Lecture 8:

Supervised Learning: Case Study for Proteins

Week of February 6, 2023

University California, Berkeley

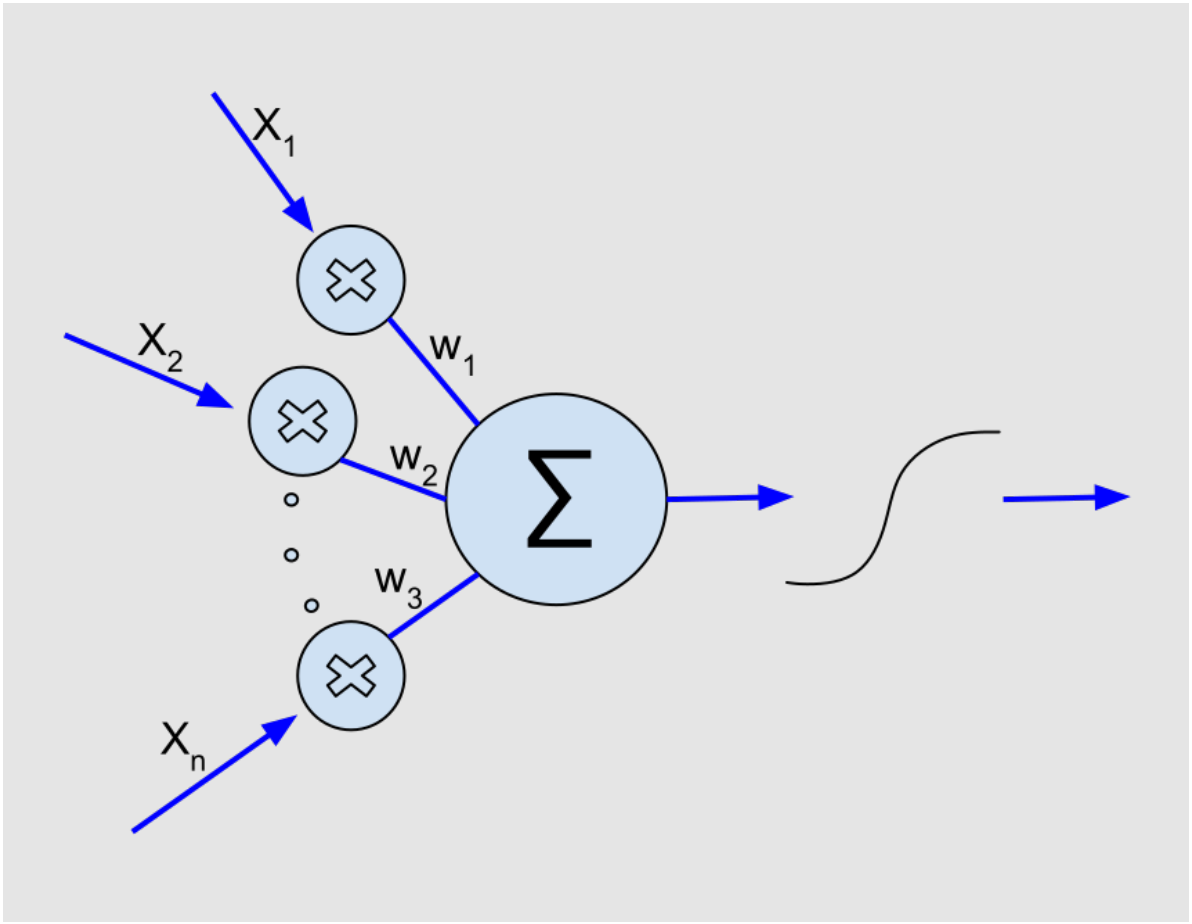
Machine Learning Algorithms

MSSE 277B, 3 Units

Spring 2023

Prof. Teresa Head-Gordon

Departments of Chemistry, Bioengineering, Chemical and Biomolecular Engineering



Review Previous Lecture: Simple Perceptron Learning

One of the best applications of simple neural networks is as optimization for classification of groups that are linearly separable. Need to determine an n-dimensional decision plane that separates the solution space correctly into these classes, given the n-dimensional input.

- the simple perceptron can solve classification of output for a simple OR or simple AND. This simple computing element can classify with use of a single decision plane.
- But in the case of XOR function, we found that the NN complexity had to increase to include a hidden layer of neurons to successfully classify the XOR output, in particular requiring two decision planes

Today's Lecture: Supervised Learning

Lecture Purpose: Supervised ANN's are often applied to classification problems where you have many, many objects that you would like to group into far fewer number of categories, but their separations involve non-linear decision boundaries.

Today we will see a NN in practice for secondary structure prediction. From this we will learn about:

Training and Testing data

Data quality

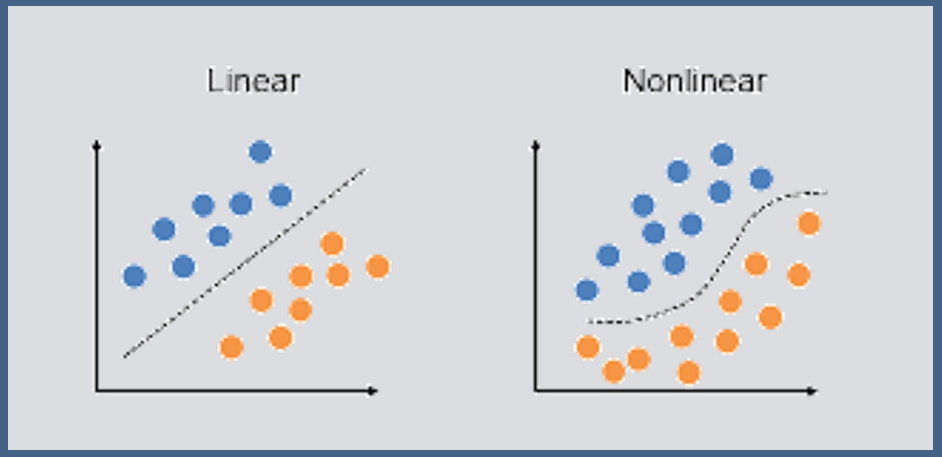
NN architectures

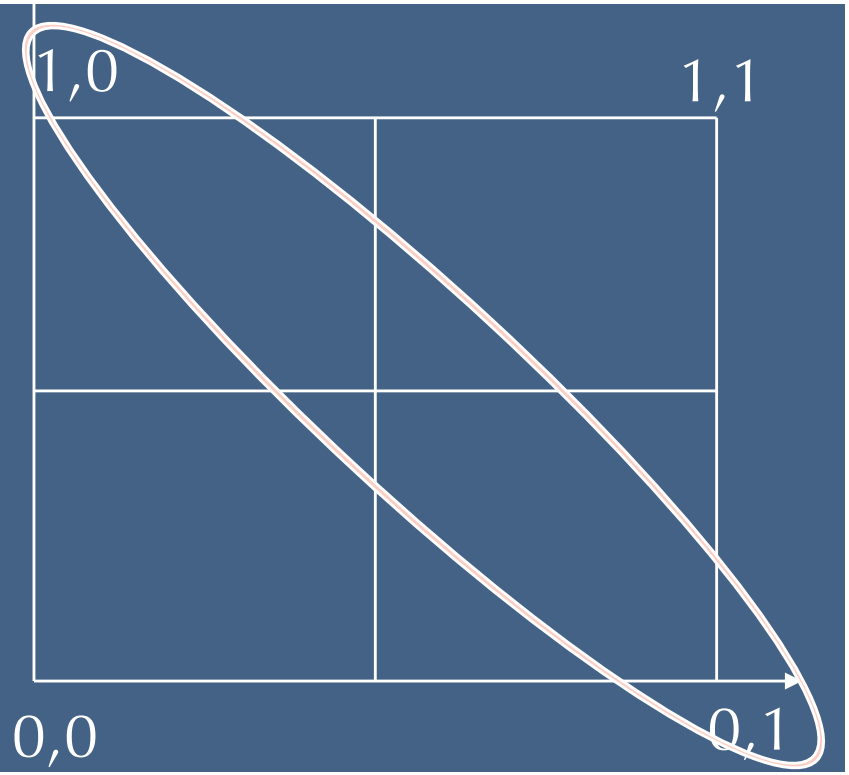
Input/Output representation

Measuring performance

Generalization

Supervised Learning





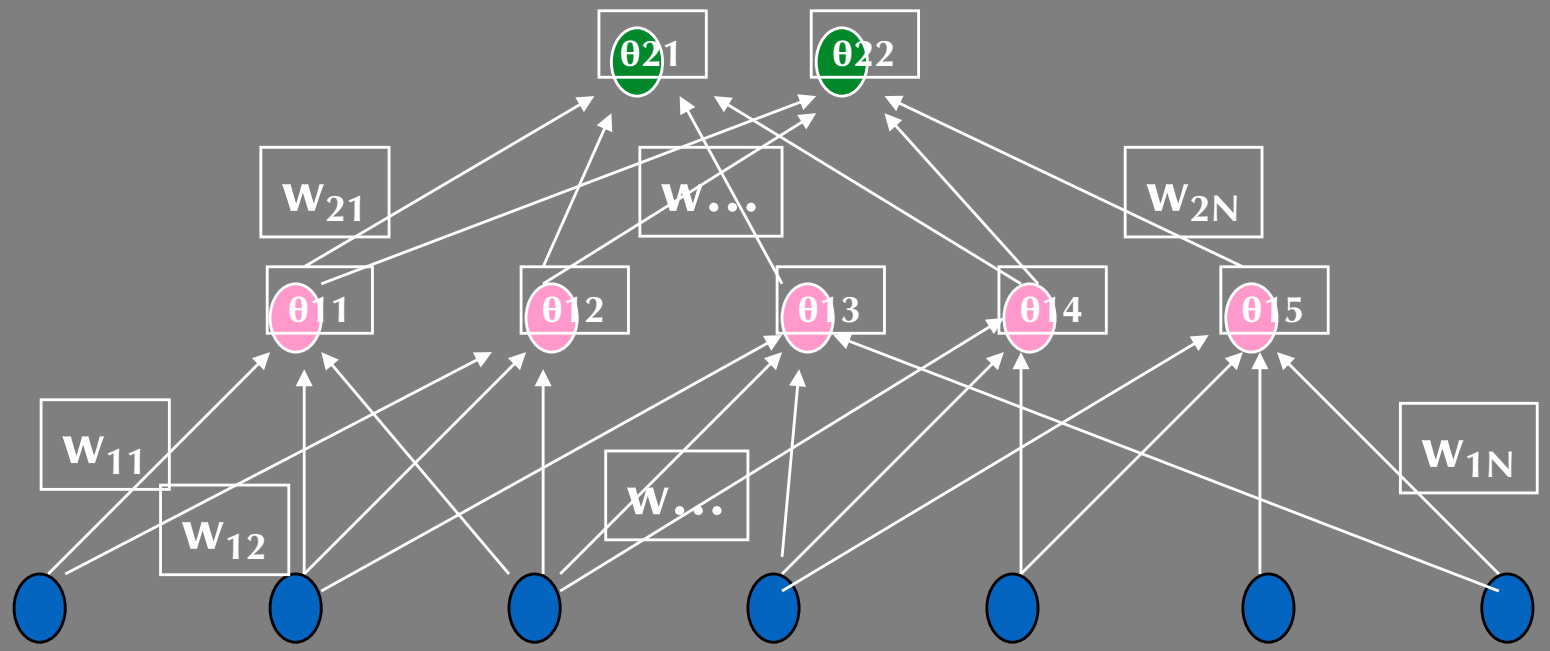
For linearly separable data the Perceptron Learning Rule determined the direction that the weight vector needed to change to bring the output closer to the right side of the decision boundary, by changing $\boldsymbol{cos\alpha}$

But often the decision plane is non-linear, and the weight and threshold update rule is not as simple. Therefore the learning algorithm adjusts the network variables to minimize the difference between the true observables and calculated ones

$$C = \frac{1}{2}\sum_{\mu}^{N}\sum_{j}^{J}\left[O_j^{\mu} - y_j^{\mu}\right]^2$$

since we must learn this nonlinear decision boundary.

$$C = \frac{1}{2}\sum_{\mu}^{N}\sum_{j}^{J}\left[O_j^{\mu} - y_j^{\mu}\right]^2$$



Feed forward

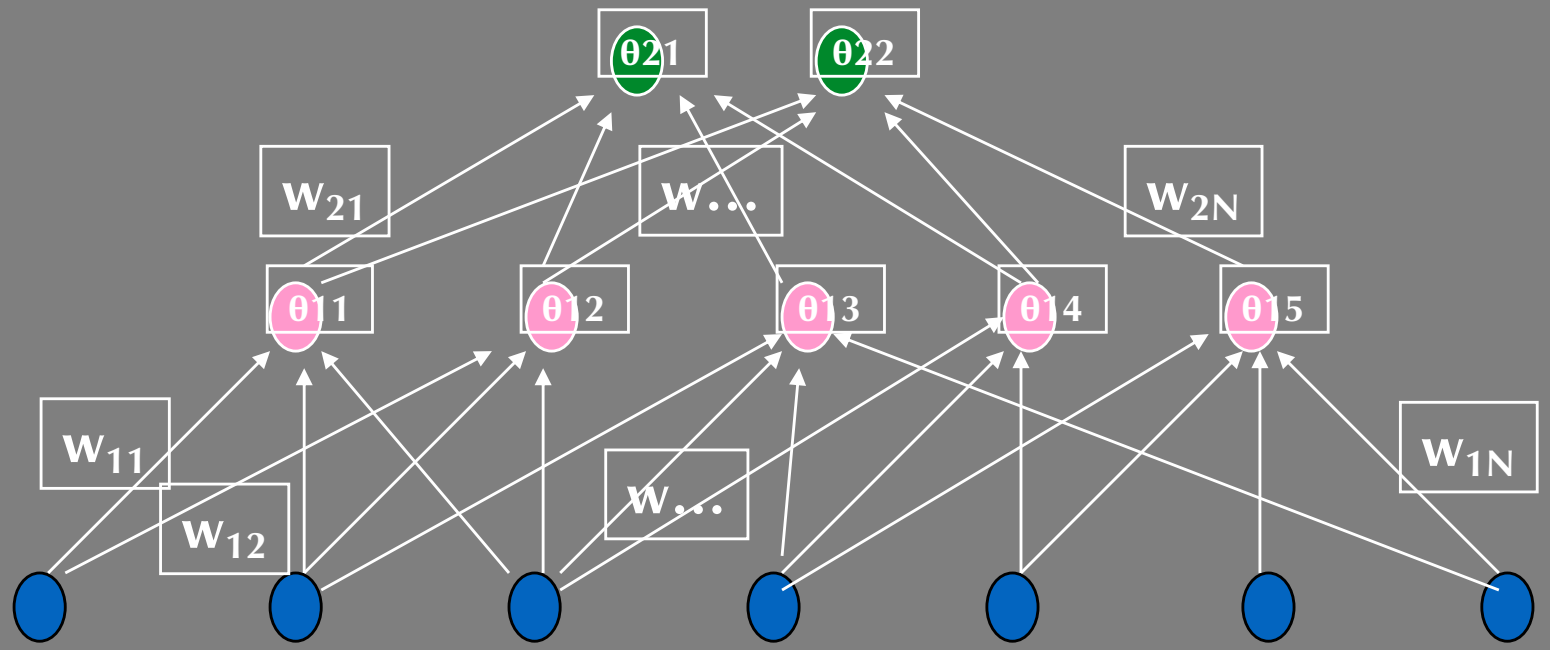
Back propagate

(1) Feed forward and evaluate all $\boldsymbol{y_i^{\mu}}$: this classifies data into groups (maybe an encoding in which y=1,1 vs y=0,0 represents two different groups)

(2) Evaluate $\boldsymbol{C}$; if $\boldsymbol{C}$ > tolerance then

Feed Forward-Back Propagation NN's

$$C = \frac{1}{2}\sum_{\mu}^{N}\sum_{j}^{J}\left[O_j^{\mu} - y_j^{\mu}\right]^2$$



Feed forward

Back propagate

(3) Back-propagate. Start by adjusting weights and biases connecting output to previous layer:

$$\delta w_{ij} = -\varepsilon\frac{\partial C}{\partial w_{ij}} = -\varepsilon\sum_{\mu}\left[O_j^{\mu} - y\left(\Sigma_j^{\mu}\right)\right]\frac{-dy}{d\Sigma_i^{\mu}}\frac{\partial\Sigma_i^{\mu}}{\partial w_{ij}} \qquad \Sigma_i^{\mu} = \sum_j w_{ij}x_j - \theta_j$$

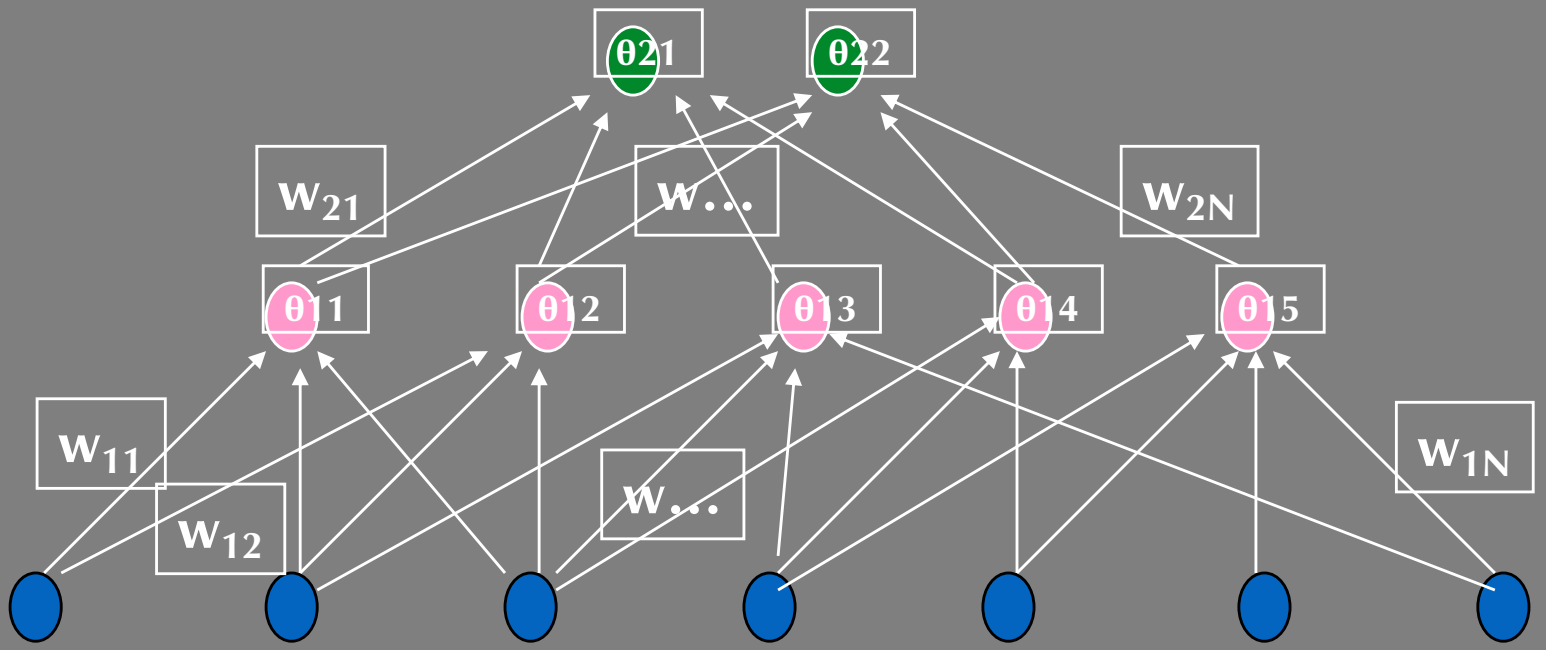
Notice like a steepest descent step with learning rate ε

Feed Forward-Back Propagation NN's

$$C = \frac{1}{2}\sum_{\mu}^{N}\sum_{j}^{J}\left[O_j^{\mu} - y_j^{\mu}\right]^2$$

Feed forward

Back propagate



(4) Then adjust weights connecting hidden layer j to hidden layer k.

$$\delta w_{jk} = -\varepsilon\frac{\partial C}{\partial w_{jk}} = \varepsilon\sum_{\mu}\sum_{i}\left[O_i^{\mu} - y(\Sigma_i^{\mu})\right]\frac{dy}{d\Sigma_i^{\mu}}\frac{\partial\Sigma_i^{\mu}}{\partial s_j}\frac{\partial s_j}{\partial w_{jk}}$$

$$\frac{\partial s_j}{\partial w_{jk}} = \frac{dy\left(\Sigma_j^{\mu}\right)}{d\Sigma_j^{\mu}}\frac{\partial\Sigma_j^{\mu}}{\partial w_{jk}}$$

Feed Forward-Back Propagation NN's

Qian & Sejnowski we first to try to learn the classification of amino acid sequence patterns and their mapping onto observed secondary structure: helix, sheet, coil.

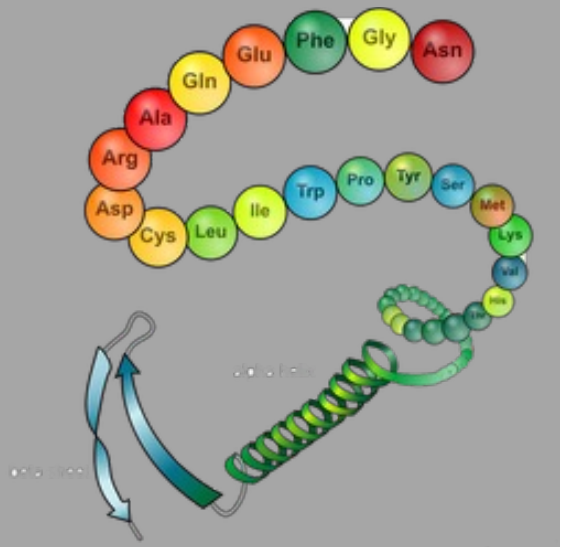
Input [aa-sequence]→output [2° structure]

Training set: Protein databank (PDB) with many examples of mapping:

- .....LSADQISTVQASF……input
- .....HHHHCCCCEEEEE…output

Training set is further decomposed into amino acid "windows" of 9-17 amino acids in length, and input-output is focused on central residue of the window

..AALSADQISTVLLSFYKLAKQ..
..HHHHHHCCCCEEEEEECCCCC..



ANN Classification: Protein Secondary Structure

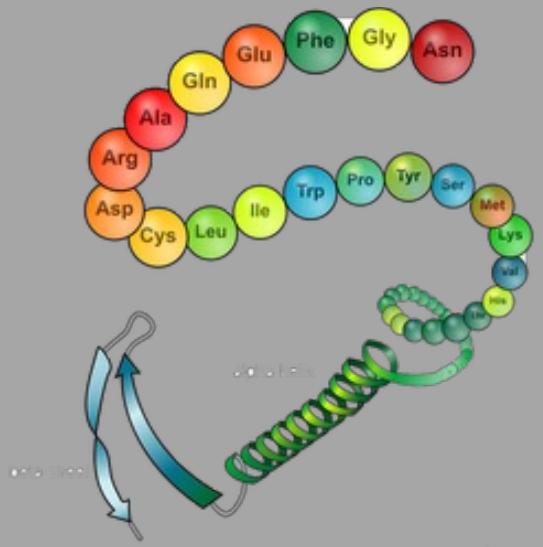
Training set has may input-output relationships

..AALSADQISTVLLSFYKLAKQ..
..HHHHHHCCCCEEEEEECCCCC..

Training pattern μ

..AALSADQISTVLLSFYKLAKQ..
..HHHHHHCCCCEEEEEECCCCC..

Training pattern μ+1



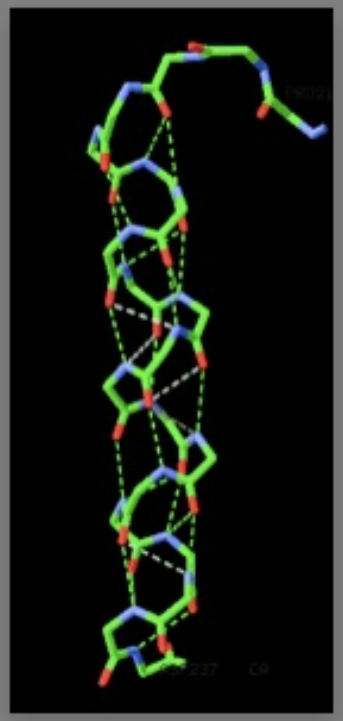
Training set was composed of 100-120 non-homologous proteins, and therefore there were ~15,000-22,500 patterns of 17 amino acid windows.

Note there is an assumption that a given x-ray solution by different researchers use same definition of 2° assignment (which they do not). Therefore must train with a consistent definition

ANN Classification: Protein Secondary Structure

Dictionary of Secondary structure in Proteins (DSSP)

Dictionary of Secondary structure in Proteins



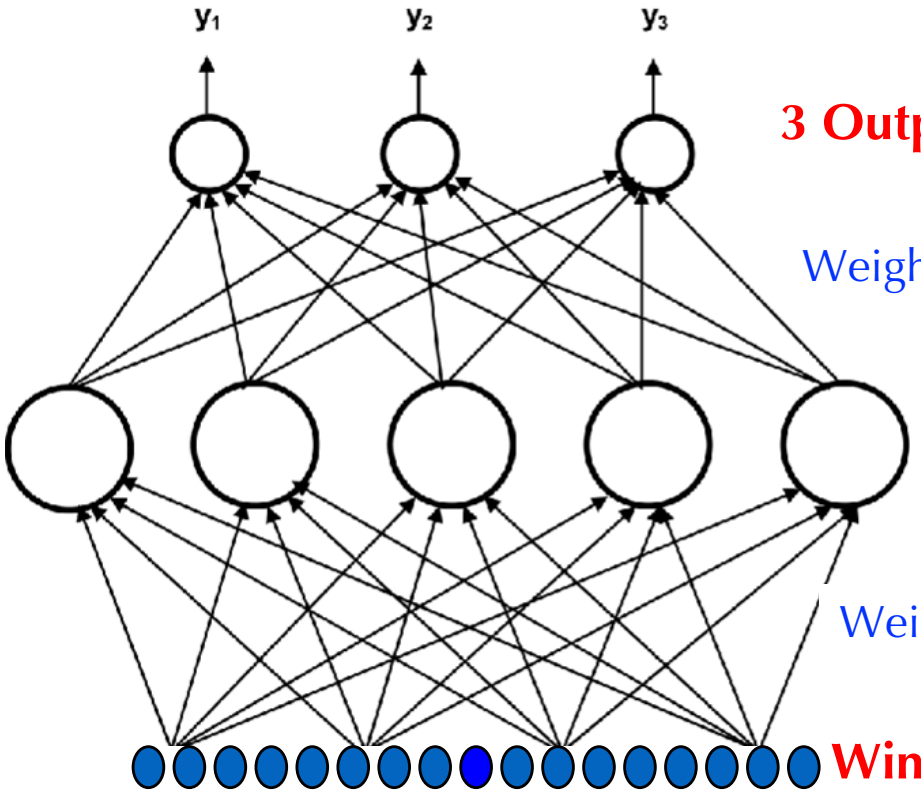
Provides a standardized definition based on definition of allowed ranges in ϕ and ψ, and hydrogen-bond geometry.

DSSP Class	2° assignment NN
H,G	H
E	E
B,I,S,T	C

where

H=helix	I=pi-helix
G=3/10 helix	S=bend
E=extended b-strand	T=turn
B=isolated b-bridge	C=coil

Neural Network Architecture



Number of adjustable variables:

17aa x 21input units/aa = 357 input nodes

357 input x 5 hidden nodes = 1785 weights

5 hidden x 3 output nodes= 15 weights

8 thresholds

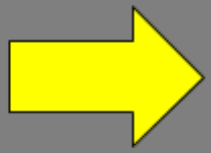
Total 1808 adjustable variables

Rule of thumb: # patterns should be ~20 x # network variables.

20x1808 = 36160

(actually used ~0.5 # patterns)

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Output representation: Binary

- 100 H
- 010 E
- 001 C

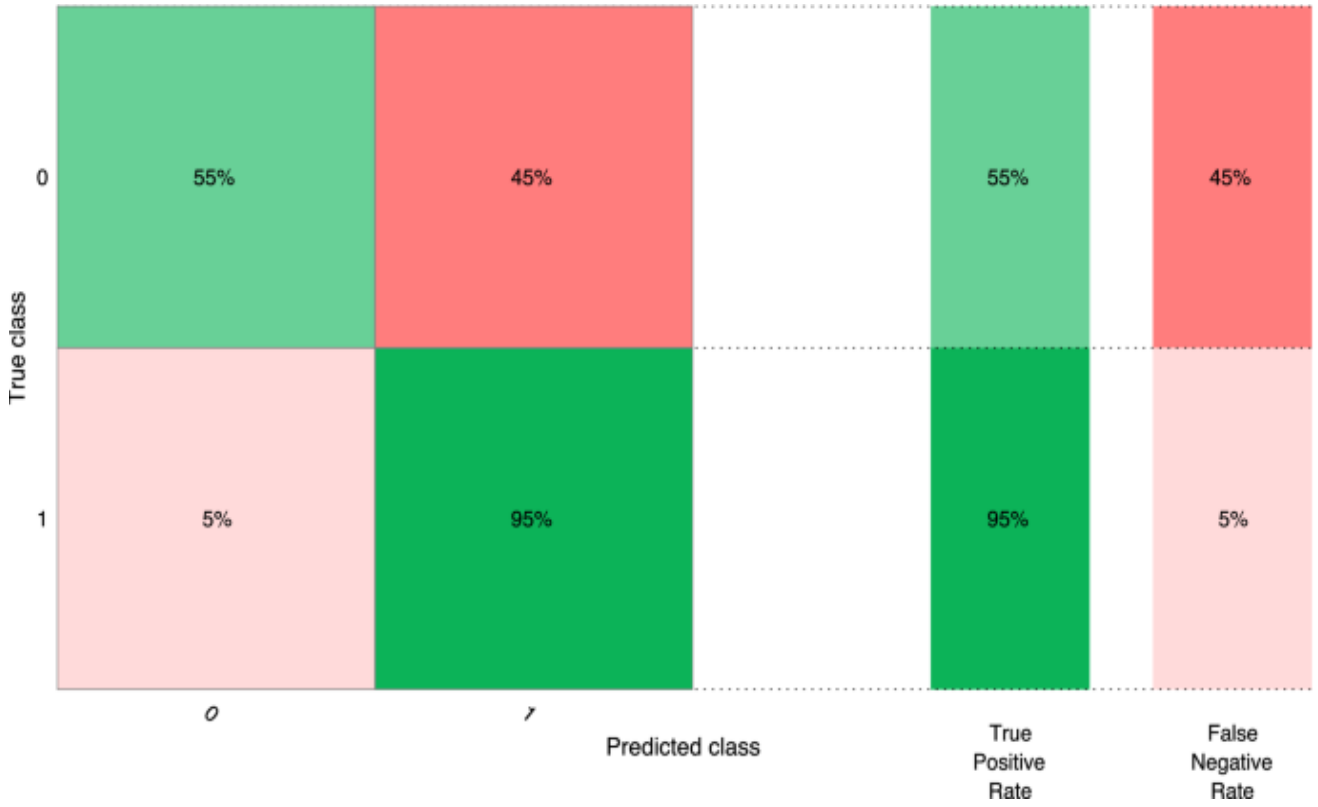
Input representation: Binary

- 10000000000000000000 Gly
- 01000000000000000000 Ala
- 00100000000000000000 Ser
- Etc

We will return to this issue of input/output encoding later in the lecture…

Input/Output Representation: One Hot Encoding

Measuring ANN Performance



At the end of training we can measure performance on correctly classifying aa sequence patterns into predicted secondary structure category

Q_3: % of correctly classified patterns among H, E, C. This is a weak measure of success. For example, database is typically distributed as

- 20% sheet
- 30% helix
- 50% coil

So could "predict" that everything is coil and be right 50% of the time! Q_3 does not reflect how badly I under-predicted H and E, and how badly I over-predicted C.

Measuring Performance

		True/Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

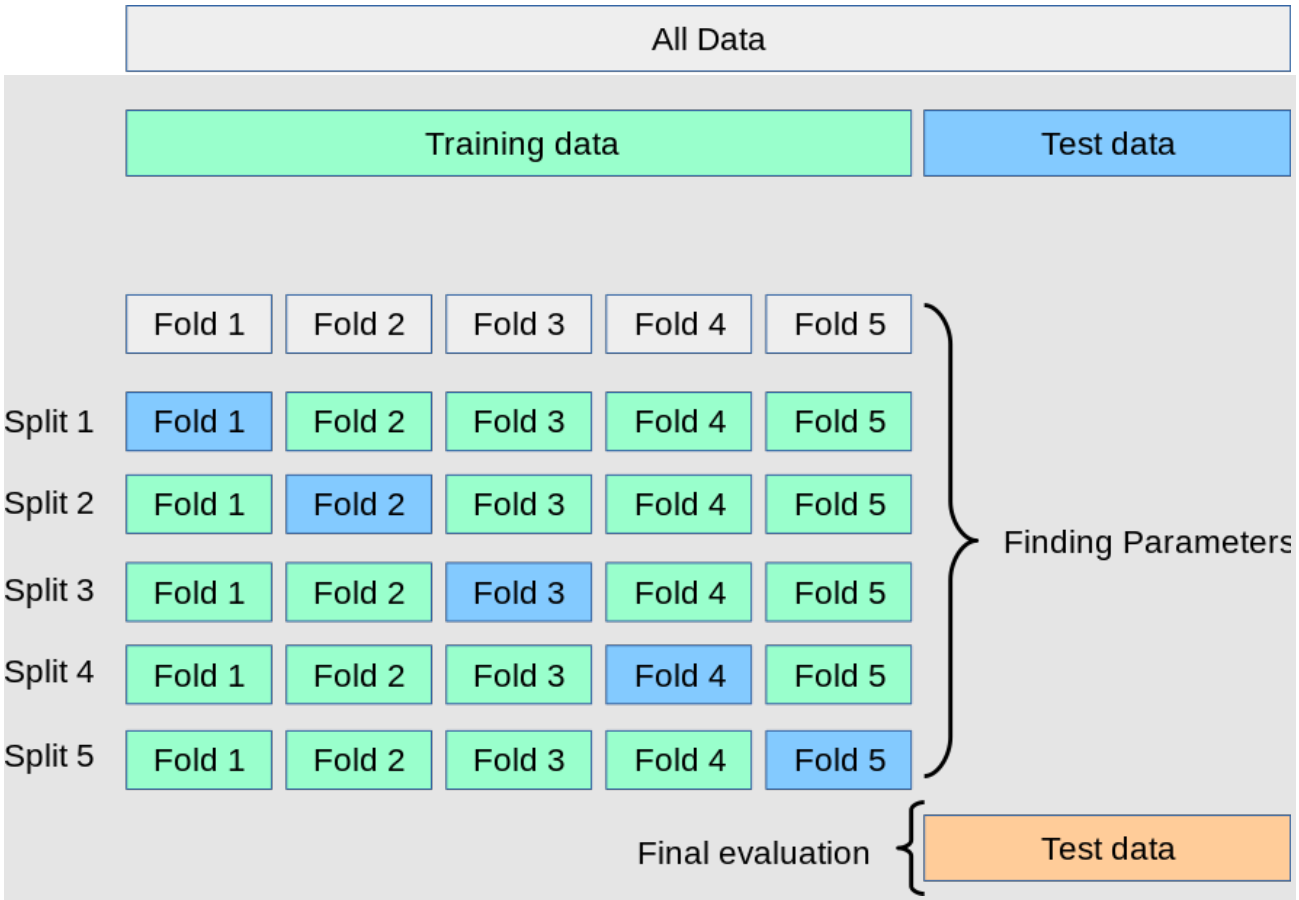
(2) C_{2°: Correlation coefficients.

$$C_{2^\circ} = \frac{pn}{\left([p+o][p+u][n+o][n+u]\right)^{1/2}}$$

- p = patterns correctly predicted to be 2°
- n = patterns correctly predicted to NOT be 2°
- o = patterns incorrectly predicted to be 2°
- u = patterns incorrectly predicted to NOT be 2°

$C_{2^\circ} = 0$ no correlation; $C_{2^\circ} = 1$ perfect correlation

N-fold Cross Validation



Full jack knife testing: train network P times on a training set of P-1 patterns, test on single remaining pattern.

Not possible with ~10^4-10^5 patterns!

Training	Validation
Proteins 1-100	Proteins 101-120
Proteins 1-80, 101-120	Proteins 81-100
Proteins 1-60, 81-120	Proteins 61-80
Proteins 1-40, 61-120	Proteins 41-60
Proteins 1-20, 41-120	Proteins 21-40
Proteins 21-120	Proteins 1-20

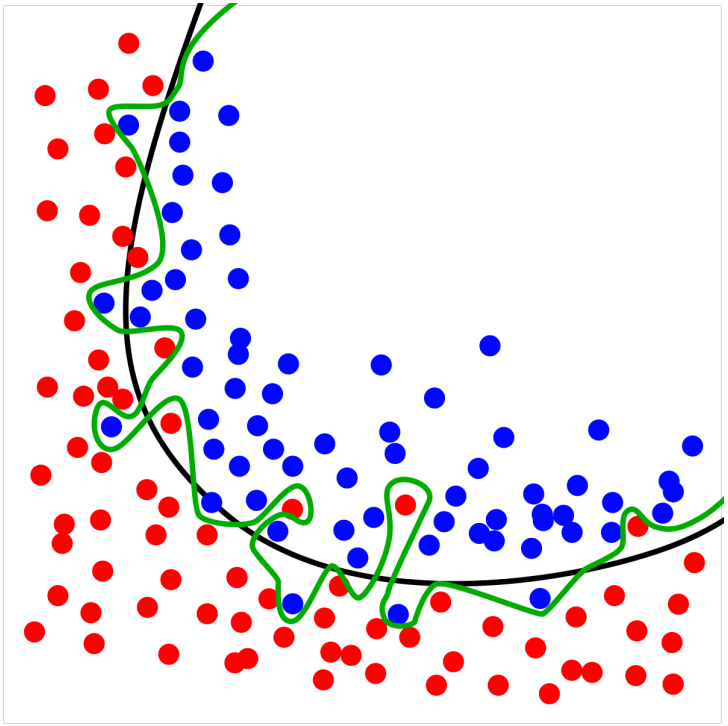
And average Q_3 and C_{2° results

Qian & Sejnowski ANN Results

Train	Test
• Q_3 =66%	Q_3 =62%
• Q_H=51% C_H=0.42	Q_H=48% C_H=0.38
• Q_E=38% C_E=0.39	Q_E=28% C_E=0.31
• Q_C=82% C_C=0.36	Q_C=84% C_C=0.35

We interpret this to mean that the NN supervised learning was able to understand some aspects of aa=>2° structure mapping

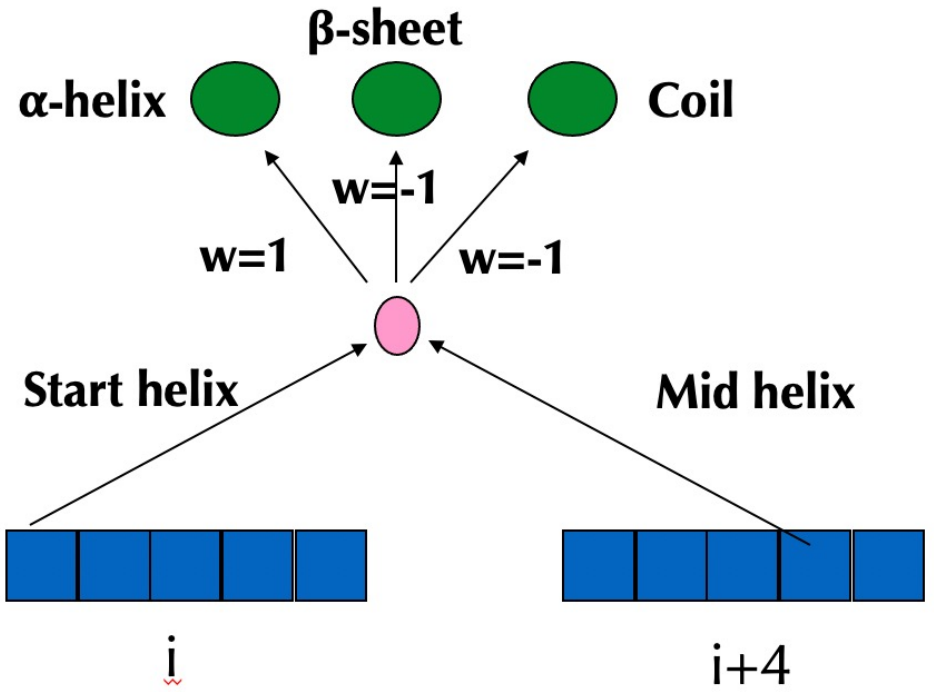
However generalization of learning was not well translated to test set as we see systematic degradation in performance by all measures



What are possible Improvements?

- Predictions were based on local amino acid context (17 aa window); undoubtedly long-range sequence effects are important.
- Input representation was opaque, i.e one-hot encoding was unhelpful. AA's should be made more distinguishable according to physiochemical properties.
- Predictions made in isolation at each aa position; no knowledge of 2° prediction of surrounding amino acids, global content
- Predictions made with only a single network topology- perhaps more optimal topology?
- Too many network variables given number of training patterns?

Engineered Features



(1) Change Input representation. Each amino acid is represented as a set of 5 real numbers [-1,1]. Amino acids are ranked for their propensity in 5 categories:

- 1st position: propensity to start a helix
- 2nd position: propensity to be in middle of helix
- 3rd position: propensity to end a helix
- 4th position: propensity to be in sheet
- 5th position: propensity to be hydrophobic

(2) Reduces network variables considerably:

- 5x17=85 weights
- 425+15+8=448 network variables
- Need 448x20=9400 patterns

(3) Network design:

- Recognize context features of aa=>2°

Engineered vs One Hot Encoding

No Design:

Train		Test	
Q_3 =66%		Q_3 =62%	
Q_H=51%	C_H=0.42	Q_H=48%	C_H=0.38
Q_E=38%	C_E=0.39	Q_E=28%	C_E=0.31
Q_C=82%	C_C=0.36	Q_C=84%	C_C=0.35

Overall improvement, especially in structured categories!

Designed (Yu and Head-Gordon, Phys Rev E 1995):

Train		Test	
Q_3 =67%		Q_3 =67%	
Q_H=66%	C_H=0.52	Q_H=64%	C_H=0.48
Q_E=63%	C_E=0.46	Q_E=53%	C_E=0.43
Q_C=69%	C_C=0.43	Q_C=73%	C_C=0.44