Predicting the Cellular Localization Sites of Proteins Using Decision Tree and Neural Networks

Yetian Chen
Department of Computer Science
Iowa State University
yetianc@cs.iastate.edu

Abstract
In this paper, we describe the implementation of a set of machine learning techniques: Decision Tree, Perceptrons, Two-layer feed-forward Neural Networks. We describe the application of these techniques to the problem of classifying proteins into their various cellular localization sites based on their amino acid sequences. We evaluate the performance of each technique by analyzing the prediction accuracies on two datasets: classifying 336 E.coli proteins into 8 classes and classifying 1484 yeast proteins into 10 classes. The results show that the three techniques achieve similar prediction accuracy, 65%~70% on E.coli dataset, 46%~50% on yeast dataset, which are much better then those of simple majority algorithm (45% and 29% respectively).

1. Introduction
Knowing a protein's localization helps elucidate its function, its role in both healthy processes and in the onset of diseases, and its potential use as a drug target. Experimental characterization of protein localization is accurate but slow and labor-intensive. However, the amino acid sequence of a protein usually provides crucial indication to its cellular localization sites. On the other hand, sequenced genomic data is experiencing an exponential increase in recent years due to maturation of High-Throughput sequencing techniques. Thus, many computational methods have been proposed to try to set up the link between a protein sequence and its cellular location. These include McGeoch's method for signal sequence recognition, discriminant analysis of the amino acid content of outer membrane and periplasmic proteins, etc. However, each of these methods can only deal with one protein category, i.e. giving the probability of a sequence being a membrane protein, or deciding whether it is a nucleus protein or not. Thus, for a new protein sequence on which people have no pre-knowledge, the only way to decide its localization site is to check all available methods to get a sense. However, people still need to judge between these results to decide which method is more reliable, what is the cutoff probability for it to be safe to say a protein is in a certain cellular localization site but not in other sites. Thus, it is in a great need to develop a comprehensive system, integrating protein sequence-derived data and prediction results from all the methods described above. It has been showed that a variety of machine learning methods can be used for this purpose.

In this paper, I implemented three different techniques: Decision tree, Perceptrons, Two-layer feed-forward Neural Network, for predicting a proteins' subcellular localization site. The inputs are vectors, each of which corresponds to a protein. Each atom in a vector is the output (a score between 0 and 1.0) obtained by running a certain computational method on this protein sequence. The output is the predicted localization site of a protein. Two datasets are used to evaluate the performances of these techniques. One is a set of proteins from E.coli (336 instances), representing the kingdom of prokaryote; the other is a set of proteins from yeast (1484 instances), representing the kingdom of eukaryote. The actual localization sites of the proteins are already known. Each dataset are divided to training set and test set. For each strategy implemented, the accuracies on test sets are calculated. Then I compare the accuracies of the three strategies.

2. Materials and Implementations

2.1 Datasets
The E.coli dataset includes 336 examples; the yeast dataset includes 1484 examples.

Each of the attributes used to classify the localization site of a protein is a score (between 0 and 1) corresponding to a certain feature of the protein sequence. The higher the score is, the more possible the protein sequence has such feature.

In the E.coli dataset, seven features (attributes) are used: mcg, gvh, lip, chg, aac, alm1, alm2. And proteins are classified into 8 classes: cytoplasm (cp), inner membrane without signal sequence (im), periplasm (pp), inner membrane with uncleavable signal sequence (imU), outer membrane (om), outer membrane lipoprotein (omL), inner membrane lipoprotein (imL), inner membrane with cleavable signal sequence (imS). In the yeast dataset, eight
features (attributes) are used: mcg, gvh, alm, mit, erl, pox, vac, nuc. And proteins are classified into 10 classes: cytosolic or cytoskeletal (CYT), nuclear (NUC), mitochondrial (MIT), membrane protein without N-terminal signal (ME3), membrane protein with uncleaved signal (ME2), membrane protein with cleaved signal (ME1), extracellular (EXC), vacuolar (VAC), peroxisomal (POX), endoplasmic reticulum lumen (ERL).


2.2 Decision Tree

Preprocessing the datasets
Since most attributes are continuous numbers, for simplicity, we discretize the data points into finite number of bins. We first determine the minimum and maximum values for each attributes in the dataset. Then divide this range into five equal-width bins: tiny, small, medium, large, huge. Finally we discretize all of the data into these bins. This is done independently for each of the continuous features.

Since the training set and test set are not provided separately, we need to split the dataset into a training set and a test set. We randomly split dataset to two such that 70% are in the training set and 30% in test set. In the dataset, for some examples, some feature values are missing (a '?' at such position). If so, replace '?' with 'tiny'.

Implementation details
We implement the C5 algorithm to train the decision tree. When choose an attribute, we calculate the gain for each attribute with the remaining examples. And we always choose the attribute with the greatest gain.

2.3 Neural Networks

2.3.1 Perceptrons

Perceptrons are single layer feed-forward neural networks. They have all the input nodes connected directly to the output nodes. Perceptrons can only model functions that are linearly separable.

Fig 1 gives an example structure of a perceptrons network for our problem. For each attribute, we construct an input node. For each class, we construct an output node. In this example, four attributes are used to classify examples to three classes: cp, imS, im. Thus, there are totally four input nodes and three output nodes. For any pair of input node and output node, there is a directed edge between them. If the desired class is cp, then the desired output put is <1, 0, 0> on the three output nodes.

When training the network, we calculate the output at each output node, which is a double number. The prediction will be the node which has the maximal value of all output nodes. The activation function we use is the sigmoid function $g(x) = 1/(1 + e^{-x})$.

Since we do not know whether the datasets are linearly separable, we use a threshold to determine the termination condition. This threshold is defined as the ratio of correctly classified example in the training set. When the ratio is met, the training is terminated. We test threshold $t = 0.1, 0.2, \ldots, 1.0$ to find the maximum $t$ such that the program terminates within one hour.

Fig 1. The structure of the Perceptrons network: an example. Assume the dataset use four attributes, and there are three classes: cp, imS, and im. Assume the desired class is cp.

The algorithm is formulated as follows.

Algorithm 1 Perceptrons Learning
Function PERCEPTRONS-LEARNING (examples, network)
  initially set correct=0
  initialize the weight matrix $w[i][j]$ with randomized number within[-0.5,0.5]
  While(correct < threshold) //threshold =0.0, 0.1, 0.2,…, 1.0
    for each e in the example do
      for each output node calculate
        $O_j = g\left(\sum_{i=1}^{m} w[i][j]x_i[e]\right)$
      prediction = r such that $O_j = \max_j(O_j)$
      if $r \neq y(e)$
        for each output node $j$
          $Err_j = y_j(e) - g\left(\sum_{i=1}^{m} w[i][j]x_i[e]\right)$
          for $i=1, \ldots,m$
            $w[i][j] = w[i][j] + \alpha \times Err_j \times x_i[e] \times g\left(\sum_{i=1}^{m} w[i][j]x_i[e]\right) \times \left\{1 - g\left(\sum_{i=1}^{m} w[i][j]x_i[e]\right)\right\}$
        endfor
      endif
    endfor
  endwhile
  Return $w[i][j]$

2.3.2 Two-Layer Feed-Forward Neural Network

To enlarge the function space that the neural network can represent, we implement the two-layer feed-forward
network which involves one layer of hidden nodes. With a single, sufficiently large hidden layer, it is possible to represent any continuous function of the inputs with arbitrary accuracy.

Fig 2 illustrates the structure of a two-layer feed-forward neural network. The definitions of input nodes and output nodes are exactly like those in the perceptrons network. The only difference is that we have one layer of hidden nodes between the input and output nodes.

Similarly, we also use the ratio of correctly classified examples in the training set as the threshold for the termination condition. The major difference of the algorithm for training two-layer feed-forward neural network is when we update the weights for the hidden layer, we should back-propagate the error from the output layer to the hidden layer.

At the output layer, the weight-update rule is identical to that for perceptrons. For each output nodes, we defined a modified error \( \Delta_j = (y_j(e) - g(in_j)) \times g'(in_j) \), so that the weight-update rule becomes

\[
W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_j
\]

At the hidden layer, we need to back-propagate the error from the output layer to it. The propagation rule is

\[
\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i
\]

\[
W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j
\]

Since we use sigmoid function as the activation function, \( g'(x) \) in all equations has the form

\[
g'(x) = g(x) \times (1 - g(x))
\]

3. Results and Discussion

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DT</th>
<th>Perceptrons</th>
<th>2-layer NN (hidden node=5)</th>
<th>Majority</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.coli</td>
<td>68.04±5.03%</td>
<td>66.76±4.34% (Threshold=0.7)</td>
<td>65.68±6.09% (Threshold=0.7)</td>
<td>45.05%</td>
</tr>
<tr>
<td>Yeast</td>
<td>46.63±2.55%</td>
<td>50.41±2.74% (Threshold=0.5)</td>
<td>30.28±2.23% (Threshold=0.5)</td>
<td>28.82%</td>
</tr>
</tbody>
</table>

Note: The results for decision tree are average over 100 runs. The results for perceptrons and two-layer NN are average over 50 runs.

Table 1 summarizes the accuracies of four different algorithms on the two datasets. Majority denotes the simple majority algorithm, which classifies all examples in the test set to the major class that dominates the training set.

For the decision tree, the algorithm achieves about 68% correctness on the E.coli dataset while only has accuracy of 46% on the yeast dataset. Fig 3 shows a decision tree constructed for the E.coli dataset (only the first three levels are showed).

For perceptrons, for Ecoli dataset the algorithm always terminates within one hour if the threshold is set to 0.7. For threshold above 0.7, it doesn’t always terminate within one hour even it may terminate in one random run. This means the perceptrons cannot represent the function that fits the data.
ecoli dataset. For yeast dataset, it is poorer. The algorithm only converges to 50% of the examples in the dataset. This suggests that both the two datasets are linearly inseparable. And it only achieves an accuracy of about 66% on the ecoli dataset and 50% on the yeast dataset by using the network trained under the corresponding threshold.

For the 2-layer neural network, the results are very similar to those for the perceptrons, even with one hidden layer it can represent a larger function space. Here we used 5 hidden nodes. Even we use 10 hidden nodes, the algorithm give the similar results. This indicates the function needed to fit these two datasets should be discontinuous function.

![Fig 4](image)

**Fig 4.** Accuracy on test sets achieved by using the two types of neural networks trained under different thresholds.

Fig 4 showed the relation between the threshold used to train the neural networks and the achieved accuracies on the test sets. For the E.coli dataset, the perceptrons and the two-layer NN have the same accuracy on the test set as that on the training set. This indicates the training set and test set have the similar distributions. Interestingly, for the yeast dataset, the networks trained to meet the correctness ratio of 0.1 for training set can achieve 0.3~0.4 accuracy on the test set. However, when the threshold is increased to 0.5, the accuracies on the test set don’t increase as fast as the threshold does.

Our results show that decision tree, perceptrons and two-layer feed-forward network achieve the similar accuracies on E.coli dataset. On yeast dataset, the two neural network techniques have slightly better accuracy than decision tree. However, all the three machine learning techniques are significantly better than the simple majority function, which indicates that there indeed exists some patterns in these two datasets that can be learned.

In this paper, we implement three machine learning algorithms: decision tree, perceptrons, two-layer feed-forward network. We applied them to the problem of classifying proteins to their cellular localization sites based on the amino acid sequences of proteins. The experiments on Ecoli and yeast datasets shows the three algorithms achieve the similar accuracies on these two datasets, which are much better than simple majority function. Furthermore, it is showed that the two datasets are linearly inseparable. The 2-layer neural network with 5 hidden nodes is not able to completely separate the datasets. However, we cannot say these two datasets cannot be separated by continuous function. We need to explore using larger number of hidden nodes in the network. Certainly, we could implement three-layer feed-forward neural network which could represent discontinuous function.

Since the prediction of proteins’ cellular localization sites is a typical classification problem, many other techniques such as probability model, Bayesian network, K-nearest neighbors etc, can be applied to this problem. Thus, an aspect of future work is to examine the performance of these techniques on this particular problem.

Knowing a protein’s localization helps elucidate its function, its role in both healthy processes and in the onset of diseases, and its potential use as a drug target. Experimental characterization of protein localization sites is slow and labor-intensive. Computational analysis based protein sequence can guide the experimental design for characterizing the location of proteins, thus save significant labor and time. Thus, it is greatly rewarding to improve the techniques for computationally predicting the cellular localization sites of proteins.

### References

