

## Neural Networks in bioinformatics

Olof Emanuelsson

Stockholm Bioinformatics Center  
Stockholm University

web: <http://www.sbc.su.se/~olof/>

e-mail: [olof@sbcsu.se](mailto:olof@sbcsu.se)

tel: 08-5537 8574

## This lecture:

1. Introduction to Neural Networks (NN)
  - the idea of "artificial neurons"
  - important concepts of artificial neural networks
  - two different types: supervised vs. unsupervised
2. One type of NN covered in more detail: *feed-forward* NN
  - the over-all principle
  - the components (nodes, weights)
  - the training procedure
3. An example application: *TargetP*
  - the biological problem
  - creating training sets
  - the neural networks
  - sequence logos

## The idea behind artificial neural networks

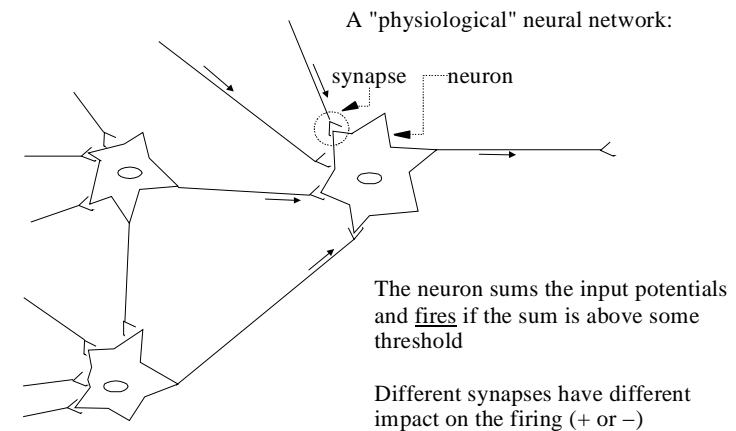
The brain of a vertebrate is (in general) capable of *learning* things

Example: having seen a number of trees, a normally gifted person will be able to recognise almost all types of trees

The idea: to construct networks of *artificial* neurons and make them *learn* and *generalize* in a way similar to how the physiological neural networks do that

As an example, the artificial neural networks may learn to recognize a particular type of sequence motif, e.g.:

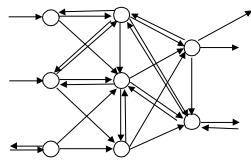
- the local sequence environment around a residue that is part of a secondary structure element
- a subcellular localization signal ("address label")



## This is formalised into an *artificial* neural network

The neurons are computing nodes: summing the input, outputting something based on that sum

The connections have different strengths (even negative)



An example (not fully connected)

## What are neural networks used for?

Pattern recognition and classification!

This means to learn to generalize from a set of training data and to apply the knowledge for predictive performance, *e.g.* :

- speech recognition
- radar target detection
- medical diagnostics
- signal detection (finding a signal in noise), *e.g.* recognition of biological sequence motifs in DNA & proteins

## Detection and prediction of sequence motifs

Several pattern recognition techniques are used in molecular biology/bioinformatics:

- patterns (regular expressions) [covered by AE]
- profiles [covered by AE]
- hidden Markov models (HMM) [covered by AE and GvH]
- support vector machines (SVM) [not covered in this course]
- artificial neural networks (NN) [today]

## Why not use sequence motifs and profiles?

Profiles/motifs (e.g. from PROSITE) do not capture long– (or medium–)range dependencies

Each position is treated as if it is independent of the other ones

but:

Neural networks are able to take into account such dependencies

Markov models of higher order are also capable of this but they very quickly become computationally intractable

(Note: if your problem is satisfactorily solved using profiles/motifs, then no need to bother with other methods of course!)

## Different types of NNs

### Supervised vs. unsupervised NNs:

Supervised – e.g. "feed-forward":

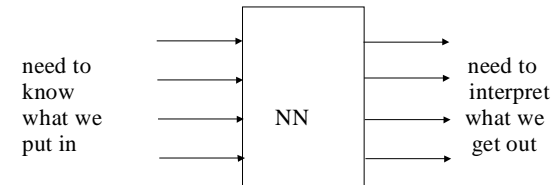
- used for classification of patterns, e.g. does this protein contain a particular sequence motif ?
- the "correct answer" is known and used in the training
- synapse weights are updated based on difference between *actual* and *desired* output

Unsupervised – e.g. "SOM":

- used for clustering of sequences
- the "correct answer" is not used in the training
- synapse weights are updated based on how often a particular synapse is used

## Important neural network concepts

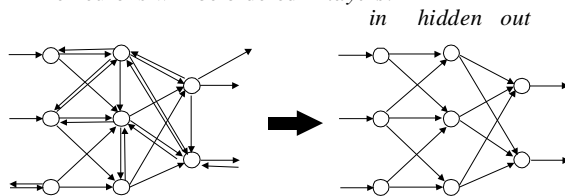
- network architecture: *feed-forward, kohonen, hopfield, ...*
- the "neurons" [ANN: nodes]: *threshold, logistic/sigmoidal, linear, ...*
- the "synapses" [ANN: weights]
- "training" of network: we need good data sets
- interpretation/post-processing:



## The feed-forward neural network – *architecture*

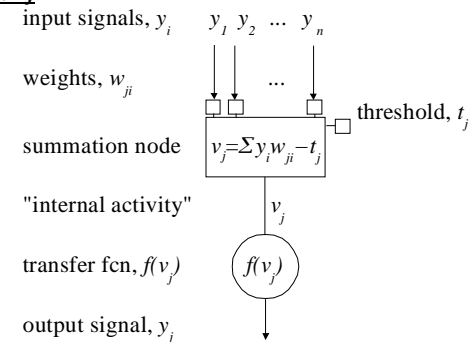
Allow connections only in one direction (feed-forward NN)

The neurons will be ordered in layers:



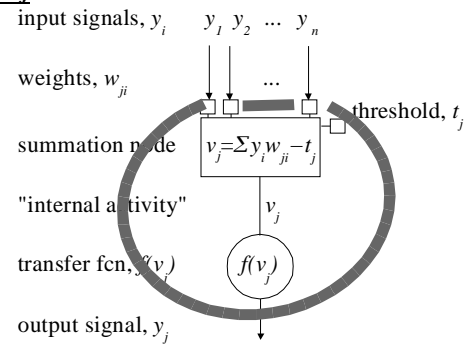
## The feed-forward neural network – *the neuron/the node*

### Neuron *j*



## The feed-forward neural network – *the neuron/the node*

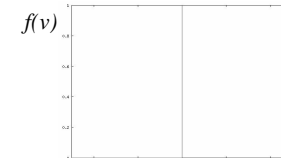
### Neuron $j$



## The feed-forward neural network – *the transfer function $f(v)$*

### threshold neuron

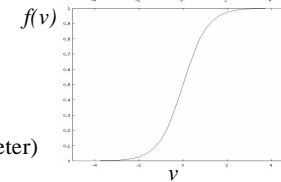
$$f(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ 0, & \text{if } v < 0 \end{cases}$$



### logistic neuron

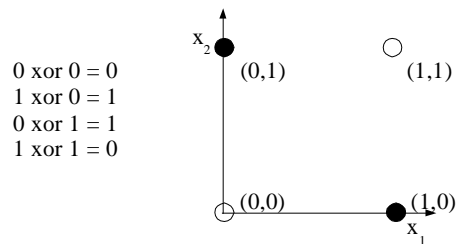
$$f(v) = \frac{1}{1 + \exp(-av)}$$

(where  $a$  is the slope parameter)

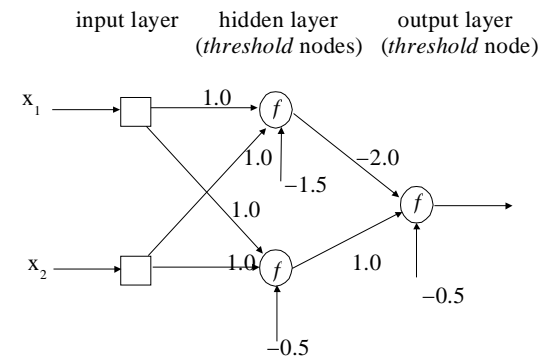


## The feed-forward neural network – *the XOR problem*

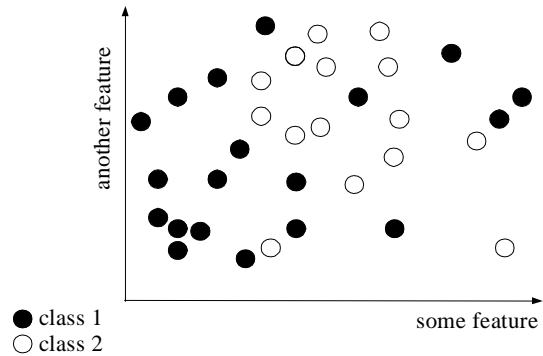
A non-linear *classification* problem



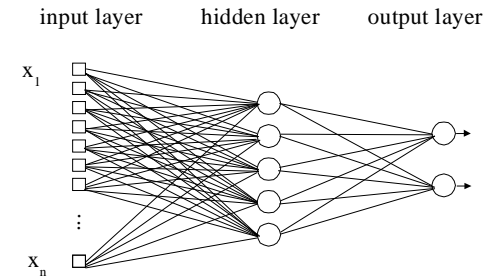
## The feed-forward neural network – *solving the XOR problem*



## The feed-forward neural network – a (slightly) more realistic data set



## The feed-forward neural network – a more realistic architecture



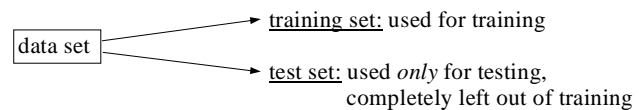
- logistic neurons is the most commonly used type
- number of hidden nodes typically in the range [2,100]

## The feed-forward neural network – important considerations

- (learning = training)  
=> will use the example of amino acid sequence from now on
- encoding of input (amino acid sequence into numbers)
  - learning algorithm and learning rate
  - the training set: must be *representative*
  - when to stop the training (the number of *training cycles/epochs*)?
  - post-processing (i.e. *interpretation* of output)
  - how to measure performance?

## The feed-forward neural network – important considerations

The training set should contain sequences from both (all) classes in approx. equal amounts



### cross-validation:

- divide the data set into  $n$  parts
- use  $n-1$  parts for training
- use 1 part for testing

## Input encoding – going from amino acid sequence to numbers

– sparse encoding [20 numbers per residue]

MASL... =>

M
A
S
L  
 00000000001000000000 10000000000000000000 00000000000000010000 00000000010...

– physicochemical encoding

- e.g. [3 numbers per residue]
1. residue side-chain volume
  2. hydrophobicity
  3. charge

## Input encoding – going from amino acid sequence to numbers

– sparse encoding [20 numbers per residue]

PROBLEM:

what if the motif we are looking for has different lengths in different proteins!?

SOLUTION:

==> use a "sliding window technique"

i.e. looking at a *piece* of the sequence at a time

## Input encoding – going from amino acid sequence to numbers

– sliding input window

sequence: MASLVLRSLAVAFLDAGRSVRAASAVEGPA...

if window size is 7:

1<sup>st</sup> window MASLVLV  
 2<sup>nd</sup> window ASLVLVR  
 3<sup>rd</sup> window SLVLVRS  
 ...  
 11<sup>th</sup> window ... AVAFLDA  
 ...

## Input encoding – going from amino acid sequence to numbers

– sliding input window

sequence: MASLVLRSLAVAFLDAGRSVRAASAVEGPA...

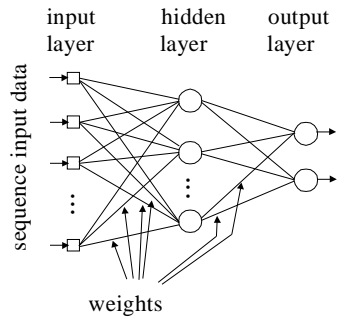
if window size is 7:

1<sup>st</sup> window MASLVLV  
 2<sup>nd</sup> window ASLVLVR  
 3<sup>rd</sup> window SLVLVRS  
 ...  
 11<sup>th</sup> window ... AVAFLDA  
 ...

1<sup>st</sup> window translated using "sparse encoding":

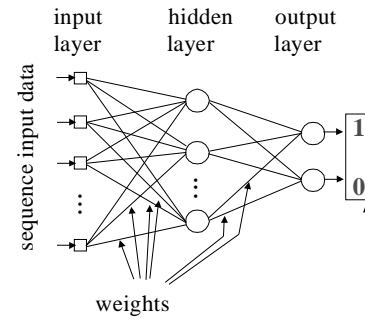
000000000100000000 10000000000000000000 000000000100000000 00000000000000000000 000000000000000100 00000000000000000000 00000000000000000100

### The feed-forward neural network – the training principle



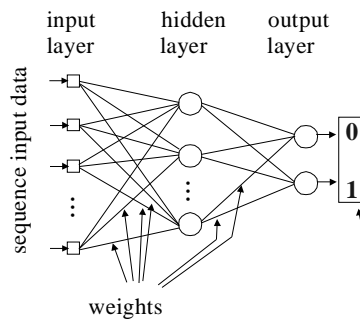
- (sequence) data presented at input
- correct answer fixed at output
- difference between correct and actual output used for weight adjusting (*training*)
- present both "positive" and "negative" training examples

### The feed-forward neural network – the training principle



- (sequence) data presented at input
- correct answer fixed at output
- difference between correct and actual output used for weight adjusting (*training*)
- if motif *is present* in the sequence at the input ("positive" training example)

### The feed-forward neural network – the training principle



- (sequence) data presented at input
- correct answer fixed at output
- difference between correct and actual output used for weight adjusting (*training*)
- if motif *not present* in the sequence at input ("negative" training example)

### Input encoding – going from amino acid sequence to numbers

– sliding input window

sequence:	MASLVLVRS	LAVAF	LDAGRS	VRAAS	AVEGPA . . .
motif:	yyyyyyyyyy	ynnnnnnnnn	nnnnnnnnnn	nnnnnnnnnn	. . .
<i>if window size is 7:</i>					
1 <sup>st</sup> window	MASLVLV			1	0 (y)
2 <sup>nd</sup> window	ASLVLVR			1	0 (y)
3 <sup>rd</sup> window	SLVLVRS			1	0 (y)
...	...				
11 <sup>th</sup> window		AVAFLDA		0	1 (n)
...		...			

1<sup>st</sup> window translated using "sparse encoding": target:

00000000000000000000 10000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 0

## The feed-forward neural network – *the training algorithm*

1. Start with random weights
2. Show one input example and calculate output
3. Calculate output errors (difference between observed and desired output)
4. Adjust weights to decrease the error (using error backprop. alg.)
5. Repeat (2)–(4) for all input examples
6. Repeat (2)–(5) until error minimum is found (each repeat is called a *training cycle/epoch*) (typically 50–1000 epochs)

## The feed-forward neural network – *the training algorithm*

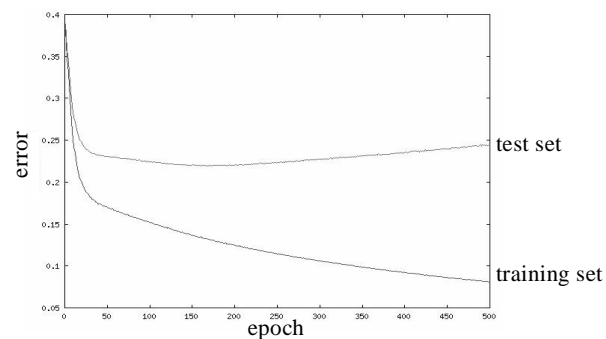
"Error backpropagation" algorithm:

1. Measure error:  $e_j = y_j - d_j$ , where  $j$  is an output node
2. Calculate total error: 
$$E = \frac{1}{2} \sum_{j \in \text{output}} e_j^2$$
3. Update the weights: 
$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = (\dots) = -\eta \delta_j y_i$$

$\eta$  = learning rate (typically 0.001–0.1)  
 $\delta_j$  = the "local gradient" associated with neuron  $j$
4. Repeat until some stopping criterion is met

## When to stop training

We want to get a good *generalization* performance *and* to avoid over-fitting of the parameters to the *training* set (over-training)



## The training set

- large enough
- contain all possible classes in approximately equal amounts
- unbiased, i.e. no particular type *within* a class should be overrepresented — this is important for two reasons:
  - if training set is biased towards a particular type, so will the ANN be
  - if training and test set contain too similar examples, the performance will be over-estimated

in short: the training set should be *representative*

## Cross-validation

- split the total data set into (*e.g.*) 5 parts:
  - 4 parts for training
  - 1 part for testing
- performance on *training* part measures the *reproduction* capability of the NN
- performance on *testing* part measures the *generalization* capability of the NN

## How to measure performance for classifications

Matthew's correlation coefficient for a 2-category predictor:

$$MCC = \frac{tp * tn - fp * fn}{\sqrt{(tn + fn)(tn + fp)(tp + fn)(tp + fp)}}$$

Sensitivity =  $tp / (tp + fn)$   
 the fraction of the sequences that *belong* to a certain class that really are *predicted* to that class

Specificity =  $tn / (tn + fp)$   
 the fraction of the sequences that are *predicted* to a certain class that really *belong* to that class

## An example:

On a test set with 20 motif-containing (m+) and 47 motif-lacking (m-) proteins, the following results were obtained:

predicted:	m+	m-		
true :	m+	17	3	tp
	m-	8	39	fn
				tn

$$\text{sensitivity} = tp / (tp + fn) = 17 / (17 + 3) = 0.85$$

$$\text{specificity} = tn / (tn + fp) = 39 / (39 + 8) = 0.68$$

$$MCC = \dots = 0.64$$

$$\text{correct overall (protein level)} = 56 / 67 = 84\%$$

$$\text{correct overall (residue level)} = 5946 / 7341 = 81\%$$

## Post-processing of NN output

In the example here, output from the network is *one score per residue*

Often (but not always) we want *one score per protein*; i.e. does this particular protein contain the motif we are looking for or not

Need to log the output scores for all residues and based on these decide whether the protein does contain the motif or not

Use basically any kind of function/method to go from residue-wise prediction to protein-wise:

- a linear discrimination function
- another neural network
- etc.

## Recapitulation – Artificial Neural Networks

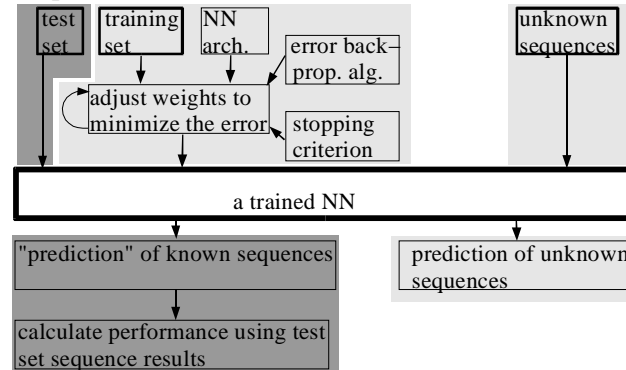
The inspiration comes from physiological neural networks

Neural networks are used for hard problems when "linear" methods won't work well

Idea: learn/train the NN to recognize patterns – of almost any kind *e.g.* sequence patterns

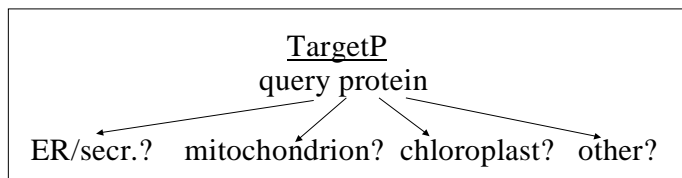
We want to get a good *generalization* performance, *i.e.* the NN should be able to classify patterns it hasn't seen before

## The feed-forward neural network – training, performance calculation prediction



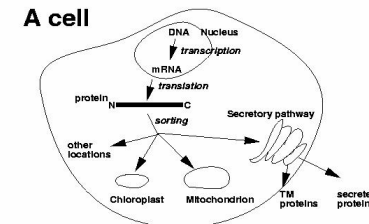
## TargetP: An example application of neural networks in molecular biology

TargetP – predicts the subcellular localization of proteins based on their amino acid sequence

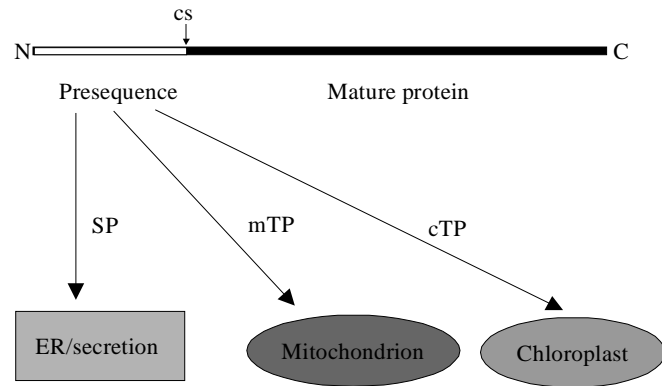


## What is the problem?

- Proteins are synthesized in the cytosol
- They need to be sorted to their proper location



## Three N-terminal presequences

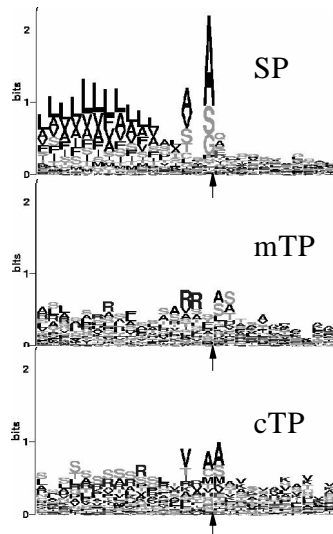


## The presequences; weak consensus

[I,V]-x-[A,C] ↓ A  
 Signal Peptide, SP  
 ~25 aa's

R-10 / R-3 / R-2 ↓  
 Mitochondrial Targeting Peptide, mTP  
 ~35 aa's

VRA ↓ AA  
 Chloroplast Transit Peptide, cTP  
 ~50 aa's



## Sequence logos

A way of visualising multiple sequence alignments, and the degree of conservation at the positions

Here: Plant sequences aligned around their annotated cleavage site

## Measuring the degree of conservation, $I$

Based on Shannon entropy,  $H$

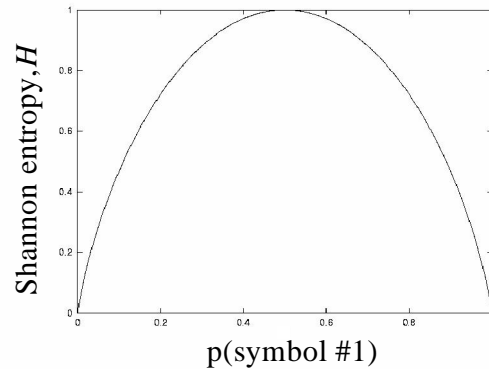
$$H = - \sum_{s=1}^n p_s \log_2 p_s, \text{ where}$$

$p_s$  is the probability of symbol  $s$ ,  
 $n$  is the number of symbols (=20 in amino acid case),

the degree of conservation,  $I$ , is (in amino acid case):

$$I = H_{max} - H = \log_2 20 - \sum_{s \in \{A, C, \dots, Y\}} p_s \log_2 p_s$$

## Entropy in the case of 2 symbols



## Data sets

1. Extract sequences from SWISS-PROT
2. Check in literature (for cleavage site reliability)
3. Remove too similar sequences (redundancy/homology reduction)

## Swiss-Prot rel. 40; 102 504 entries

Extensively annotated. Data checked by human experts. Kept well up-to-date. Free for academic use.

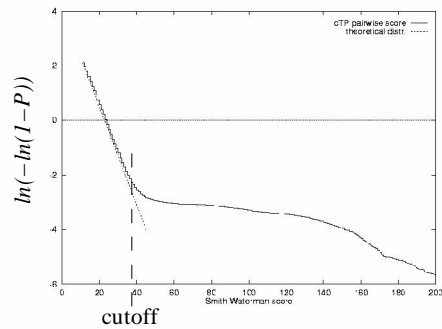
```
ID  ATPG_TOBAC  STANDARD:  PRT;  377 AA.
AC  P29790;
DT  01-APR-1993 (Rel. 25, Created)
DT  01-APR-1993 (Rel. 25, Last sequence update)
DT  01-OCT-1994 (Rel. 30, Last annotation update)
DE  ATP SYNTHASE GAMMA CHAIN, CHLOROPLAST PRECURSOR (EC 3.6.1.34).
GN  ATPC.
OS  Nicotiana tabacum (Common tobacco).
OC  Eukaryota; Viridiplantae; Embryophyta; Tracheophyta; Spermatophyta;
   Magnoliophyta; eudicotyledons; core eudicots; Asteridae; euasterids I;
   Solanales; Solanaceae; Nicotiana.
/.
RX  MEDLINE=92322965 [NCBI, Expasy, Israel, Japan]; PubMed=1535803;
RA  Larsson K.H., Napier J.A., Gray J.C.;
RT  "Import and processing of the precursor form of the gamma subunit of
   the chloroplast ATP synthase from tobacco.";
RL  Plant Mol. Biol. 19:343-349(1992).
/.
FT  TRANSIT      1    55    CHLOROPLAST.
FT  CHAIN        56   377    ATP SYNTHASE GAMMA CHAIN.
FT  ACT_SITE     143   143    BY SIMILARITY.
FT  DISULFID     253   259    BY SIMILARITY.
SQ  SEQUENCE 377 AA; 41446 MW; 60A262F08013F3E0 CRC64;
MSSNSLTMV SSKPFLSDSS ALSFRSSVSP FQLPNNHTFG PSNFSRSSSV TPVHCGLRDL
RRIIEVKEK QKITEAMKLV AAKRVRAGS AVVQAPFSE TLVEVLEIN EQIQTEDIV
PLTKVRPVKK VALVVVTGDR GLCGGFNNYL IKKARARLR LKALGIDYTI ISVGKGNYSY
FIRRPYIPVD KFLEGSNLPT AKDAQALADD VSLFVSEVH DKVELLYTFF VSLVKSEPVV
HLLLELRPKS EICINDNICY DANRPFEL TTKRGLVYE RDIIRKTTD PSILQFQKQ
PVQILDALLE LVYNSQLIRA LQESLASELA ARMSAMSSAT DNATELKKNL SRVYNRQQA
KITGEILEIV AGADALV
```

## Removing too similar sequences

1. Pairwise alignment (Smith-Waterman, PAM250)
2. Compare distribution of scores to extreme value distribution to get a similarity cutoff
3. Count the "neighbours" of each protein (*i.e.* pairwise alignment score > cutoff)
4. Remove the protein with the highest number of neighbours
5. Repeat (3) and (4) until no neighbours left

## The extreme value distribution

$$P_{score \geq x} = 1 - \exp(-e^{-\lambda(x-u)})$$

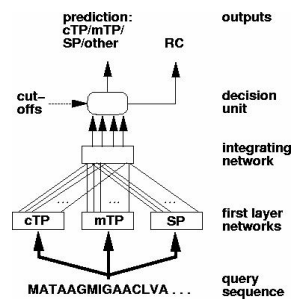


## TargetP data sets

(number of proteins before/after redundancy reduction)

<u>Plant sets</u>		<u>Non-plant sets</u>	
cTP	432 141	–	
mTP	658 368	mTP	702 371
SP	648 269	SP	2292 715
other	751 162	other	6311 1652

## The neural networks of TargetP



Some technical details:

Logistic neurons

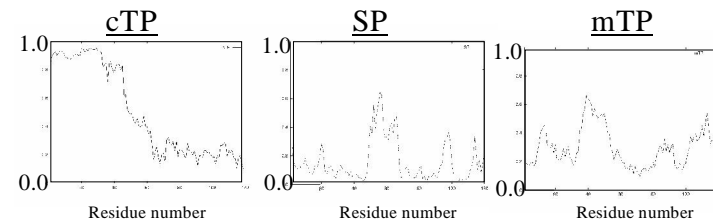
Feedforward connections

Trained using error back-propagation and 5-fold cross-validation

An "integrating network" used for post-processing

## Neural network output

–The output of the first layer networks looks like this:



– This is the input to the integrating network.

## TargetP output

### ### ### T A R G E T P 1.1 prediction results ### ### ###

# Number of input sequences: 4  
 # Cleavage site predictions included.  
 # Using PLANT networks.

#	Name	Length	cTP	mTP	SP	other	Loc	RC	TPlen
#-----									
	P11043	516	0.873	0.012	0.004	0.320	C	3	65
	P07505	222	0.977	0.015	0.002	0.046	C	1	67
	P12352	97	0.397	0.555	0.014	0.150	M	5	40
	P48786	688	0.199	0.070	0.067	0.822	-	2	-
#-----									
# cutoff			0.00	0.00	0.00	0.00			

## The training of TargetP

For each of the NNs, several parameters were tested:

- learning rate: 0.001, 0.05, 0.01
- sliding window size: 7-55 residues
- number of epochs: 200-1000
- number of nodes in hidden layer (0-10)

... and the best performing parameter combination was chosen

## TargetP results

Plant test set (redundancy reduced): 940 proteins.  
 In total 85.3% correct.

		<i>predicted</i>				<i>sens.</i>
		cTP	mTP	SP	other	
<i>true</i>	cTP	120	14	2	5	0.85
	mTP	41	300	9	18	0.82
	SP	2	7	245	15	0.91
	other	10	13	2	137	0.85
<i>spec.</i>		0.69	0.90	0.96	0.78	

## Psi-pred, another NN application

Psi-pred predicts secondary structure (2D str.) of proteins using neural networks.

Feed-forward networks with sigmoidal nodes, trained using error backpropagation algorithm.

New: Using *sequence profiles* as input!

A way to incorporate evolutionary information in the predictions

## Psi-pred, another NN application

1. Using Psi-blast to create profiles
2. Profile used as the input instead of sparse encoding:  
(sequence in window: FD...L)

```

      A R N D C Q E G H I L K M F P S T W Y V
1:  -3-4-4-4-3-4-4-4-2-1-1-4-1 8-5-3-3 0 2-2
2:   0-1-1 3-4 3 4 1-1-4-4 0-3-4-2-1-2-4-3-3
...
15: -2-3-1-5-3-3-4-5-4 3 4 0 4 2-4-3-2-3-2 0
}

1:   0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
2:   0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
...
15:  0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
}

```

## Psi-pred, another NN application

Want a 2<sup>nd</sup> structure assignment to all residues:

```

GYPSLSALRAAHREGVAYE
---HHHHH---EEEE-E--

```

Thus, here we do *not* want only one score per protein

Post-processing still necessary to "clean" the output

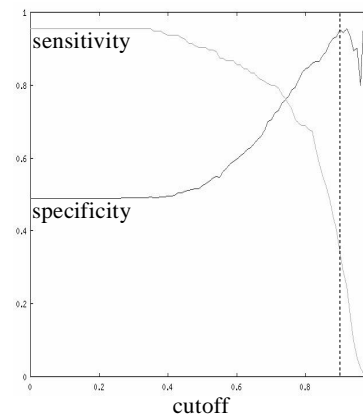
## More about Neural Networks

Neural networks. A comprehensive foundation *by* Simon Haykin

Neural Networks for Pattern Recognition *by* Christopher M Bishop

Bioinformatics. The Machine Learning Approach *by* Pierre Baldi *and* Søren Brunak: chapters 5 and 6

## Specificity vs. sensitivity



- Modulate cutoff to obtain desired sens/spec balance
- Example: TargetP prediction of mTP on human set (from Swiss-Prot)