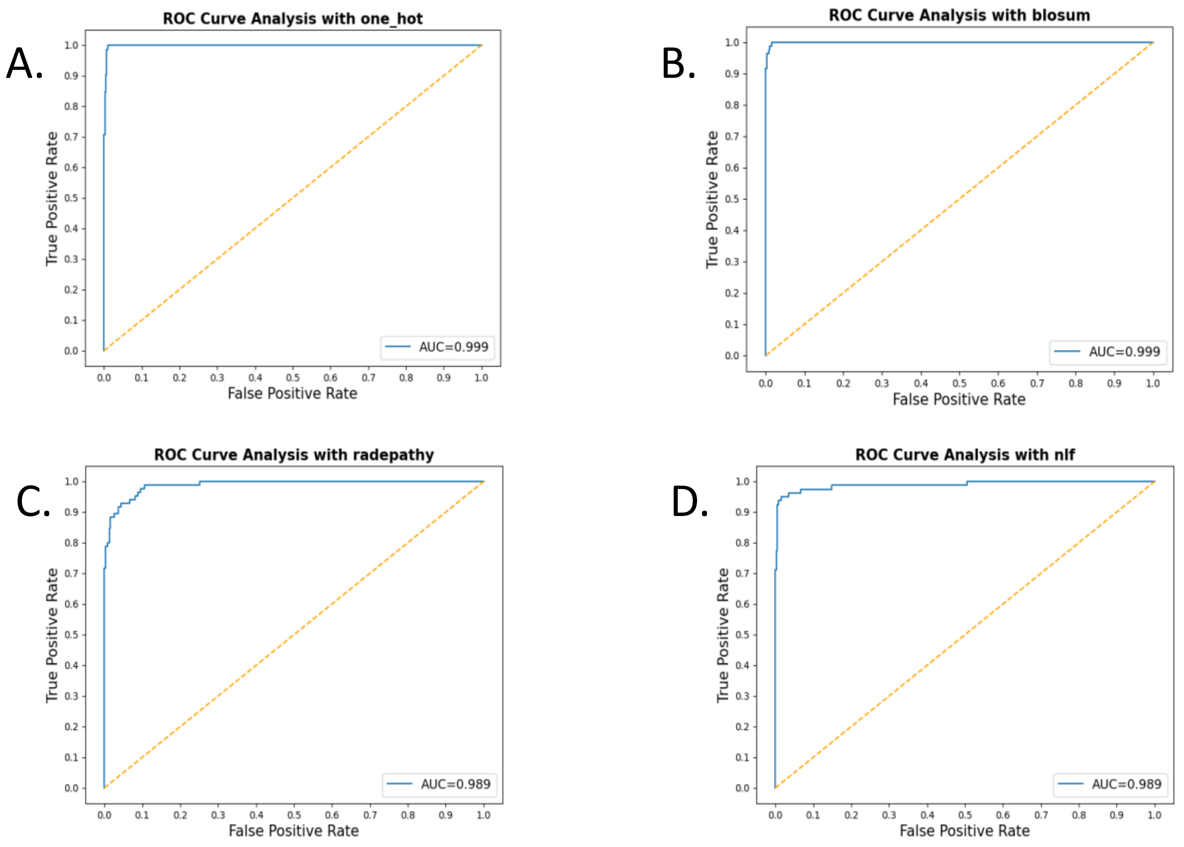

Figure 1: *ROC curves of the respective encoders using the training set.*

While testing on the never-seen-before-by-the-classifier benchmark data, the NLF encoder clearly outperformed all the other encoders with an area under the curve score of 0.988 (**Figure 2**).

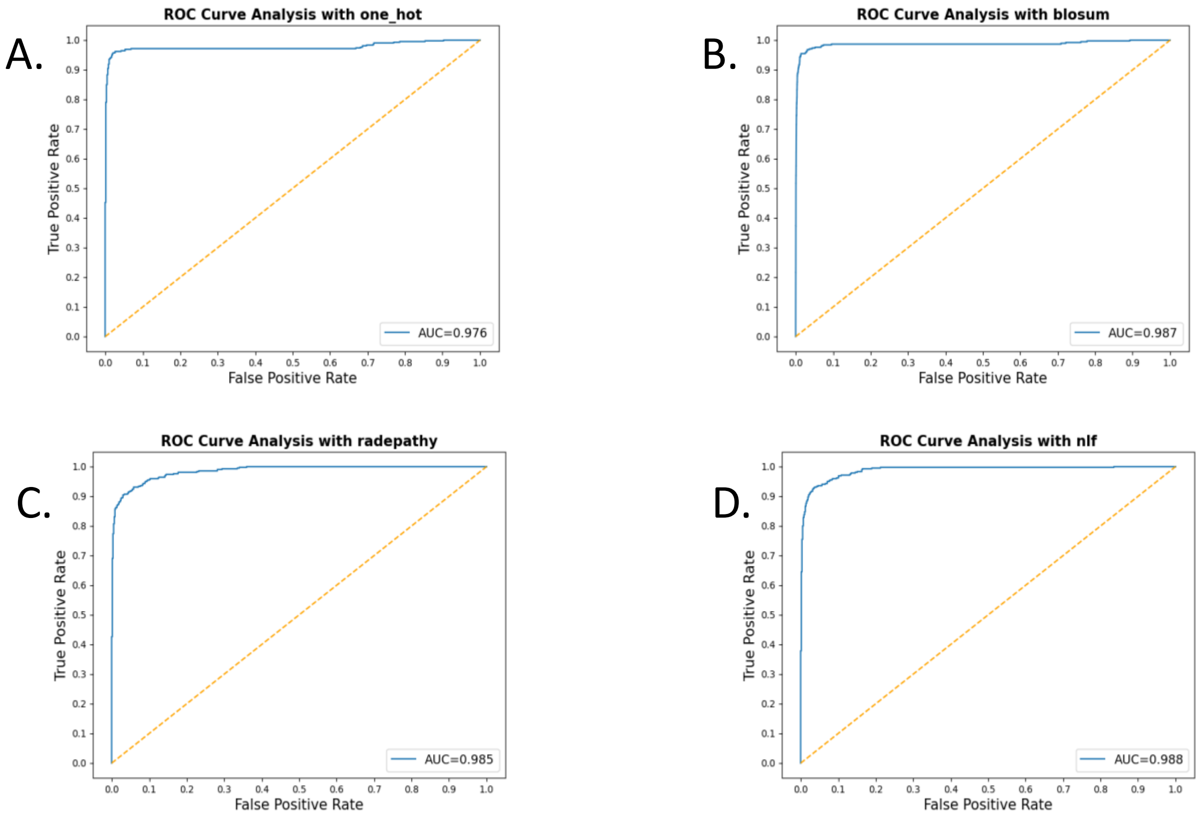

Figure 2: *ROC curves of the respective encoders using the benchmark (never seen before) set.*

Conclusion

Utilizing a CNN instead of other methods (e.g. SVM, Logistic Regression, Random Forest) will result in a higher classification accuracy score. In addition, the use of a simple but yet information-rich encoder can increase the total speed of the CNN training process while keeping the accuracy high. This is the case for the Radepathy encoder. It has been noticed that using a more “traditional” model (see the commented code in model.py) yields different results; one-hot encoding performs best with radepathy coming in second place (not shown). In this report the NLF encoder performed best, most likely due to the amino acid property considerations. All in all, eventhough a CNN is a better option to use compared to other methods, this model should be optimized for your own research needs.