Sequence Classification For Protein Analysis

Sepp Hochreiter and Klaus Obermayer
Technische Universität Berlin

To analyze the sequential data obtained from genome sequencing (DNA and the translated protein sequences), from EEG and ECG measurements, from environmental sensors (e.g. to detect earth quakes), or from sensors used in machine fault detection, sequences must be classified. These examples show that in biology, medicine, and control a great demand for sequence classification methods exists in order to evaluate recorded sequences and to understand the data generation process. But until now machine learning techniques failed to offer sufficient solutions for sequence classification.

The classifying sequences by treating them as vectors is not feasible because the sequences have different length and the model complexity increases with sequence length which in turn decreases the generalization capability. Recent approaches to sequence classification first extract features from the sequence and then apply classification algorithms on the feature vectors, e.g. kernel methods using kernels which were designed for sequences. However, the quality of these approaches rise and fall with the quality of the extracted features, i.e. constructed kernels. If not enough prior knowledge about relevant sequence features is available these feature extraction approaches are not reliable.

In principle the automatic extraction of appropriate sequence features can be realized by time series methods, i.e. by dynamical systems. However, sequence classification differs from traditional time series prediction because only at sequence end the system must supply an output, the class label. Here new problems arise. At each sequence position the classifier may receive important class information needed at the sequence end, therefore it must deal with long-term dependencies which leads to the problem of the “vanishing gradient” [3, 1]. The “vanishing gradient” addresses the characteristics of non-chaotic dynamical systems that the gradient of states with respect to previous states vanishes exponentially with the temporal distance between these states. This feature of non-chaotic systems results from the fact that initial conditions do not have large influence on later states. Therefore non-chaotic system are prevented from learning to store information over time. However learning to store relevant information till sequence end is essential for sequence classification.

To avoid the “vanishing gradient” problem we have introduced the “Long Short-Term Memory” (LSTM, [3]) and now report its application to protein motif and fold recognition. A volume conserving mapping of LSTM’s central subarchitecture keeps information and avoids the “vanishing gradient”. Volume conservation is realized by the identity of one unit after applying the sigmoid unit ("memory cell input") to the incoming signals. However during sequence processing relevant information is superimposed by irrelevant information. To avoid the disturbance from irrelevant information LSTM contains an unit, called "input gate", which controls the information storage. The identity unit together with the "input gate" and an analogous unit for the output ("output gate") serves as memory subarchitecture ("memory cell"). An LSTM network is build of these subarchitectures, where all units are fully interconnected within one memory cell and between memory cells.

We applied LSTM networks to protein sequences in order to predict its functional class or its 3D structure. The sequence of locally coded amino acids is processed by shifting a window over the sequence. The gating mechanism identifies and extracts sequence regions which contribute to classification, therefore we can extract motifs typical for a class. These tasks are important for extracting binding or active sites of the protein in order to develop new drugs. Further 3D protein folding features can be identified which would allow understand diseases resulting from protein misfolding.

In the first experiment the task is to predict the functional class of proteins. We used the PROSITE motif data base where class labels are extracted by literature search and functional motifs are constructed by biochemistry experts. LSTM was trained on 15 PROSITE classes and on the "SwissProt" data set (140,000 proteins). In seven out of the 15 protein classes LSTM re-discovered the known PROSITE motifs – without expert knowledge. For the remaining eight classes, LSTM found novel motifs which yielded on average superior classification results compared to the PROSITE motifs.

In a second set of experiments the task was to predict the 3D structure of proteins by predicting their fold classes. We used the SCOP fold domain data base. This task is more difficult than the previous because single motifs are not enough to predict the folding class. We trained LSTM models for 187 SCOP fold classes after randomly withdrawing 100 test sequences which sequence identities $\leq$ 30 %. The model selection set was divided into a training set (90 %) and a validation set (10 %), which was used to determine the optimal stopping time for learning, number of memory cells, and window size. A test sequence is classified to the class for which the according network had the largest output. As accuracy measure we used the Q percentage [2] and obtained **51 %** accuracy with LSTM. In [2] two SVM methods applied to feature vectors derived from the sequences obtained 33.5 % and 43.5 % while a neural network yielded 20.5 % accuracy. Note, that the task in [2] was easier because the sequence identity was $\leq$ 35 % on only 27 SCOP folds, i.e. LSTM had 7 times more possibilities to do wrong.

References

[1] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[2] C. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.