MCB 128 AI in Molecular Biology

MCB 128 AI in Molecular Biology

Welcome!

Elena Rivas

Shivam Gandhi

Armand Ovanessians

Louis Colton

Philosophy/Objectives

- Learn first principles of DL

Philosophy/Objectives

- Learn first principles of DL
- Learn applications in molecular biology

Philosophy/Objectives

- Learn first principles of DL
- Learn applications in molecular biology
- Not a survey course

Philosophy/Objectives

- Learn first principles of DL
- Learn applications in molecular biology
- Not a survey course
- Will implement simplified version of the methods

Philosophy/Objectives

- Learn first principles of DL
- Learn applications in molecular biology
- Not a survey course
- Will implement simplified version of the methods
- Learn to use chatbots critically (and without losing your own identity)

Philosophy/Objectives

- Learn first principles of DL
- Learn applications in molecular biology
- Not a survey course
- Will implement simplified version of the methods
- Learn to use chatbots critically (and without losing your own identity)
- Learn to design your own questions

Philosophy/Objectives

- Learn first principles of DL
- Learn applications in molecular biology
- Not a survey course
- Will implement simplified version of the methods
- Learn to use chatbots critically (and without losing your own identity)
- Learn to design your own questions
- Question the jargon

0,1, ..., 6 blocks

5 quizzes (20%)

6 + 1 homework (70%)

Block	week	Dates 2026	Description	Due
b0	**b0-w1**	01/26,01/28 01/30	A single neuron / RNA functional classification [1, 2] section_b0-w1 (Colab,Pytorch)	hw-b0 out
b1	**b1-w1**	02/02,02/04 02/06	Multi-layer-Perceptron / Protein 2D structure [3, 4] section_b1-w1	hw_b1 out hw_b0 due
	b1-w2	02/09,02/11 02/13	Fundamental of Neural Network Training / [5, 6] section_b1-w2	quiz_b1
b2	**b2-w1**	02/18 02/20	Convolutional Neural Networks / DNA/RNA sequence binding motifs [7–9] section_b2-w1	hw_b2 out hw_b1 due
	b2-w2	02/23,02/25 02/27	Recurrent Neural Networks / Regulatory motif prediction [10] Splice site prediction [11] section_b2-w2	quiz_b2

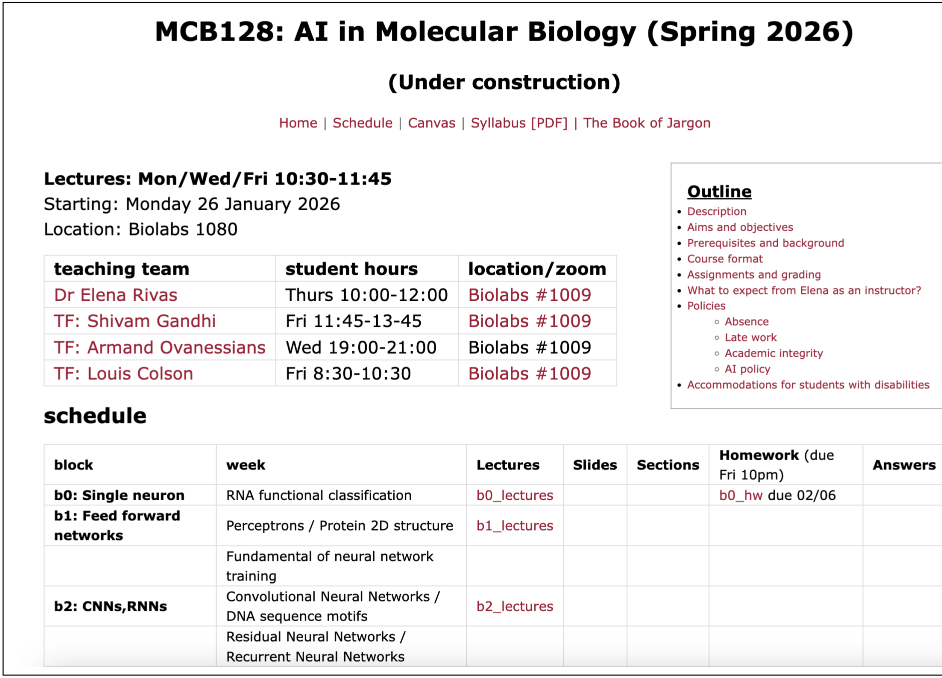
b4	**b4-w1**	03/23,03/25 03/27	Large Language Models (LLMs) [15] section_b4-w1	hw_b4 out hw_b3 due
	b4-w2	03/30,04/01 04/03	LLMs for DNA/RNA, [16] proteins, [17–19] and genomes [20] section_b4-w2	quiz_b4
b5	**b5-w1**	04/06,04/08 04/10	AutoEncoders / Gene expression profiles [21] section_b5-w1	hw_b5 out hw_b4 due
	b5-w2	04/13,04/15 04/17	Variational AutoEncoders / scRNA-seq [22] section_b5-w2	quiz_b5
b6	**b6-w1**	04/20,04/22 04/24	Diffusion Models / Protein design [23] section_b6-w1	hw_b6 out hw_b5 due
	b6-w2	04/27,04/29	Graph Neural Networks / Antibiotic discovery [24]	
Final homework		05/06		hw_b6 due

Some logistics

- canvas discussions
- video recordings
- colab/Google cloud cupons
- Questions for sections

The mcb128 Website

http://rivaslab.org/teaching/MCB128_AIMB/



MCB128: AI in Molecular Biology (Spring 2026)

(Under construction)

Home | Schedule | Canvas | Syllabus [PDF] | The Book of Jargon

Lectures: Mon/Wed/Fri 10:30-11:45
Starting: Monday 26 January 2026
Location: Biolabs 1080

teaching team	student hours	location/zoom
Dr Elena Rivas	Thurs 10:00-12:00	Biolabs #1009
TF: Shivam Gandhi	Fri 11:45-13-45	Biolabs #1009
TF: Armand Ovanessians	Wed 19:00-21:00	Biolabs #1009
TF: Louis Colson	Fri 8:30-10:30	Biolabs #1009

Outline

schedule

block	week	Lectures	Slides	Sections	Homework (due Fri 10pm)	Answers
b0: Single neuron	RNA functional classification	b0_lectures			b0_hw due 02/06	
b1: Feed forward networks	Perceptrons / Protein 2D structure	b1_lectures				
	Fundamental of neural network training					
b2: CNNs,RNNs	Convolutional Neural Networks / DNA sequence motifs	b2_lectures				
	Residual Neural Networks / Recurrent Neural Networks					

MCB128: AI in Molecular Biology (Spring 2026)

(Under construction)

block 0: A single neuron / DNA Funtional Classification

Outline

In this lecture, I follow David Mackay very closely. In particular his lectures 15 and 16, which correspond to Chapters 39, 41 and 42 of his book Information Theory, Inference, and Learning algorithms.

The original perceptron dates back to Frank Rosenblatt's paper "The perceptron: a probabilistic model for information storage and organization in the brain" from 1958.

As a practical implementation of a simple perceptron in molecular biology, we will study the work "Use of the 'Perceptron' algorithm to distinguish transational initiation sites in E. coli" ", by Stormo et al. (1986) in which they use a perceptron to identify translation initiation sites in E.coli.

A single neuron

Here is the code associated to this section

A single neuron or perceptron (Figure 1) has

- The **inputs** $\mathbf{x} = (x_1, \dots, x_I)$,
- Parameters $\mathbf{w} = (w_1, \dots, w_I)$, usually called the **weights**.
- One output y which is also called the **activity**,

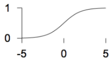
The neuron adds up the weighted sum of the inputs into a variable called the **activation** a,

$$a = w_0 + \sum_{i=1}^{I} w_i x_i$$

where w_0 called the **bias** is the activation in the absence of inputs.

The activity of the neuron y is a function of the **activation function** $f(a) = y$. Several commonly used forms for the activity are

- The **linear logistic function**



$$f(a) = \frac{1}{1+e^{-a}}$$

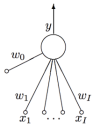
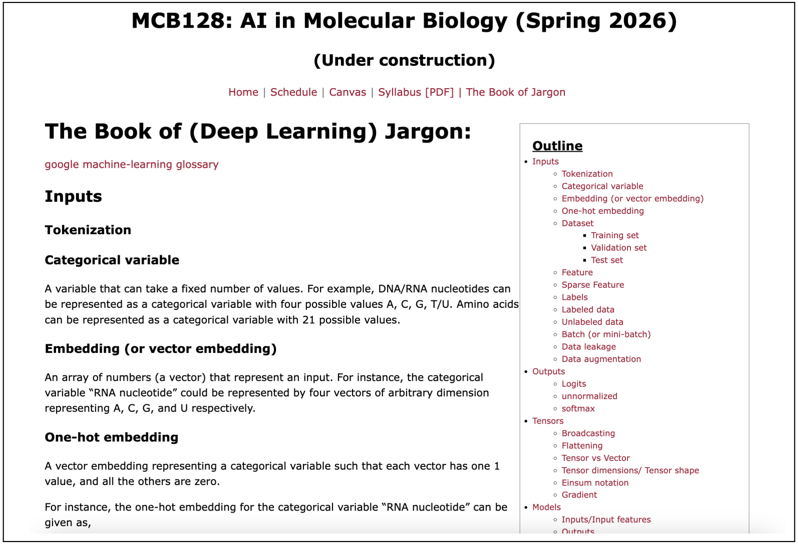


Figure 39.1. A single neuron

Figure 1. One neuron (from D. Mackay's chapter 39).

The book of Jargon



MCB128: AI in Molecular Biology (Spring 2026)

(Under construction)

Home | Schedule | Canvas | Syllabus [PDF] | The Book of Jargon

The Book of (Deep Learning) Jargon:

google machine-learning glossary

Inputs

Tokenization

Categorical variable

A variable that can take a fixed number of values. For example, DNA/RNA nucleotides can be represented as a categorical variable with four possible values A, C, G, T/U. Amino acids can be represented as a categorical variable with 21 possible values.

Embedding (or vector embedding)

An array of numbers (a vector) that represent an input. For instance, the categorical variable "RNA nucleotide" could be represented by four vectors of arbitrary dimension representing A, C, G, and U respectively.

One-hot embedding

A vector embedding representing a categorical variable such that each vector has one 1 value, and all the others are zero.

For instance, the one-hot embedding for the categorical variable "RNA nucleotide" can be given as,

Outline

Asking for contributions

The Wall of Pets/Friends/Plants

block b0

block b0

A single neuron

block b0

A single neuron

DNA funtional classification

A single neuron

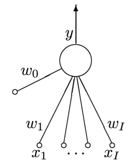


Figure 39.1. A single neuron

A single neuron or perceptron (Figure 1) has

- The **inputs** $\mathbf{x} = (x_1, \ldots, x_I)$,
- Parameters $\mathbf{w} = (w_1, \ldots, w_I)$, usually called the **weights**.
- One output y which is also called the **activity**,

The neuron adds up the weighted sum of the inputs into a variable called the **activation** a,

$$a = w_0 + \sum_{i=1}^{I} w_i x_i$$

where w_0 called the **bias** is the activation in the absence of inputs.

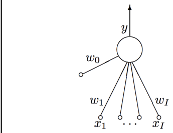


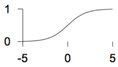
Figure 39.1. A single neuron

activation

$$a = w_0 + \sum_{i=1}^{I} x_i w_i$$

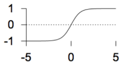
The **activity of the neuron** y is a function of the **activation function** f(a) = y. Several commonly used forms for the activity are

- The **linear logistic function**



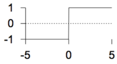
$f(a) = \frac{1}{1+e^{-a}}$

- The **sigmoid** (tanh) function



$f(a) = \tanh(a)$

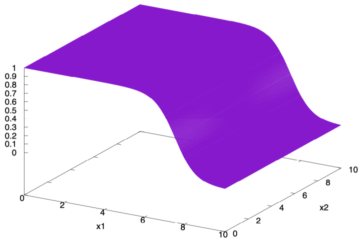
- The **step** function



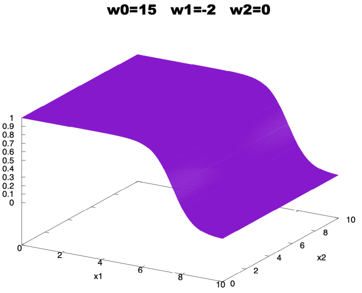
$$f(a) = \begin{cases} 1 & a > 0 \\ 0 & a \leq 0 \end{cases}$$

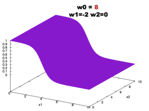
The space of weights

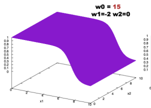
w0=15   w1=-2   w2=0

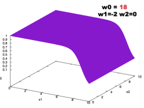


The space of weights

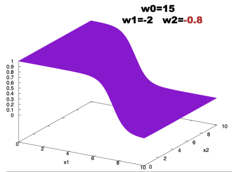




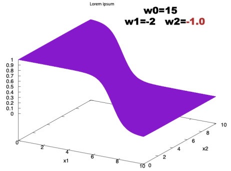


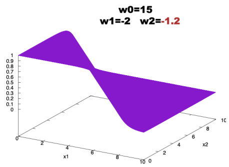


The space of weights

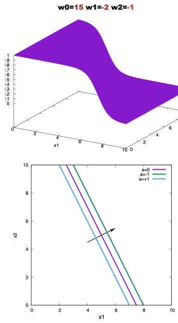


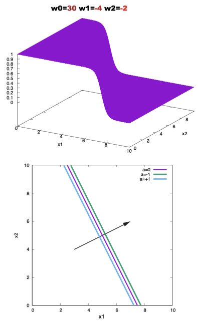
Lorem ipsum

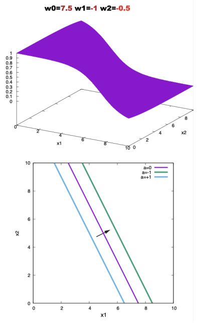




The space of weights







b0 Wednesday 1/28

b0 Wednesday 1/28

- colab/Google cloud cupons

b0 Wednesday 1/28

- colab/Google cloud cupons
- Questions for sections

b0 Wednesday 1/28

- colab/Google cloud cupons
- Questions for sections
- b0_homework due Friday 2/06 at 10 pm

b0 Wednesday 1/28

- colab/Google cloud cupons
- Questions for sections
- b0_homework due Friday 2/06 at 10 pm
- quiz on alternative Fridays–In person

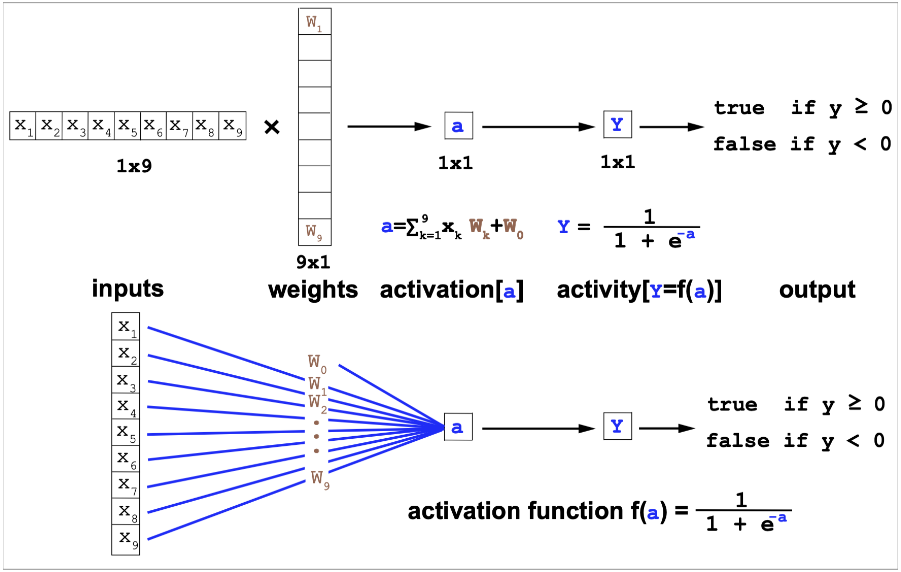
b0 Wednesday 1/28

- colab/Google cloud cupons
- Questions for sections
- b0_homework due Friday 2/06 at 10 pm
- quiz on alternative Fridays–In person
- student hours: Wed 7-9pm BL 1009

b0 Wednesday 1/28

- colab/Google cloud cupons
- Questions for sections
- b0_homework due Friday 2/06 at 10 pm
- quiz on alternative Fridays–In person
- student hours: Wed 7-9pm BL 1009
- rec: to read lecture notes before class / lecture notes demo code

Single Neuron (perceptron)



$$a=\sum_{k=1}^{9} x_k W_k + W_0 \qquad Y = \frac{1}{1+e^{-a}}$$

activation function $f(a) = \dfrac{1}{1+e^{-a}}$

What can a single neuron learn?

What can a single neuron learn?

to be a binary classifier

What can a single neuron learn?

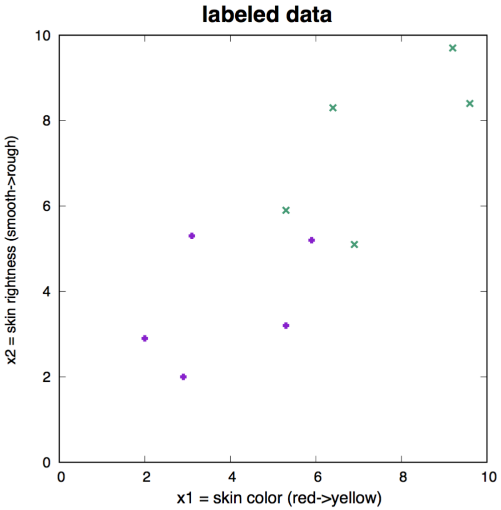
to be a binary classifier

separate apples from oranges

What can a single neuron learn?

to be a binary classifier

separate apples from oranges



The learning rule - Supervised learning

Data: $\mathrm{D} = \{\mathbf{x}^{(1)}, \mathrm{t}^{(1)}, \ldots, \mathbf{x}^{(\mathrm{N})}, \mathrm{t}^{(\mathrm{N})}\}$

Outputs: $\{\mathrm{y}^{(1)}, \ldots, \mathrm{y}^{(\mathrm{N})}\}$

Error: $\{\mathrm{y}^{(1)} - \mathrm{t}^{(1)}, \ldots, \mathrm{y}^{(\mathrm{N})} - \mathrm{t}^{(\mathrm{N})}\}$ where we expect these errors to be small.

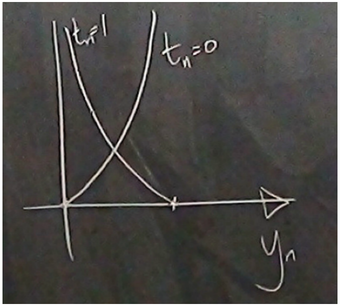
Learning is equivalent to adjusting the weights

such that the outputs of the network are close to the input labels.
$y^{(n)} \approx t^{(n)}$ for all $1 \leq n \leq N$ examples

The Error (Loss) Function

$$G(\mathbf{w}) = -\sum_{n=1}^{N}\left[t^{(n)}\log(y^{(n)}) + (1-t^{(n)})\log(1-y^{(n)})\right]$$

where $y^{(n)} = y(\mathbf{x}^{(n)}, \mathbf{w})$



Training the perceptron

training = minimize $G(\mathbf{w})$

Adjust the weights so that $G(\mathbf{w}) \geq 0$
is as small as possible

Backpropagation - gradient descent

Adjust the weights by gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{0ld} - \eta\,\mathbf{g}$$

where the **gradient** $\mathbf{g} = \frac{\delta G(\mathbf{w})}{\delta \mathbf{w}}$

η is **the learning rate**.

A bit of math to calculate the gradient of the Loss

The error/loss function is

$$G(\mathbf{w}) = -\sum_{n=1}^{N} \left[t^{(n)} \log(y^{(n)}) + (1 - t^{(n)}) \log(1 - y^{(n)}) \right]$$

We want to calculate

$$\frac{\delta G(\mathbf{w})}{\delta w_k} \quad for \quad 1 \leq k \leq K.$$

The dependency on the weights is hiding in the outputs

$$y^{(n)} = \frac{1}{1+e^{-\mathbf{wx}^{(n)}}} \qquad \text{with} \qquad \mathbf{wx}^{(n)} = \sum_{k=1}^{K} w_k \cdot x_k^{(n)}.$$

Taking the derivative wrt the weights using the chain rule

$$\frac{\delta G(\mathbf{w})}{\delta w_k} = -\sum_n \frac{\delta G(\mathbf{w})}{\delta y^{(n)}} \frac{\delta y^{(n)}}{\delta w_k}$$

The gradient of the Loss

The derivative of the loss wrt outputs is

$$\frac{\delta G(\mathbf{w})}{\delta y^{(n)}} = -\left[\frac{t^{(n)}}{y^{(n)}} - \frac{1-t^{(n)}}{1-y^{(n)}}\right] = -\frac{t^{(n)}-y^{(n)}}{y^{(n)}(1-y^{(n)})}$$

The derivative of the outputs wrt to the weights is

$$\frac{\delta y^{(n)}}{\delta w_k} = x_k^{(n)} \frac{e^{-\mathbf{w}\mathbf{x}^{(n)}}}{(1+e^{-\mathbf{w}\mathbf{x}^{(n)}})^2} = x_k^{(n)} y^{(n)}(1-y^{(n)}),$$

putting it together

$$\frac{\delta G(\mathbf{w})}{\delta w_k} = -\sum_n \left[t^{(n)} - y^{(n)}\right] x_k^{(n)}.$$

Taking all derivative together we construct **the gradient vector**[K],

$$\mathbf{g} = \frac{\delta G(\mathbf{w})}{\delta \mathbf{w}} = -\sum_n \left[t^{(n)} - y^{(n)}\right] \mathbf{x}^{(n)}.$$

The vector

$$e^{(n)} = t^{(n)} - y^{(n)}$$

is referred to as the **error**.

Gradient descent optimization

$$\boxed{\mathbf{w}^{it+1} = \mathbf{w}^{it} - \eta\,\mathbf{g}(\mathbf{w}^{it})}$$

Batch gradient-descent learning algorithm

Update weights using all training examples

grandient descent algorithm

$$\mathbf{w}_1 = \mathbf{w}_0 + \eta \sum_{n=1}^{N} \left[t^{(n)} - y^{(n)}(\mathbf{w}_0) \right] \mathbf{x}^{(n)}$$

$$\mathbf{w}_2 = \mathbf{w}_1 + \eta \sum_{n=1}^{N} \left[t^{(n)} - y^{(n)}(\mathbf{w}_1) \right] \mathbf{x}^{(n)}$$

$$\ldots$$

On-line gradient-descent learning algorithm

Update weights using one random example at the time

stochastic grandient descent algorithm

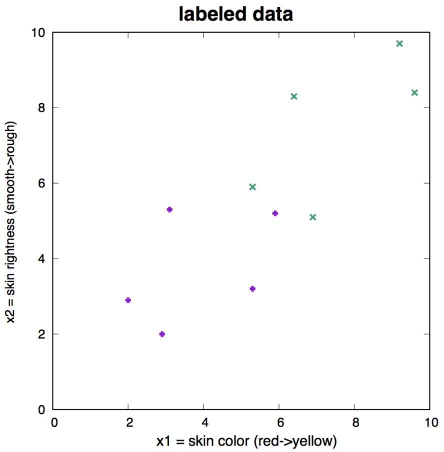
$$\mathbf{w}_1 = \mathbf{w}_0 + \eta \left[t^{(m)} - y^{(m)}(\mathbf{w}_0) \right] \mathbf{x}^{(m)} \quad m \in [1, N]$$

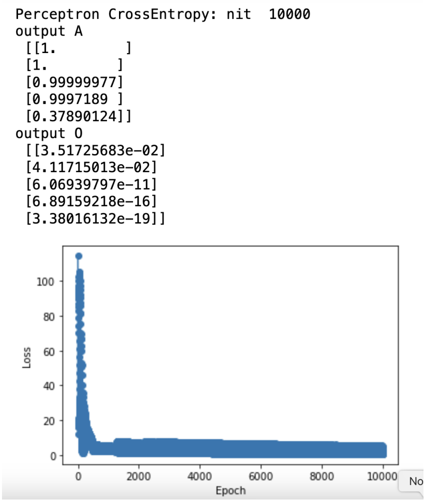
$$\mathbf{w}_2 = \mathbf{w}_1 + \eta \left[t^{(m')} - y^{(m')}(\mathbf{w}_1) \right] \mathbf{x}^{(m')} \quad m' \in [1, N]$$

$$\ldots$$

How well does the learning algorithm do?

$N = 10,\ K = 2 + 1$

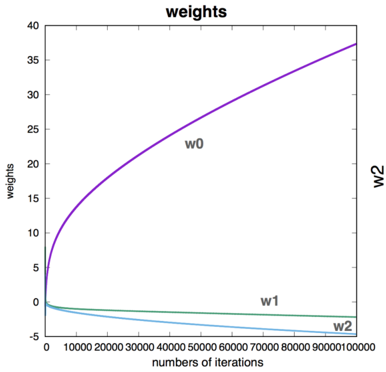


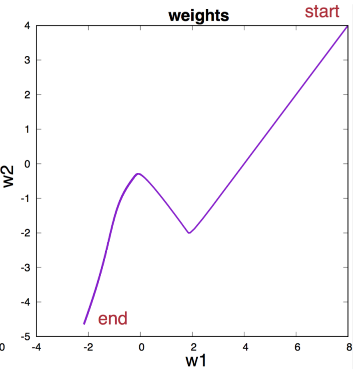


```
Perceptron CrossEntropy: nit  10000
output A
 [[1.        ]
 [1.        ]
 [0.99999977]
 [0.9997189 ]
 [0.37890124]]
output O
 [[3.51725683e-02]
 [4.11715013e-02]
 [6.06939797e-11]
 [6.89159218e-16]
 [3.38016132e-19]]
```

How well does the learning algorithm do?

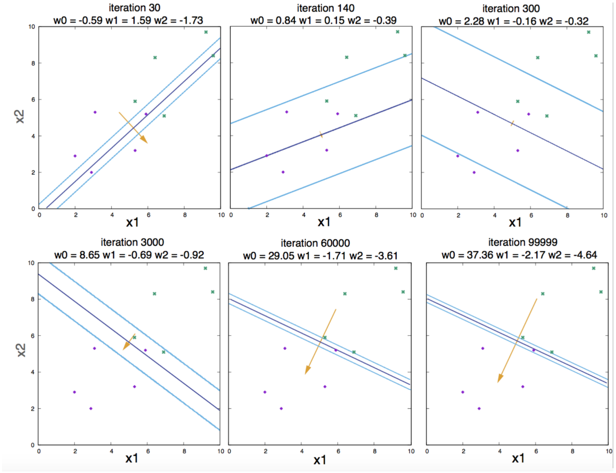
$N = 10$, $K = 2 + 1$





How well does the learning algorithm do?

too well?



overfitting

Regularization: beyond descent on the error function

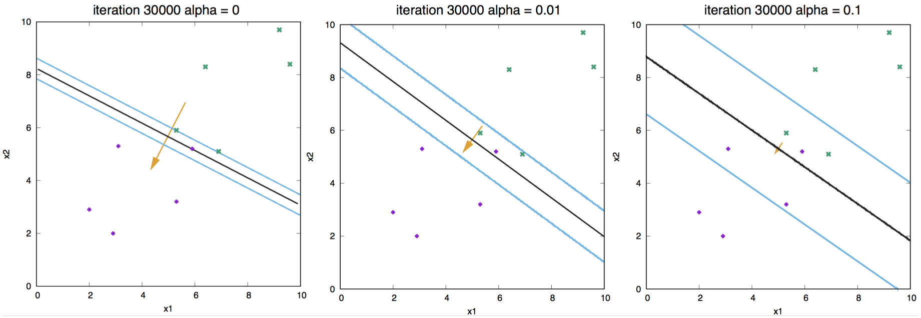
$$Loss(\mathbf{w}) = G(\mathbf{w}, \{\mathbf{x}^{(n)}\}_1^N) + \alpha R(\mathbf{w})$$

where
$R(\mathbf{w}) = \frac{1}{2} \sum_i w_i^2.$
only depends on the weights.

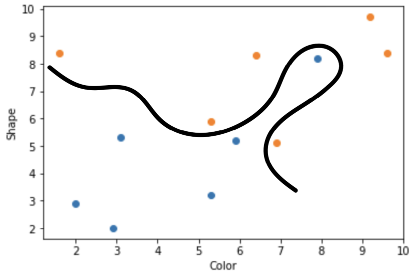
The weight update rule in the presence of this regularization becomes
$\mathbf{w}' = (1 - \alpha\eta)\,\mathbf{w} + \eta \sum_n \left(t^{(n)} - y_0^{(n)} \right) \mathbf{x}^{(\mathbf{n})},$

Regularization: beyond descent on the error function

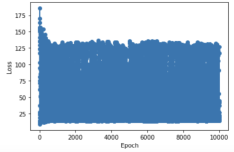
$$\boxed{Loss = G(\mathbf{w}, \mathbf{x}) + \alpha R(\mathbf{w}); \quad \alpha = \text{weight decay regularizer}}$$



What does a perceptron cannot do?



```
Perceptron CrossEntropy: nit  10000
output A
 [[1.        ]
 [1.        ]
 [1.        ]
 [0.99999778]
 [0.99999971]
 [0.99825799]]
output O
 [[0.9999958 ]
 [0.99999989]
 [0.9939444 ]
 [0.99880651]
 [0.89815603]]
```

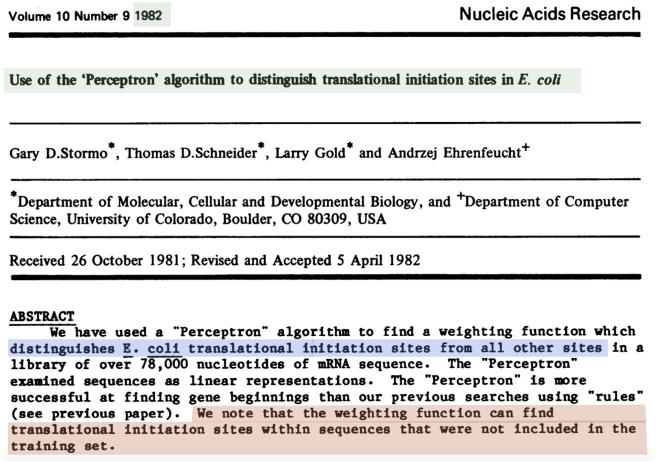


Frank Rosenblatt: The perceptron: a probabilistic model for information storage and organization in the brain, 1958.

The Ice-Age of machine learning.

Misky & Papert 1968 $\longrightarrow$ Multilayer perceptrons 1980sih

Translation Initiation sites with a perceptron



Volume 10 Number 9 1982 Nucleic Acids Research

Use of the 'Perceptron' algorithm to distinguish translational initiation sites in *E. coli*

Gary D.Stormo[*], Thomas D.Schneider[*], Larry Gold[*] and Andrzej Ehrenfeucht[+]

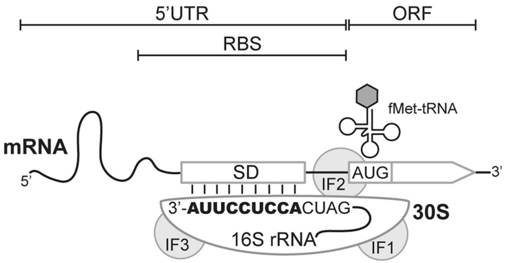
[*]Department of Molecular, Cellular and Developmental Biology, and [+]Department of Computer Science, University of Colorado, Boulder, CO 80309, USA

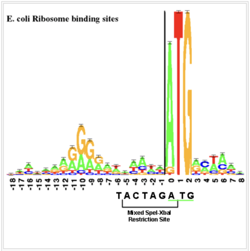
Received 26 October 1981; Revised and Accepted 5 April 1982

ABSTRACT

We have used a "Perceptron" algorithm to find a weighting function which distinguishes E. coli translational initiation sites from all other sites in a library of over 78,000 nucleotides of mRNA sequence. The "Perceptron" examined sequences as linear representations. The "Perceptron" is more successful at finding gene beginnings than our previous searches using "rules" (see previous paper). We note that the weighting function can find translational initiation sites within sequences that were not included in the training set.

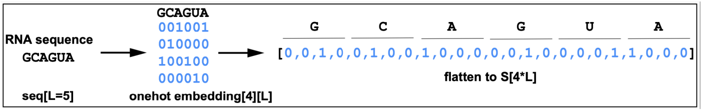
Translation Initiation sites in bacteria



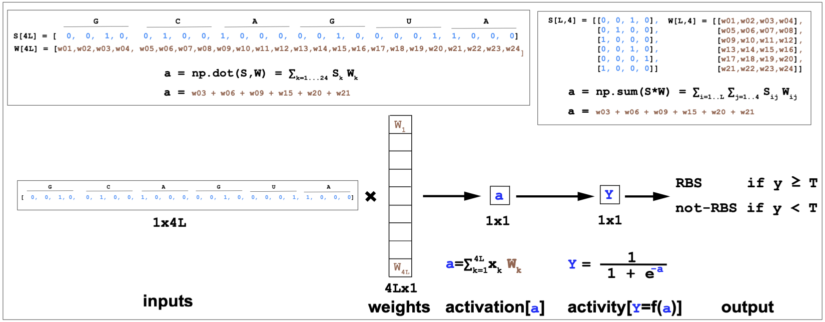


The RBS sequence logo, from Tom Schneider, [http://www-lmmb.ncifcrf.gov/%7Etoms/sequencelogo.html "A Gallery of Sequence Logos"]</small>

Translation Initiation sites with a perceptron



Translation Initiation sites with a perceptron



```
             G            C            A            G            U            A
S[4L] = [ 0, 0, 1, 0,  0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]
W[4L] = [w01,w02,w03,w04, w05,w06,w07,w08,w09,w10,w11,w12,w13,w14,w15,w16,w17,w18,w19,w20,w21,w22,w23,w24]
```

$$a = \texttt{np.dot(S,W)} = \sum_{k=1...24} S_k W_k$$

$$a = w03 + w06 + w09 + w15 + w20 + w21$$

```
S[L,4] = [[0, 0, 1, 0],  W[L,4] = [[w01,w02,w03,w04],
          [0, 1, 0, 0],             [w05,w06,w07,w08],
          [1, 0, 0, 0],             [w09,w10,w11,w12],
          [0, 0, 1, 0],             [w13,w14,w15,w16],
          [0, 0, 0, 1],             [w17,w18,w19,w20],
          [1, 0, 0, 0]]             [w21,w22,w23,w24]]
```

$$a = \texttt{np.sum(S*W)} = \sum_{i=1..L} \sum_{j=1..4} S_{ij} W_{ij}$$

$$a = w03 + w06 + w09 + w15 + w20 + w21$$

```
   G           C           A           G           U           A
[ 0, 0, 1, 0,  0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]
```

$$a=\sum_{k=1}^{4L} x_k W_k \qquad Y = \frac{1}{1 + e^{-a}}$$

Gradient descent Learning

- Calculate gradient
  $$g = -\sum_n \begin{cases} (1 - y^{(n)})S^{(n)}, & \text{if} \quad S^{(n)} \quad is \quad + \\ (-1 - y^{(n)})S^{(n)}, & \text{if} \quad S^{(n)} \quad is \quad - \end{cases}$$
- Update weights
  $W \leftarrow W + \eta g$

Stormo's training

It defines a threshold T (set to T=0), and does the following updates

- if S^+ and $S^+ \cdot W < T$, update: $W \leftarrow W + S^+$
- if S^- and $S^- \cdot W > T$, update: $W \leftarrow W - S^-$
- otherwise W remain unchanged

Translation Initiation sites with a perceptron

The authors mention two advantages of their method over conventional consensus sequence approach

- Each site S is evaluated quantitatively by $S * W$.

Translation Initiation sites with a perceptron

The authors mention two advantages of their method over conventional consensus sequence approach

- Each site S is evaluated quantitatively by $S * W$.
- Nothing is specified about the sequences except for their inclusion in a class.
  The algorithm finds the weights that best provide the classification.
  This advantage has the side effect that the weights are often hard to interpret.

Translation Initiation sites with a perceptron

Potential issues:

- Scores are not easily comparable

Translation Initiation sites with a perceptron

Potential issues:

- Scores are not easily comparable
- Weights may be uninterpretable

Translation Initiation sites with a perceptron

Potential issues:

- Scores are not easily comparable
- Weights may be uninterpretable
- All example inputs have to have the same number of features. problematic if each residue in input sequence is a one-ot feature