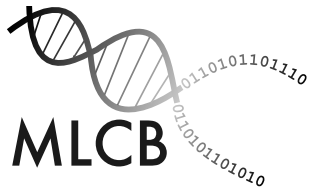




# Neural Networks: A Very Brief Tutorial

Chloé-Agathe Azencott

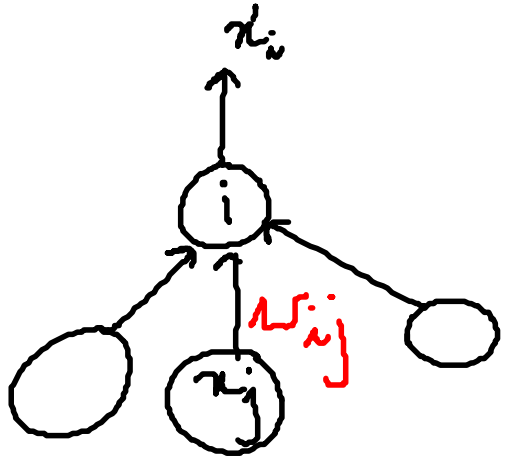
**Machine Learning & Computational Biology**
MPIs for Developmental Biology & for Intelligent Systems
Tübingen (Germany)

cazencott@tue.mpg.de

October 30, 2013

# 1950s – 1970s

# Units




$$x_i' = \varphi(x_1, \dots, x_j, \dots x_m)$$

transfer function

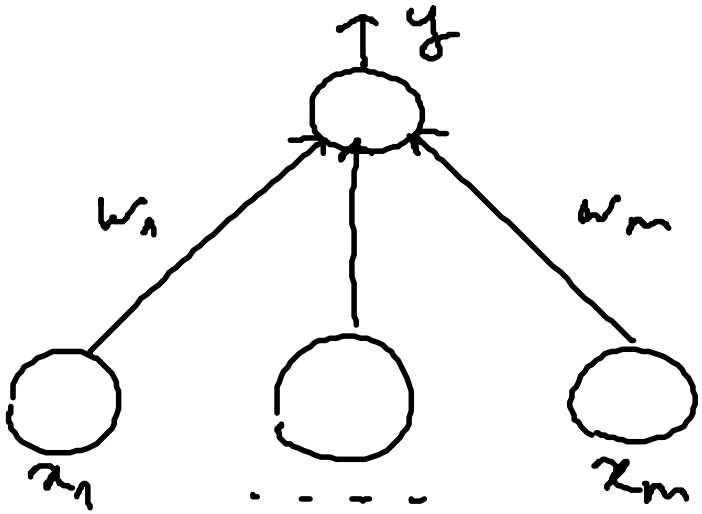
$$= f\left(\sum_{j \in Pr(i)} w_{ij}\, x_j + w_i\right)$$

output; activation; bias

- Linear
- Squashing $x \mapsto \begin{cases} x & \text{if } |x| < \Theta \\ 0 & \text{otherwise} \end{cases}$
- Threshold $x \mapsto \begin{cases} 1 & \text{if } |x| > \Theta \\ 0 & \text{ow} \end{cases}$
- Logistic $1/(1+e^{-x})$
- Sigmoid tanh, etc.
- Softmax (normalized exponential) $\dfrac{e^{-x_i}}{\sum_k e^{-x_k}}$

# Perceptron

Rosenblatt (1958)
Minsky & Papert (1969)




output

$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i + w_0 > 0 \\ 0 & \text{otw} \end{cases}$$

input

Training rule

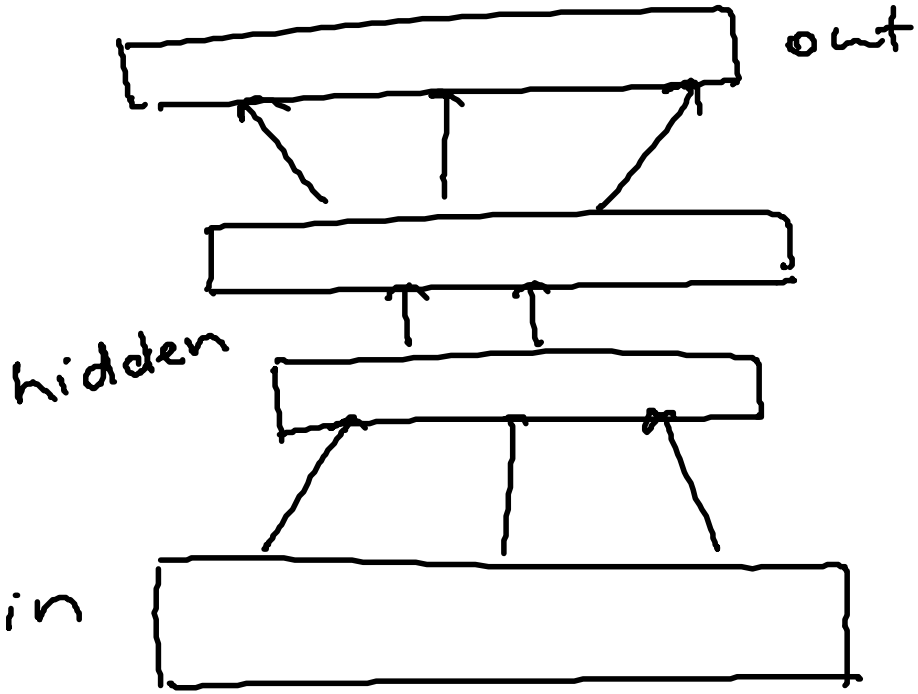
learning rate

$$w_i \leftarrow w_i + \eta\,(\text{true} - y)\,x_i$$

For each training sample,
for each feature

# 1980s – early 1990s

# Multi-Layer Perceptron



# Multi-Layer Perceptron

**Universal approximation**: Any continuous function on a compact subset of $\mathbb{R}^n$ can be approximated to any arbitrary degree of precision by a feed-forward multi-layer perceptron with a single hidden layer containing a finite number of neurons.

Cybenko (1989), Hornik (1991)

# Backpropagation

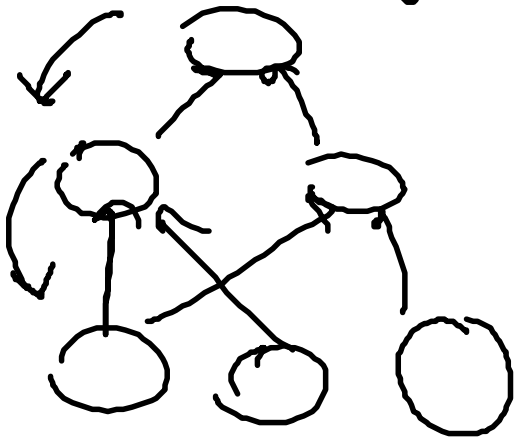
$y_i = f\left(\underbrace{\sum_{k \in rcv(i)} y_k w_{ik}}_{a_i}\right)$

Bryson, Denham & Dreyfus (1969)

Werbos, Rumelhart, Williams & Hinton (1974)

Backwards propagation of errors

$E = y - target$



$\frac{\partial E}{\partial w_{ij}}$

$w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij}$

$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial w_{ij}}$

$\frac{\partial y_i}{\partial w_{ij}} = f'(a_i) \frac{\partial a_i}{\partial w_{ij}}$

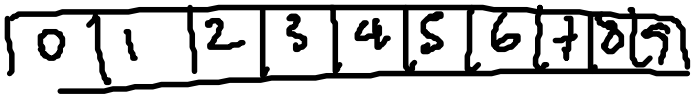
$\frac{\partial a_i}{\partial w_{ij}} = y_j$

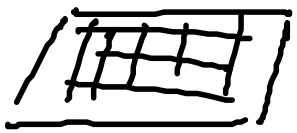
$\frac{\partial E}{\partial y_i} = f'(a_i) y_j$

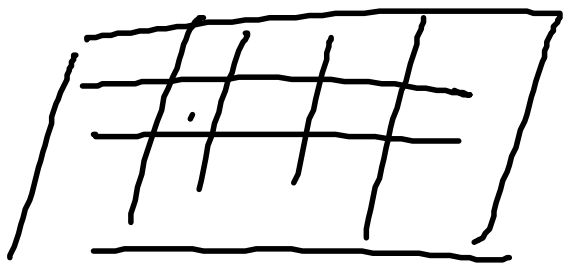
$\frac{\partial E}{\partial y_i} = \sum_{k \in rcv(i)} \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial y_i} w_{ki}$

# Handwritten digits recognition

Le Cun et al. (1990)





Shared weights

4 hidden layers

$H_2$, $H_4$: local averaging

# Recurrent Neural Networks

As opposed to **feed-forward** architectures.

Include directed cycles. feedback loops

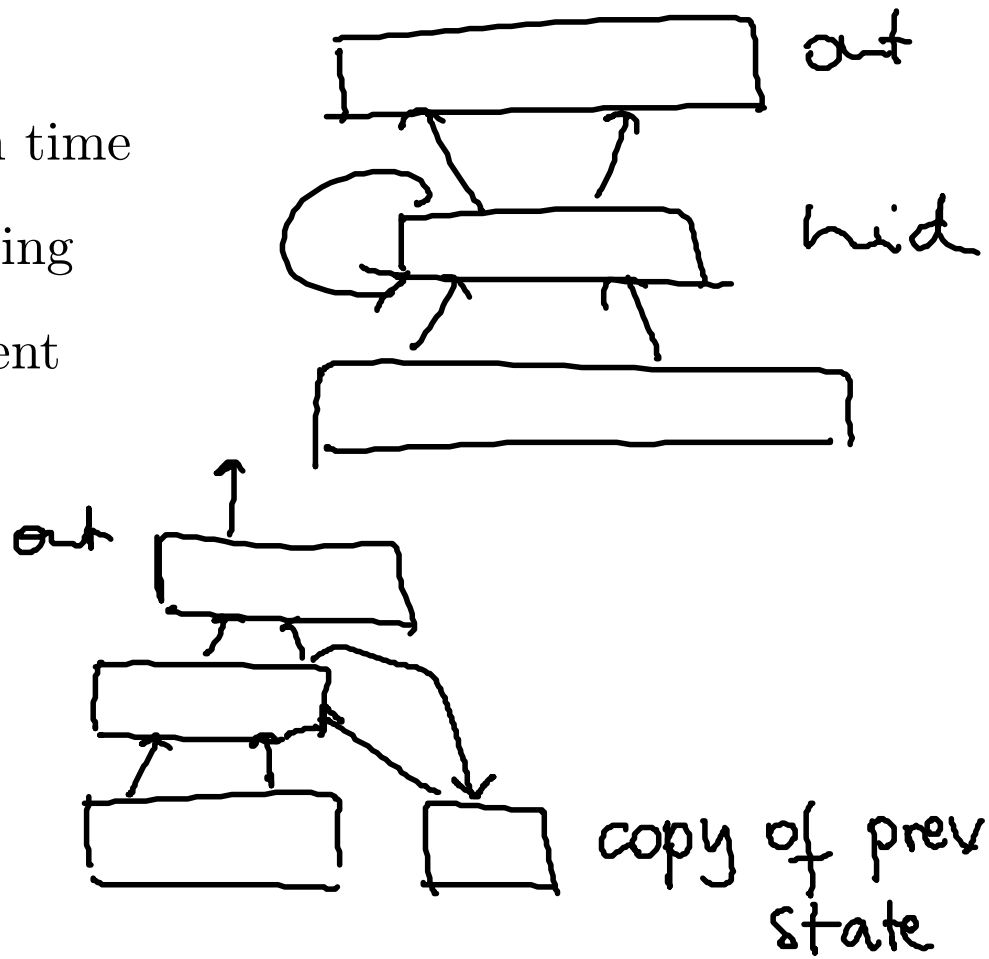
Can encode dynamic temporal behaviour.

Training:

- Backpropagation through time
- Real-time recurrent learning
- High-order gradient descent

Examples:

- Elman networks
- Jordan networks
- Hopfield networks
- Liquid state machines



# Neural Networks (MLPs) in Practice

# Packages

## Matlab

- Neural Network Toolbox
- Deep Learn Toolbox, Deep Belief Networks, and others at `http://deeplearning.net/software_links/`

## Python

- `bpnn.py`
- `fnnet`
- (C bindings) `FANN`
- `PyNN`
- `PyAnn`
- `nnutils` based on `Monte`

## R

- `neuralnet`
- `nnet`

# Neural Network Magic

## Architecture

Start with one hidden layer.

Stop adding layers when you overfit.

Never use more ~~nodes~~ weights than training samples.

## Learning algorithm

Use batch learning.

Use Newton/quasi-Newton methods for small numbers of weights. Prefer Levenberg-Marquardt.

Use conjugate-gradient descent for large numbers of weights.

# Neural Network Magic

## Preconditionning

An **ill-conditionned** network cannot learn.

The best learning rate is typically different for each weight.

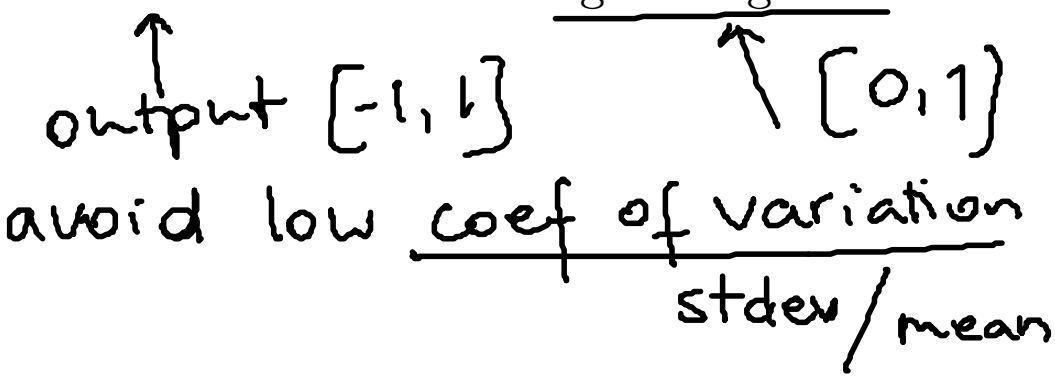
Solutions:

Standardize inputs and targets.

Initialize the weights carefully.

Set local learning rates.

Use *tanh* rather than a logistic sigmoid for hidden layers.



# Neural Network Magic

## Standardization

* Remove outliers

### Features

- Mean 0, standard deviation 1.
- Midrange 0, range 2.
- Orthonormalize. (SVD, PCs ...)

### Targets

- To equalize target importance: mean 0, standard deviation 1.
- To force targets into the range of the output activation function: midrange 0, range 2
  use lower/upper bounds rather than min/max.

### Samples

Standardize to a Euclidean distance of 1 for Kohonen networks or ratio-level data.

# Neural Network Magic

## Escaping saturation

Large weights ⇓ Saturation

### Weight initialization

$$w_{ij} \in [-r, r] \qquad r = \frac{1}{\sqrt{|\mathrm{fan}(i)|}}$$

### Weight decay

regularization

$E \longrightarrow E + \text{weight decay}$

$$\sum w_{ij}^2$$

weight elimination

$$\sum_i \frac{w_i^2}{w_i^2 + 6}$$

# Neural Network Magic

## Escaping local minima

**Online learning** or mini batch

**Momentum**

$$w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij} + m \Delta^{t-1} w_{ij}$$

**Learning rate adaptation**

$$w_{ij} \leftarrow w_{ij} + \mu_{ij} \Delta w_{ij}$$

1. $\mu$ while the gradient keeps pointing in the same direction
   $\mu_{ij} \leftarrow \mu_{ij} + q \, \Delta^{(t)} w_{ij} \, \Delta^{(t-1)} w_{ij}$
2. Prevent $\mu_{ij} < 0$ => apply to $\log(\mu_{ij})$ instead
3. $\mu_{ij} \leftarrow \mu_{ij} \times \max\left(0.5, 1 + q \, \Delta^{(t)} w_{ij} \, \Delta^{(t-1)} w_{ij}\right)$

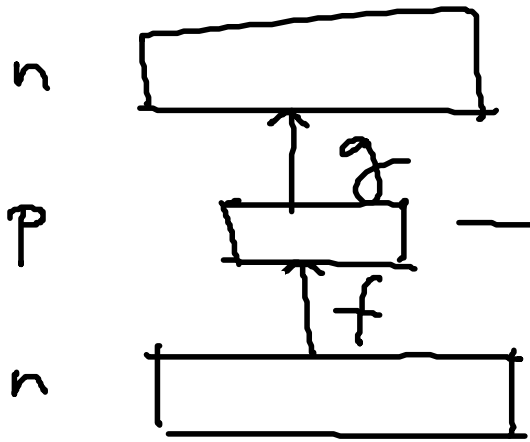
approximate to avoid computing exp

avoid too small $\mu$ etc...

**since 2006**

# Autoencoders

Rumelhart, Hinton & Williams (1986)
Dimensionality reduction with neural networks.




goal: output matches input

compact representation of input

$$\min_{f,g} \sum_x \Delta(f \circ g(x), x)$$

Hinton: sigmoidal neurons + backprop

$$\Delta(y, x) = \|y - x\|_2^2$$

Baldi: Linear autoencoders

$$\min_{A,B} \sum_x \|ABx - x\|_2^2$$

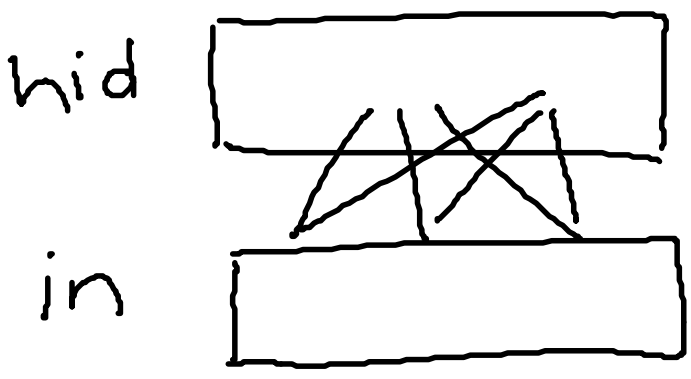
$A, B$ ↑ matrices

# Restricted Boltzmann Machines

Hinton & Sejnowsky (1985) BM

Smolensky (1986) RBM

Unsupervised learning with stochastic neural networks.




activation : probabilistic

bipartite graph: all inputs connected to all hidden

activation energy $E_i = \sum_{j \in ch(i)} w_{ij} x_j$

proba of activation : $p_i = \sigma(E_i)$ ← logistic

# Restricted Boltzmann Machines

**Contrastive divergence** – Hinton (2002)

Given data $V$ (visible units)

$$\underset{W}{\operatorname{argmax}} \sum_{v \in V} \log P(v)$$

$$P(v) \propto e^{-E(v,h)}$$

hidden

$$-E(v,h) = h^T W v - \text{bias}$$

Learning with CD

gradient descent + Gibbs sampling

* compute $P(\text{hidden})$ ← distribution
* sample $h$ from $P(\text{hidden})$
* sample $v'$ (reconstr. of $v$), sample $h'$

$$w_{ij} \leftarrow w_{ij} + \eta \left( v h^T - v' h'^T \right)$$

# Deep Learning

`http://deeplearning.net`

Many (hidden) factors ⇒ many (hidden) layers

2006: "Deep Breakthrough"

Hinton, Osindero & Teh
Bengio, Lamblin, Popovici & Larochelle
Ranzato, Poultney, Chopra & LeCun

Dropout (Hinton, 2012)
Maxout (Goodfellow et al., 2013)

Deep Belief Networks (DBNs)

Convolutional Neural Networks (CNNs)

Stacked auto-encoders

# Deep Learning in Bioinformatics

Predicting protein structure (Di Lena, Nagata & Baldi, 2012).

Predicting aqueous solubility (Lusci, Pollastri & Baldi, 2012).

Predicting protein properties (Qi, Oja, Weston & Noble, 2012).

Predicting drug-target interactions (Wang & Zheng, 2013).

Automating phenotyping (Lasko, Denny & Levy, 2013).

# Resources

Usenet newsgroup `comp.ai.neural-nets` FAQ
`http://www.faqs.org/faqs/ai-faq/neural-nets/`

Introduction to Neural Networks
N. Schraudolph and F. Cummins
`http://www.csi.ucd.ie/staff/fcummins/NNcourse/intro.html`

Neural Networks
C. Stergiou and D. Siganos
`http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html`

# References I


Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle.
Greedy layer-wise training of deep networks.
In NIPS, 2006.

C. M. Bishop.
Neural Networks for Pattern Recognition.
Oxford University Press, November 1995.

A. E. Bryson and W. F. Denham.
Optimal programming problems with inequality constraints. II – solution by steepest-ascent.
AIAA Journal, 2(1):25–34, 1964.

G. Cybenko.
Approximation by superpositions of a sigmoidal function.
Mathematics of Control, Signals and Systems, 2(4):303–314, December 1989.

P. Di Lena, K. Nagata, and P. Baldi.
Deep architectures for protein contact map prediction.
Bioinformatics, 28(19):2449–2457, 2012.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio.
Maxout networks.
arXiv e-print 1302.4389, 2013.
JMLR WCP 28 (3): 1319-1327, 2013.

# References II


G. E. Hinton.
Training products of experts by minimizing contrastive divergence.
Neural Computation, 14(8):1771–1800, 2002.

G. E. Hinton.
A practical guide to training restricted Boltzmann machines.
Technical Report UTML TR 2010-003, U. Toronto, 2010.

G. E. Hinton, S. Osindero, and Y.-W. Teh.
A fast learning algorithm for deep belief nets.
Neural Computation, 18(7):1527–1554, 2006.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov.
Improving neural networks by preventing co-adaptation of feature detectors.
arXiv e-print 1207.0580, 2012.

K. Hornik, M. Stinchcombe, and H. White.
Multilayer feedforward networks are universal approximators.
Neural Netw., 2(5):359âĂŞ366, 1989.

T. A. Lasko, J. C. Denny, and M. A. Levy.
Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data.
PLoS ONE, 8(6), 2013.

Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson.
Handwritten digit recognition with a back-propagation network.
In NIPS 2, pages 396–404, San Francisco, CA, USA, 1990.

# References III


A. Lusci, G. Pollastri, and P. Baldi.

Deep architectures and deep learning in chemoinformatics: The prediction of aqueous solubility for drug-like molecules.

Journal of Chemical Information and Modeling, 53(7):1563–1575, July 2013.

M. Minsky and S. Papert.

Perceptrons: An Introduction to Computational Geometry.

The MIT Press, 1969.

G. B. Orr and K.-R. Müller, editors.

Neural Networks: Tricks of the Trade.

Number 1524 in LNCS. Springer Berlin Heidelberg, January 1998.

Y. Qi, M. Oja, J. Weston, and W. S. Noble.

A unified multitask architecture for predicting local protein properties.

PLoS ONE, 7(3), 2012.

M. Ranzato, C. Poultney, S. Chopra, and Y. Le Cun.

Efficient learning of sparse representations with an energy-based model.

In NIPS, 2006.

F. Rosenblatt.

The perceptron: A probabilistic model for information storage and organization in the brain.

Psychological Review, 65(6):386–408, 1958.

# References IV


D. E. Rumelhart, G. E. Hinton, and R. J. Williams.

Learning internal representations by error propagation.
In David E. Rumelhart and James L. McClelland, editors, Parallel Distributed Processing, Vol. 1: Foundations, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

P. Smolensky.

Information processing in dynamical systems: Foundation of harmony theory.
In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing, Vol. 1: Foundations, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

D. F. Specht.

A general regression neural network.
Neural Networks, IEEE Transactions on, 2(6):568–576, 1991.

Y. Wang and J. Zeng.

Predicting drug-target interactions using restricted Boltzmann machines.
Bioinformatics, 29(13):i126–i134, 2013.

P. J. Werbos.

Backpropagation through time: what it does and how to do it.
Proceedings of the IEEE, 78(10):1550–1560, 1990.