Machine Learning and Data Mining -
Perceptron Neural Networks

Nuno Cavalheiro Marques (nmm@di.fct.unl.pt)

Spring Semester 2010/2011 – MSc in Computer Science
Model logic with neurons (McCulloch and Pitts, 1943).

- $x_0$ usually is set to 1, for all samples. In that case $w_0$ is called neuron bias.
Most Relevant Work in Neural Networks

- McCulloch and Pitts in 1943
  - Introduces Neural Networks as computing devices: A Logical Calculus of the Ideas Immanent in Nervous Activity.
- Hebb, 1949 The basic model of network self-organization.
- Rosenbatt, 1958
  - Learning with a teacher and the Perceptron
  - Minsky and Papert (1969), show that a perceptron can not learn a simple XOR
  - multilayer (feedforward) perceptron (or MLP), and backpropagation of the error: Rumelhart, Hinton, and Williams (1986)
Backpropagation Basic Algorithm

- Backpropagation — learning in networks of perceptrons.
- BP (and variants) are the best way to train MLPs.
- Generalized Delta Rule:

\[ \Delta \vec{w} = \eta \nabla (E[\vec{w}]) \]
Why Several Neural Network models?

Models, Paradigms and Methods

- The exact description of a biological neuron is still unknown.
  - In Neuro-physiology each cell is a complex dynamical system controlled by electric fields and chemical neurotransmitters.

- (Mathematical) **Model**
  - Representation of an existing system which presents (some) knowledge in usable form.
  - Finite set of variables and interactions.

- (Eg. Scientific) **Method**
  - A way how to solve a given problem.
  - Eg. **Bodies of techniques** for investigating phenomena, acquiring, correcting and integrating knowledge.
  - Collection of data through observation and experimentation, and the formulation and testing of hypotheses.
Models, Paradigms and Methods for ANN

- Paradigms
  - Model or case example of a general theory
- Artificial Neural Networks
  - Paradigms for some biological neuronal behaviour.
  - Available models are incorrect or incomplete regarding the biological neuron.
  - What is the goal?
    - Understand the biological brain?
    - Get better machine learning methods?
    - Solve specific problems?
- Question for each new model: what is its added value? (Can we solve it with a previous model?)
- Eg. Self Organizing Models (SOM): Unsupervised model that can be used for clustering and visualizing data.
ANN models

- JASTAP
- Recurrent SOM
- ART
- PNN
- RBF
- ...

Relevant problems for new models
  - How to train/learn?
  - How to understand and represent knowledge?
  - What is this model contribution?
Spiking models

- Implement more neuro-realistic models, taking time into account.

- Input is a time series
  - Some works use spike input trains.
Issues on Any Neural Networks

- How can NN learn in more complex models?
  - Genetic algorithms can learn weights and architecture.
  - Generalised BP for spiking neurons but model harder to understand.
  - What are the advantages of more complex models?
  - Even in simple MLPs: only sub-optimal learning (e.g. Vapnik).

- Simulate or understand biology?
  - Spiking models are good models of biological neurons?
  - In SOM: hundreds of simple units similar to biological neural maps.
  - Hans Moravec: why use silicon to duplicate biology.
  - Hölldobler: MLP implement any logical function kernel.
Backpropagation Basic Algorithm

- Backpropagation — learning in networks of perceptrons.
- BP (and variants) are the best way to train MLPs.
- Generalized Delta Rule:

\[ \Delta \vec{w} = \eta \nabla (E[\vec{w}]) \]
Consider simpler *linear unit*, where:

\[ o = w_0 + w_1 x_1 + \cdots + w_n x_n \]

Let’s learn \( w_i \)’s that minimize the squared error of simpler *linear unit*

\[ E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \]

Where \( D \) is set of training examples.
Gradient Descent: concepts (from [TM97]) II

- **Gradient:**
  \[ \nabla E[\vec{w}] \equiv \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right] \]

- **Training rule:**
  \[ \Delta \vec{w} = -\eta \nabla E[\vec{w}] \]

- i.e.,
  \[ \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \]
Gradient Descent: concepts (from [TM97]) III

\[
\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\
= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\
\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)(-x_{i,d})
\]
Gradient Descent Algorithm for Perceptron (from [TM97])

Gradient-Descent($training\_examples, \eta$)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where $\vec{x}$ is the vector of input values, and $t$ is the target output value. $\eta$ is the learning rate (e.g., .05).

**Note:** [TM97] is assuming the linear unit

- Initialize each $w_i$ to some small random value
- Until the termination condition is met, Do
  - Initialize each $\Delta w_i$ to zero.
  - For each $\langle \vec{x}, t \rangle$ in $training\_examples$, Do
    - Input the instance $\vec{x}$ to the unit and compute the output $o$
Gradient Descent Algorithm for Perceptron (from [TM97])

- For each linear unit weight $w_i$, Do
  \[ \Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i \]

- For each linear unit weight $w_i$, Do
  \[ w_i \leftarrow w_i + \Delta w_i \]
Sigmoid versus step function

- Step function $x > 0$ is non differentiable.
- Sigmoid functions are similar but differentiable.
- Function:
  \[
  \sigma(x) = \frac{1}{1 + e^{-x}}
  \]
  has the nice property:
  \[
  \frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))
  \]
Error Gradient for a Sigmoid Unit (from [TM97])

\[
\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\
= \frac{1}{2} \sum_{d} \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
= \frac{1}{2} \sum_{d} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
= \sum_{d} (t_d - o_d) \left( - \frac{\partial o_d}{\partial w_i} \right) \\
= - \sum_{d} (t_d - o_d) \frac{\partial o_d}{\partial net_d} \frac{\partial net_d}{\partial w_i}
\]
Error Gradient for a Sigmoid Unit (from [TM97]) II

But we know:

\[
\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(\text{net}_d)}{\partial \text{net}_d} = o_d(1 - o_d)
\]

\[
\frac{\partial \text{net}_d}{\partial w_i} = \frac{\partial (\vec{w} \cdot \vec{x}_d)}{\partial w_i} = x_{i,d}
\]

So:

\[
\frac{\partial E}{\partial w_i} = - \sum_{d \in D} (t_d - o_d) o_d(1 - o_d)x_{i,d}
\]
Initialize all weights to small random numbers.
Until satisfied, Do

- For each training example, Do
  1. Input the training example to the network and compute the network outputs
  2. For each output unit $k$
    \[ \delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \]
  3. For each hidden unit $h$
    \[ \delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{h,k} \delta_k \]
MLP Backpropagation Algorithm (from [TM97]) II

4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$
More on Backpropagation (from [TM97])

- Gradient descent over entire *network* weight vector
- Easily generalized to directed graphs (Elman)
- Will find a local, not necessarily global error minimum
  - In practice, often works well (can run multiple times)
- Often include weight *momentum* $\alpha$

$$\Delta w_{i,j}(n) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(n - 1)$$

- Minimizes error over *training* examples
  - Will it generalize well to subsequent examples?
- Training can take thousands of iterations $\rightarrow$ slow!
- Using network after training is very fast
MLP-Decision Regions

Single Perceptron: linear discriminator unit

One hidden layer: any logic function
Overfitting in Neural Networks

- Too much units on hidden layers can prevent good generalization.
- Blackbox: Non-localized knowledge representation
Local Minima with Backpropagation Algorithm

- Local minima: usual problem with greedy methods.
- Wrong starting point can be problematic.
- Usual near zero random initialization can have problems.
- We should have a good starting point (based on our knowledge).
Using Visualization for Learning Analysis

- Discovering what the neural Network learned
  - Sensitivity Analysis
  - Rule Generation
- Standard Graphics
- Neural Network Graphic
- Hinton Diagram for weights
- Clustering and Segmentation Visualization
- Using Domain knowledge to view the output
Tom Mitchell’s Binary Encoding (in Mitchell, 1997)

Visualization of the Neural Network
Tom Mitchell’s Iterations for Binary Encoding (in Mitchell, 1997)
POS Tagger Network Graphic
POS Tagger weights
Both logic and neural networks are two founding areas in AI.

- Logic is applied to Computational Logic or Logic Programming
  - We can reason over structured objects.

- Neural Networks are particularly good for modeling sub-symbolic information
  - Can learn, adapt to new environments, graceful degradation.
  - Used in Industrial processes or controllers
  - Namely after training, specialized hardware and very fast response time.

Core method intends to integrate both approaches.
Issues on Neuro-Symbolic Integration I

- BP doesn’t work so well with strict logic rules ([BHM08]).
- In machine learning we need ways to guide learning.
- Propositional Fixation (only atomic propositions).
- Care should be taken while encoding symbols.
  - FOL and logical arguments can be taken into account.
  - Neural Networks can implement an Universal Turing machine.
  - We need recursion (difficult to understand).
  - In general, no clear semantics in integrated information.
- Only by understanding the starting model and the learning that is taking place we can advance beyond black-box approaches to machine learning.
Issues on Neuro-Symbolic Integration II

- Neural network usage in data mining requires good knowledge and explanation of data.
- Can the basic ideas be extended to other models and knowledge representations?
How can neural networks take advantage of express knowledge?

- Knowledge extraction achieved by machine learning in neural networks (backpropagation)
- After proper training, networks extract meaningful knowledge from data
  - Implicitly: classification and clustering models
  - Explicitly: Neuro-symbolic approach
    - New relevant connections in learned model (i.e. we can understand the neural network)
- Domain knowledge (rules) used to guide learning
  - Feed-Forward neural networks have many parameters
  - Rules bias learning to a nearby local-minima
  - Recursion (and neuro-computation) can be inserted with Elman Neural Networks.
Core method basic algorithm [HK,94]

- Implements the $T_p$ operator (next logical step)
  - If appropriate, recursion will lead to stable models.
- $m \leftarrow$ nb. of propositions in $P$
  - Input and output layer have size $m$, each unit has threshold 0.5 and represents one proposition.
  - For each clause $A \leftarrow L_1 \land L_2 \land \ldots \land L_k$ with $l \geq 0$ in $P$:
    1. Add a binary unit $c$ to hidden layer.
    2. For each literal $L_j$, $1 \leq j \leq k$, connect input unit for $L_j$ to $c$ with weight $\omega$ if $L_j$ is a positive literal, or $-\omega$ if $L_j$ is a negative literal.
    3. Set threshold $T_c$ of unit $c$ as $l \cdot \omega - 0.5$, where $l$ is the number of positive literals.
Discussion of $\omega$ while encoding rules

- For $\omega = 0$ network stops to improve at some point.
- For $\omega > 0$ network outperforms network $\omega = 0$.
- For $\omega = 5$ network did not learn very well.

Avoiding Local Minima

With errors on only a few samples, back-propagation can get into a local minima.
The MLP’s Sigmoid Function [M09]

Understanding Backpropagation Learning in the Core Method Networks

\[ \sigma(x) = \frac{1}{1 - e^{-x}} \]
Backpropagation and NeSy encoding I

- Parameter $\omega$ gives stability to rules while training.
- Recall gradient trains sigmoid units with:
  \[
  \frac{\partial E}{\partial w_i} = - \sum_d (t_d - o_d) \frac{\partial o_d}{\partial net_d} x_{i,d}. 
  \]  
  (1)
- bigger values of $\omega$ make the sigmoid approximate a stepwise function and make $\frac{\partial o_d}{\partial net_d}$ go near zero.
- good if certain on the rules.
- Not advisable for rules needing revision.
- There is a learning zone that depends on $\omega$ and $net$.
- Free units (with near zero weights) are the first to learn.
Basic Core Method for Probabilistic/Numeric Encodings

- How to handle other inputs (e.g. a probability of some observation).
- Problem with direct use of probabilities, in core method: $0.6 \geq 0.5$, but $0.6 \times 0.6 = 0.36 \leq 0.5$
- We need additional constraints, e.g.: $X > T_X$.
- But can the network learn the best value of $T_X$?
  - Use a sigmoid neuron having as input $X$ and bias $\omega \times T_X$:
    \[
    1 - \alpha < \frac{1}{1 + e^{-\omega \times (X - T_X)}} < \alpha.
    \]
  - Will be true iff $X > T_X$
Illustrative Example: Learning AND

1HL – Original Core Method

2HL – Extended Core Method

Figure: Initial core methods neural networks, before training and initial weight disturbance.
Differences in Backprop for Learning AND

**Figure:** Train error for learning an AND with four distinct encodings.
Elman Neural Network [E1990]

Core Method can not encode variable length patterns

Novel methods for encoding temporal rules to extend the power of neuro-symbolic integration.

- Elman neural networks allow rules with preconditions at non-deterministic time (e.g. important in NLP [FP01]
- (Analogical) Recursive connections allow a super-Turing power (e.g. Siegelmann, 1994/1999).
- Learning space: bias learning to a nearby local-minima and allow complex (but controled) feedback loops.
Elman Neural Networks

Neuronal Unit

Elman Neural Network
Learning in Elman Neural Networks

- Non-parametric statistical models learning from examples.
- Error Backpropagation algorithm iteratively adjusts weights.
- Find patterns in a sequences of values.
Main References


- Jeffrey L. Elman
  Finding Structure in Time
Other References I

- David E. Rumelhart and Geoffrey E. Hinton and Ronald Williams,

- Sebastian Bader, Steffen Hölldobler and Nuno Marques.
  Guiding Backprop by Inserting Rules

- Nuno Marques.
  An Extension of the Core Method for Continuous Values