Predicting cellular location using UniRep embedding of protein sequence

Buisson-Chavot Guillaume, Cnudde Anthony, Ody Jessica, Van den Schilden Jan

29-06-2020

1. Introduction

Protein synthesis (Miller 2001) and subsequent subcellular localization involves complex mechanisms frequently studied over the years . Many of these synthesized proteins are excreted into the extracellular environment (Benham 2012). Such extracellular proteins allow cells to interact with the outside world or other cells. Without, multicellular organisms would be unable to form intercellular interaction, tissues or body parts (Yoshitaka Shirasaki 2014). But also in bacterial cells there can be up to a third non-cytoplasmic proteins (Orfanoudaki 2017). It is of great interest to know how to distinguish secreted proteins from cytoplasmic ones, especially for those expressing recombinant proteins. Recombinant proteins are a tool frequently used in the medical sector or for industrial protein production (Peng Chong 2019).

Signal peptides are short peptide sequences between 15 and 30 residues at the N-terminal end of the protein that are recognized by cell pathways to excrete the protein (Benham 2012, Klatt 2012). During transit, the signal peptide is cleaved off the rest of the protein (Peng Chong 2019). Even though signal peptides display low evolutionary sequential similarity, the underlying biophysical properties display an high degree of conservation (Orfanoudaki 2017). It consists of three regions: a positively charged region, a hydrophobic region, and a region with the cleavage site (Klatt 2012, Peng Chong 2019). Software like SignalP-5.0 use deep neural networks to recognize these signal peptides and predict cellular localization (Armenteros 2019).

Even though signal peptides are often necessary for protein excretion, they are in themselves insufficient. Other features, such as secondary structure propensity, dynamics, and amino acid composition, play an important role in determining protein location. Predictors that use these features have been reported with a success rate of 95.5% (Loos 2019).

One limitation of including features is that the accuracy of the methods depends on how much information each feature can provide. There might also be very important hidden features we don't know about, but necessary for optimal performance. On the other hand, some features could be very descriptive but experimentally expensive to obtain. Protein sequencing on the other hand has become very cheap in the last years, and protein sequence information is abundant. As on the moment of writing, the public database UniProt (Morgat 2019) contained 181,252,700 protein sequences. However, only 562,253 (0.32 %) contained manual annotations of features with experimental evidence. Often ML-based predictors, are trained on well defined features and as a result discard more than 99% of the available sequences. However just the knowledge that the sequence exists and is from and extant protein is already useful information.

A research group (Alley 2019) took inspiration from a state-of-art natural language processing method, which trains a mLSTM RNN on a next character prediction problem. Subsequently the hidden states are used to form a rich representation of the text. Similarly, Alley et al. trained a mLSTM RNN on next residue prediction of the protein sequences. To achieve this goal, the neural network will encode its own features in the hidden nodes to predict the next residue. A trained network can subsequently be used to generate a fixed length vector in which important protein features are encoded. They have shown that such vector can significantly improve the performance of ML based predictors as opposed to using raw sequence information.

In this work, The UniProt REST API was used to generate a dataset of cytoplasmic and periplasmic proteins. After reducing the sampling bias with CD-HIT, UniRep vectors were generated with the mLSTM RNN of Alley 2019 et al. Using these fixed length vectors, we trained our own implementation of linear regression, decision tree, support vector machine (SVM), and recurrent neural network. The performance of the different methods was compared.

2. Methods

2.1. Data mining

The protein dataset for this work was generated using the Uniprot REST API. The API provides a programmatic access to download the sequences through queries. The

goal in this work was to make a classifier that can distinguish between cytoplasmic and periplasmic proteins. Only Gram-negative Bacteria have a periplasm, but this is not one phylogenetic group. In this work we limited ourselves to Gammaproteobacteria.

To generate a set of cytoplasmic proteins, the query asked for proteins with an annotation of being located in the cytoplasm or cytosol. As an extra safeguard, it was specified that those proteins could not have an annotation of containing a signal peptide as cytoplasmic proteins normally do not have signal peptides. The query is shown below.

```
QUERY="taxonomy:Gammaproteobacteria
(locations:(location:cytoplasm)
OR locations:(location:cytosol))
NOT annotation:(type:signal)"
```

To generate the set of periplasmic proteins, a similar search was performed. This time looking for an annotation of the protein being in the periplasm and the presence of a signal peptide (shown below).

```
QUERY="taxonomy:Gammaproteobacteria
locations:(location:periplasm)
annotation:(type:signal)"
```

As we explain before, signal peptides are used to identify destination of proteins, but it has been noted that certain biophysical features are necessary in addition to guarantee their translocation. Therefore, a third dataset was generated which was identical to the periplasm dataset, except that the signal peptides were cut off to compare the performance of the different ML methods when trained with and without signal peptide.

To limit sampling bias, the software CD-HIT a calculates the percentage identity between the protein sequences. Sequences with more identity than 50 percent are clustered together and a representative was chosen.

Finally, the sequences were transformed into a fixed length vector representation using UniRep. This method extracts states from an unsupervised trained mLSTM-RNN and combines them into a fixed length UniRep representation. This representation contains essential structural and functional features that can be used by ML algorithms to distinguish between Periplasmic and Cytoplasmic proteins.

For the decision tree and the neural network, and linear regression, 3000 sequences of cytoplasmic protein and 3000 sequences of periplasmic protein are selected from the datasets previously shuffled to randomize the samples This gives a working dataset of 6000 sequences.

2.2 Linear regression

Principle and algorithm

In a multiple linear regression model the value to predict (dependent variable, column vector Y) depends on one or multiple independent variables (matrix X) in a linear way. Each independent variable has its own weight (column vector B).

$$Y = XB$$

Optimizing the model is choosing the weights in the Column vector B so that observations Y' deviate as little as possible from the predicted values Y. To give a quantification to "as little as possible", the mean square error cost function was used.

$$MSE = \frac{1}{N}\sum(Y - Y')^2$$

While this gives an indication of how good a certain set of weights is, it doesn't tell in which directions these weights should be modified to get better weights. To get this direction, the partial derivative is of the cost function is taken in each of the direction. To not "jump" to far, this new direction is multiplied by a fraction. This method is also called gradient descent.

$$B_{n+1} = B_n - \gamma * \Delta MSE(B_n)$$

2.3 Decision tree

Principle

A decision tree is a binary tree in which each node corresponds to a subset of the data in the parent node. The splits used to create the child nodes are found by minimizing a cost function which represents how 'pure' are the nodes, i.e. how mixed are the classes of the data in the node. This cost function is computed on each split of each feature in the training set. The data subset in the node is then distributed between the two children nodes according to the fact that the value at the chosen feature is lower or higher than the value chosen for the split. The objective is to obtain nodes that are as pure as possible, hopefully with only one class in it. Once the tree is trained, the prediction is simply done by travelling from node to node according to the split defined in the current one.

Algorithm

The algorithm is simple. For each feature in the dataset, we begin by sorting the values vector of that feature in ascending order. Then, for each possible value in this vector, we split the data in left subset and right subset and compute the score on these subsets. The value giving the subsets with the highest score among all features is selected as split point. This step is repeated for each node, giving birth to the left and right children nodes, and the process is repeated recursively until all the nodes are pure or a certain depth is reached. The cost function used here is the GINI purity score. This score tells how pure a node is, i.e. how similar the classes in the node are. A node with only one class in it would then have a GINI score of 1. The GINI score is computed as follows:

$$G = \sum_{i=1}^{c} (p_i)^2$$

With c the number of classes in the dataset, and pi the probability for a datapoint of being of class i. This score holds for each subset given by the split. To compute the total score for the split, the GINI score is computed for each subset, and then weighted by the proportion of the data of the parent node that is present in each subset and summed. So, if m is the size of the starting dataset, i the size of the left subset and j the size of the right subset obtained by splitting the starting dataset:

$$G_{split} = \frac{i}{m} G_{left} + \frac{j}{m} G_{right}$$

2.4 Support Vector Machine (SVM)

Principle

SVM is a supervised machine learning algorithm used in general for binary classification problems. Given a set of training examples, each marked as belonging to one or the other of two categories, a SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped such that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

Algorithm

The SVM algorithm is trained by feeding a dataset with labeled examples (xi, yi). For instance, if our set is composed of proteins and our problem is to determine from which cellular compartment they are, then:

- An example protein x is defined as an n-dimentional feature vector that can be plotted on n-dimensional space.
- The feature vector, as its name indicates, contains features of your protein in numerical form
- Each feature vector is labeled with a class yi depending on the compartment.
- The class y can either be a +ve or -ve (for instance, periplasm protein = 1, cytoplasm protein = -1)

Using this dataset, the algorithm have to find a hyperplane (or decision boundary) whose properties are as follows:

- It creates a separation between proteins of the two classes with a maximum margin
- the hyperplane equation (w.x + b = 0) yields a value 1 for proteins from +ve class and -1 for proteins from -ve class The training algorithm has to find the optimal values w and b which define this hyperplane. These optimal values are found by minimizing a cost function. Once the training algorithm has identified these optimal values, the SVM model f(x) is then defined as : f(x) = sign (w. $x + b$)

Our objective is to find a hyperplane that separates +ve and -ve proteins with the largest margin while keeping the misclassification as low as possible. To achieve this goal, we will minimize the following cost function:

$$J(w) = \frac{1}{2}||w||^2 + C\left[\frac{1}{N}\sum_{i}^{n} max(0, 1 - y_i * (w \cdot x_i + b))\right]$$

In the training phase, larger C results in the narrow margin and smaller C results in the wider margin

The cost function is essentially a measure of how bad our model is doing at achieving the objective. Then we will move on to its gradient which will be used in the training phase to minimize it :

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i}^{n}\left[\frac{1}{2}\|\mathbf{w}\|^2 + C\max(0,\ 1 - y_i * (\mathbf{w} \cdot x_i))\right]$$

$$\nabla_w J(\mathbf{w}) = \frac{1}{N}\sum_{i}^{n} \begin{cases} \mathbf{w} & \text{if } \max(0,\ 1 - y_i * (\mathbf{w} \cdot x_i)) = 0 \\ \mathbf{w} - C y_i x_i & \text{otherwise} \end{cases}$$

If you look closely at J(w), to find it's minimum, we have to:

- minimize $\|w\|^2$ which maximizes margin $(2/\|w\|)$
- minimize the following sum of Hinge loss function which minimizes misclassifications

$$max(0, 1 - y_i * (w.x_i + b))$$

In order to reach this goal, we use Stochastic Gradient Descent (SGD).As stop criterion, we build a function with a loop which iterate for instance 1000 times and we will stop the loop and thus the training when the current cost hasn't decreased much as compared to the previous cost.

Note : SVM algorithms can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

2.5 Neural network

Principle

A neural network is a set of layers composed of nodes called neurons, connected to each other by weighted connections. These layers are of three type: one input layer, in which the input is given, one or more hidden layers, and an output layer, giving the result of the prediction. One more node can be added to each layer: the bias node. The idea behind neural network is to adapt the weights of the connections between the nodes, as well as the biases, to get an accurate prediction. To assess the accuracy of the network, we need to use k-fold cross validation. In k-fold cross validation, the dataset is divided in k folds, and one fold is used as testing set while the others are used to train the classifier. Once this step is done, the process is repeated using another fold as testing set and reimplementing the previous one in training process, and so on until all folds have been use.

Algorithm

To goal of the algorithm is to find a good set of weights and biases to get an accurate prediction. The principal tool used to achieve this goal is called backpropagation. Backpropagation is an algorithm using the error on the output layer to retro-compute the errors generated by the weights and biases to adapt them step by step. The first step in a neural network is the feedforward step, where we will compute the output given the current parameters. To do so, we first initialize the weights at random. Then, each node is computed as the combination of each node of the previous layer, weighted by the given weight connecting this node to the corresponding node of that previous layer. Then, the computed node is 'activated' by a so-called activation function that maintains it in a certain range (for example, between 0 and 1 with a sigmoid activation function). This process is repeated until the output node, where the obtained values gives the output. The problem then becomes an optimisation problem: we define a cost function, reflecting the error in the output layer, and we will try to minimize it. This is done by finding an area where the derivative of this function is close to 0, meaning the cost function does not evolve anymore (at least in this area as this could be a local minimum). This is where backpropagation comes into play. By chain rule, we can compute the error on the layer thanks to the error on the layer , i.e., we can get the error on each layer given the error on the output. With this process, we will adapt the weights by passing the inputs of the training set through the network one by one. The activation function used here is the classical sigmoid activation. It is given by the following formula:

$$\sigma = \frac{1}{1 + e^x}$$

This function is classically used in neural networks. It keeps value in a range between 0 and 1. This also allows to add non-linearity in the network, allowing to perform non-linear mappings. The cost function used is the cross-entropy function. This cost-function makes training easier compared to a simple MSE score because it avoids slow-down in learning by not having a partial derivative dependant of the activated derivative that can be close to 0 due to the nature of sigmoid activation. The cross-entropy function is computed as follows:

$$C = -\frac{1}{n} \sum_x [y ln_a + (1 - y) ln(1 - a)]$$

3. Results and Discussion

We can see on the PCA plot (Fig. 1) that the distribution of the sequences is different with or without the signal peptide. With the signal peptide the delimitation is clearer between the cytoplasmic and periplasmic protein. The third principal component allows a good separation between the two set of sequences. Without the signal peptide, we still could see the two clouds of points but there is a bigger overlap between both.

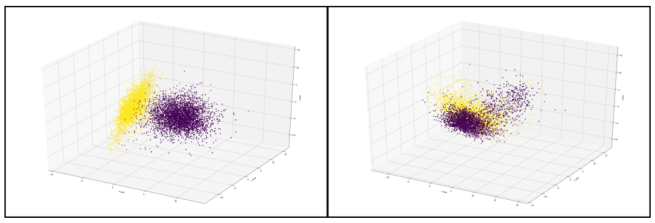

Fig. 1: PCA plot of the cytoplasmic protein (yellow dots) and periplasmic protein (purples dots) with the peptide signal (left) and without the peptide signal (right).

The accuracy of the linear regression on the dataset with peptide signal is 98.12% and the accuracy for the dataset without the signal peptide is lesser, 94.56%.
After training and testing the decision tree, we can see that the accuracy, the proportion of correctly predicted sequence, is over 95%. For the Neutral network, the modification of the parameters η and λ and the batch size does not seem to highly impact the k-fold cross validation as it is always around 99%. After training with the differents parameters, the results are slightly better with the batch size to 8 and the λ value around 0.2. When comparing the two algorithms, we could see that the accuracy of the decision tree is 97.66%, which is still below the neural network's one.

4. Conclusion

The different approaches have good accuracy to predict the excretion of the protein into the periplasm or if the protein will stay inside the cell. The best algorithms between those which were tested is the neural network one with almost every time an accuracy of 99%. We could also conclude that the signal peptide is of great significance to predict the excretion of the protein but there is clearly an other type of signal to forestall the excretion of proteins.

References

Alley, Ethan C., et al. "Unified rational protein engineering with sequence-only deep representation learning." bioRxiv (2019): 589333.

Almagro Armenteros, J.J., Tsirigos, K.D., Sonderby, C.K. et al. SignalP 5.0 improves signal peptide predictions using deep neural networks. Nat Biotechnol 37, 420–423 (2019). https://doi.org/10.1038/s41587-019-0036-z

Benham, Adam M. "Protein Secretion and the EndoplasmicReticulum." Cold Spring Harb Perspect Biol 4 (2012).
Ferro-Novick, Susan and Brose, Nils. "Traffic control system within cells." Nature 504 (2013): 98.

Klatt, Stephan, and Zoltan Konthur. "Secretory signal peptide modification for optimized antibody-fragment expression-secretion in Leishmania tarentolae." Microbial cell factories 11 (2012): 97.

Loos, Maria, et al. "Structural basis of the sub-cellular topology landscape of Escherichia coli." Frontiers in microbiology 10 (2019): 1670.

Miller, Jefferey H. "Protein Synthesis." In Encyclopedia of Genetics, by Sydney and Millern Jefferey H. Brenner, 1567. New York: Academic Press, 2001.

Morgat A, Lombardot T, Coudert E, Axelsen K, Neto TB, Gehant S, Bansal P, Bolleman J, Gasteiger E, de Castro E, Baratin D, Pozzato M, Xenarios I, Poux S, Redaschi N, Bridge A, UniProt Consortium. Enzyme annotation in UniProtKB using Rhea Bioinformatics (2019)

Orfanoudaki, G., Markaki, M., Chatzi, K. et al. MatureP: prediction of secreted proteins with exclusive information from their mature regions. Sci Rep 7, 3263 (2017). https://doi.org/10.1038/s41598-017-03557-4

Peng Chong, Shi Chaoshuo, Cao Xue, Li Yu, Liu Fufeng, Lu Fuping. "Factors Influencing Recombinant Protein Secretion Efficiency in Gram-Positive Bacteria: Signal Peptide and Beyond." Frontiers in Bioengineering and Biotechnology 7 (2019): 139.

Stephanie J. Popa, Julien Villeneuve, Sarah Stewart, Esther Perez Garcia, Anna Petrunkina Harrison, Kevin Moreau. " Genome-wide CRISPR screening identifies new regulators of glycoprotein secretion." bioRxiv , 2019: 522334.

Yoshinori Tsuchiya, Kazuki Morioka, Junsuke Shirai, Yuichi Yokomizo and Kazuo Yoshida. "Gene design of signal sequence for effective secretion of protein." Nucleic Acids Research Supplenzent 3 (2003): 261 -262.

Yoshitaka Shirasaki, Mai Yamagishi, Nobutake Suzuki, Kazushi Izawa, Asahi Nakahara, Jun Mizuno, Shuichi Shoji, Toshio Heike, Yoshie Harada, Ryuta Nishikomori, Osamu Ohara. "Real-time single-cell imaging of protein-secretion." Scientific Report 4 (2014): 4736.