Department of Computer and Information Science
The 2nd semester of academic year 2016/2017

CISB366 Bioinformatics

Project 2 – Prediction of MHC binding peptide by Artificial Neural Network

Instruction: Please complete the project in a pair of two. Report your algorithm design, running output, and analysis in a document. Pack your programs and the report in a zip file.

Deadline: 12 April 2017 23:55 (Wed)

Project Goal: MHC peptide prediction is an important step in the process of vaccine design. Here, you are asked to *implement* a three-layer Artificial Neural Network (ANN) with the use of bias terms for 9-mer MHC-peptide binding classification, you need to *analyze* your prediction models by reporting and discussing their performances.

Step I. Implement ANN using Python (80%)

Peptide sequences with known binding mode were obtained online from the MHCBN database:

Table 1: Data set

data file	classification	class label	total number of samples
data_b.dat	binder (b)	1	400
data_nb.dat	non-binder (nb)	0	400

- Encode the peptide sequences into a binary vector of 180 bits using the Sparse encoding scheme. Randomize the order of the data samples. For both binders and non-binders, use 200 samples as training data and the remaining 200 samples as validation data.
- Implement a **three-layer ANN model with bias terms**:
  - Accept the binary vector in the input layer
  - Use the sigmoidal function as the activation function in the hidden and output layer
  - Backpropagate using the square error and steepest descent algorithm. Note that you have to calculate the update equations to the weights and bias (because the lecture example doesn't include the bias term)
  - Output the classification "*predict"* based on this decision function with threshold=0.5:

$$predict = \begin{cases} 1 & if\ O \geq threshold\ \text{(binder)} \\ 0, & otherwise\ \text{(non-binder)} \end{cases}$$

- Use these initial settings, train the ANN with the training data and record the training error at the end of each cycle. The pseudocode of one training cycle and the evolution of training error over number of training cycles are shown in the figure below.

Table 2: Initial setting for the ANN for the MHC prediction problem

# hidden nodes	# training cycles	Learning rate ε	initial weights & bias
5	50	0.5	random

```
pseudocode one-training-cycle

TP=TN=FP=FN=0

foreach i in N-training-data-set
  convert i  to output O (ANN)
  calculate E
  update weights
  calculate predict
  update counts TP, TN, FP, FN
endfor

calculate training-error=(FP+FN)/N

end
```

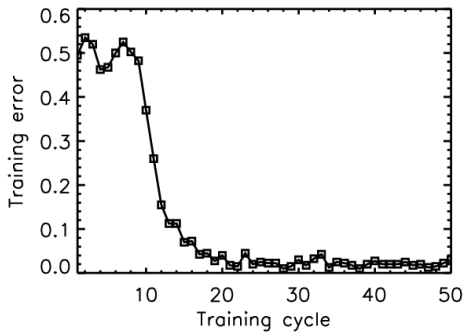


Figure 1 (a) Pseudocode of one-training cycle.
(b) Evolution of training error for ANN of 5 hidden nodes and ε =0.5.

- After all training cycles are completed, predict and calculate the accuracy = (TP+TN)/N of the validation data set with your trained ANN.
- You can program your code to output like this:

```
-- Training process --
Train data (b): 200
Train data (nb): 200
Hidden node: 5
Epsilon: 0.5
Training cycle: 50
Initial bias w: 0.696469
Initial bias v: 0.712955 0.286139 0.428471 0.226851 0.690885
Training error after the last cycle: 0.02000

-- Validation process --
Validation data (b): 200
Validation data (nb): 200
Validation accuracy: 60.75
```

Report:

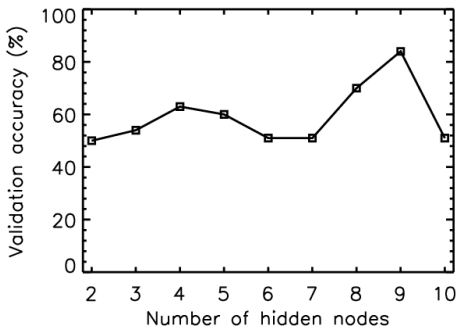
- Draw the flow chart of your program
- Show the running output and the training error figure for the initial settings (See table 2)
- Draw the ANN architecture of your final model

Step II. Find the best ANN model (20%)

Report: Discuss what you did and what you find. Support all your analysis with figures or tables.

A. Try changing the ANN architecture by varying the number of hidden nodes (from 2 to 10), compare the validation accuracies of all models and decide which ANN model performs the best. You can show your validation accuracies in a figure (sample below) or in a table.

Sample output:



B. For the ANN with the best number of hidden node you found above, try varying the learning rate ε =(0.1, 0.5, 1.0, 1.5, 2.0). Compare the training error figures, are all learning rates good choices? Why? Look at the validation accuracies, which might be the best learning rate?