Artificial Neural Networks of The Perceptron, Madaline, and Backpropagation Family

Bernard Widrow  Micheal A. Lehr

Presented by Xuanchong Li
1 Algorithms History

2 Fundamental Concepts
   - Adaptive linear combiner
   - Linear Classifier: Adaptive linear element (Adaline)
   - Non-linear Classifiers

3 Learning Algorithms
   - Principle and Rules
   - Algorithm Details
     - Error correction rules
     - Steepest Descent Rules

4 Invariance of Neural Network

5 Summary
1960: Least Mean Square (LMS) algorithm (Widrow and his student), Perceptron rule (Rosenblatt)

Mid 1960s: Madaline (Multiple adaptive linear elements) rule I (MRI) and application in speech, weather forecasting, pattern recognition (Widrow and his student)

1971: Backpropagation (Werbos). It was first ignored by community, then re-discovered in 1982 by Parker, finally became famous with work of Rumehart, Hinton, and Williams.

1987: Madaline Rule II (MRII) by Widrow and his student. The goal is for adapting multiple players network

1988: Madaline Rule III (MRIII) by David Andes. Widrow and his student found it is mathematically equivalent to backpropagation
1. Algorithms History

2. Fundamental Concepts
   - Adaptive linear combiner
   - Linear Classifier: Adaptive linear element (Adaline)
   - Non-linear Classifiers

3. Learning Algorithms
   - Principle and Rules
   - Algorithm Details
     - Error correction rules
     - Steepest Descent Rules

4. Invariance of Neural Network

5. Summary
Adaptive linear combiner

Input Pattern Vector

\[ X_k \]

Output

\[ S_k \]

Weight Vector

\[ W_k \]

Desired Response

\[ d_k \]

Error

\[ \epsilon_k \]

\[ X_k = [x_0, x_{1k}, x_{2k}, \ldots, x_{nk}]^T \]

\[ W_k = [w_{0k}, w_{1k}, w_{2k}, \ldots, w_{nk}]^T \]

\[ s_k = X_k^T W_k \]
Outline

1 Algorithms History

2 Fundamental Concepts
   - Adaptive linear combiner
   - Linear Classifier: Adaptive linear element (Adaline)
   - Non-linear Classifiers

3 Learning Algorithms
   - Principle and Rules
   - Algorithm Details
     - Error correction rules
     - Steepest Descent Rules

4 Invariance of Neural Network

5 Summary
Adaptive linear element (Adaline)

- Basic building block used in neural networks
- Adaptive threshold logic element: an adaptive linear combiner + hard-limiting quantizer

\[ y = \text{sgn}(x) \]
Outline

1. Algorithms History

2. Fundamental Concepts
   - Adaptive linear combiner
   - Linear Classifier: Adaptive linear element (Adaline)
   - Non-linear Classifiers

3. Learning Algorithms
   - Principle and Rules
   - Algorithm Details
     - Error correction rules
     - Steepest Descent Rules

4. Invariance of Neural Network

5. Summary
Polynomial Preprocessor

- Fixed preprocessing network + a single adaptive element
- The choice of preprocessing function matters a lot
Madaline I

- One of the earliest trainable layered neural networks.
- A layer of ADALINE + fix logic device (AND, OR, MAJ)
Feedforward Network

- All layers are adaptive
- Exp. a fully-connected three-layer feedforward adaptive network
Outline

1 Algorithms History

2 Fundamental Concepts
   - Adaptive linear combiner
   - Linear Classifier: Adaptive linear element (Adaline)
   - Non-linear Classifiers

3 Learning Algorithms
   - Principle and Rules
     - Algorithm Details
       - Error correction rules
       - Steepest Descent Rules

4 Invariance of Neural Network

5 Summary
The Minimal Disturbance Principle

- Minimal Disturbance Principle: Adapt to reduce the output error for the current training pattern, with minimal disturbance to responses already learned.
- It is behind every learning algorithm in the paper.
Two Classes of Rules

- Error correction rules: alter the weights of a network to correct a certain proportion of the error in the output response to the present input pattern.
- Steepest descent rules: alter the weights during each pattern presentation by gradient descent with the objective of reducing mean-square-error, average over all training patterns.
Outline

1. Algorithms History

2. Fundamental Concepts
   - Adaptive linear combiner
   - Linear Classifier: Adaptive linear element (Adaline)
   - Non-linear Classifiers

3. Learning Algorithms
   - Principle and Rules
   - Algorithm Details
     - Error correction rules
     - Steepest Descent Rules

4. Invariance of Neural Network

5. Summary
Linear Rules

α-LMS Algorithm

- Linear Rule: alter the weights of the adaptive threshold element with each pattern presentation to make an error correction which is proportional to the error itself.
- Follows Error Correction Rule
- Weight update rule:

\[ W_{k+1} = W_k + \alpha \frac{\epsilon_k X_k}{|X_k|^2} \]

- Error at k: \( \epsilon_k = d_k - W_k^T X_k \)
- Error Change: \( \Delta \epsilon_k = -\alpha \epsilon_k \)
- Weights are usually initialized as 0.
- Learning Rate: \( 0.1 < \alpha < 1 \), controls stability and convergence rate.
- Self-normalizing: choice of \( \alpha \) does not depend on the magnitude of the input signals.
Non-linear Rules
Perceptron

- A non-linear algorithm: weights change is collinear with the input pattern vector and the linear error.
- Follows Error Correction Rule
- Weight update rule: \( W_{k+1} = W_k + \alpha \frac{\tilde{e}_k}{2} X_k \)
\(\alpha\)-LMS V.S. Perceptron

- \(\alpha\) Value
  - \(\alpha\)-LMS: controls stability and speed of convergence
  - Perceptron rule: does not affect the stability of the perceptron algorithm, and it affects convergence time only if the initial weight vector is nonzero

- Binary or continuous response
  - \(\alpha\)-LMS: both binary and continuous response
  - Perceptron: only binary

- Linearly separable training patterns
  - \(\alpha\)-LMS: may fail to separate linearly separable set
  - Perceptron: separate any linearly separable set

- Nonlinearly separable training patterns
  - \(\alpha\)-LMS: does not lead to unreasonable weight solution
  - Perceptron: goes on forever is not linearly separable, and often does not yield a low-error solution. Usually end up with a small norm weight vector.
May’s Algorithm

- Non-linear error correction rule
- Introduce the ”deadzone” $\gamma$
- Separate any linearly separable set; For non-linearly separable set, Mays rule performs much better than Perceptron rule because a sufficiently large dead zone tends to cause the weight vector to adapt away from zero when any reasonably good solution exists.

\[
\begin{align*}
W_{k+1} &= \begin{cases} 
W_k + \alpha \varepsilon_k \frac{X_k}{2|X_k|^2} & \text{if } |s_k| \geq \gamma \\
W_k + \alpha d_k \frac{X_k}{|X_k|^2} & \text{if } |s_k| < \gamma
\end{cases}
\end{align*}
\]
Multi-element Networks
Madaline Rule I (MRI)
Multi-element Networks
Madaline Rule I (MRI)

- If an error happens, pick the ADALINE with smallest $|s_k|$ to adapt.
- Weights are initially set to small random values.
- Weight vector update: can be changed aggressively using absolute correction or can be adapted by the small increment determined by the $\alpha$-LMS algorithm.
- Principle: assign responsibility to the Adaline or Adalines that can most easily assume it.
- Pattern presentation sequence should be random.
### Madaline Rule I (MRI) example

The Madaline Rule I (MRI) example is illustrated with a set of 5 Adalines, each computing the dot product $S = W^T X$ and applying the sign function $\text{sgn}(s)$.

<table>
<thead>
<tr>
<th>Adaline</th>
<th>$W_1$</th>
<th>$W_2$</th>
<th>$W_3$</th>
<th>$W_4$</th>
<th>$W_5$</th>
<th>$X$</th>
<th>$S$</th>
<th>$\text{sgn}(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
<td>0.8</td>
<td>-0.1</td>
<td>-0.3</td>
<td>-0.4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

The sum $S$ is computed for each Adaline and then the final output $d$ is determined by the majority vote of the Adalines. In this example, the output $d = -1$. The diagram shows a blue circle labeled "MAJ" indicating the majority vote.
Madaline Rule I (MRI) example

\[ s = W^T X \]

\[
\begin{array}{cccc}
W_1 & \rightarrow & 0.2 & \rightarrow & 1 \\
W_2 & \rightarrow & 0.8 & \rightarrow & 1 \\
W'_3 & \rightarrow & 0.4 & \rightarrow & 1 \\
W_4 & \rightarrow & -0.3 & \rightarrow & -1 \\
W_5 & \rightarrow & -0.4 & \rightarrow & -1 \\
\end{array}
\]

5 Adalines

\[ d = 1 \]
Madaline Rule I (MRI) example

\[ S = W^T X \]

\[
\begin{align*}
W_1 & \rightarrow 0.2 & \rightarrow 1 \\
W_2 & \rightarrow -0.05 & \rightarrow -1 \\
W_3 & \rightarrow -0.1 & \rightarrow -1 \\
W_4 & \rightarrow -0.3 & \rightarrow -1 \\
W_5 & \rightarrow -0.4 & \rightarrow -1 \\
\end{align*}
\]

5 Adalines

\[ d = 1 \]
Madaline Rule I (MRI) example

\[ s = W^T X \]

\[ sgn(s) \]

5 Adalines
Typical two-layer Madaline II architecture
Madaline Rule II (MRII)

- Weights in both layers are adaptive
- Weights are initially set to small random values
- Random pattern presentation sequence
- Adapting the first-layer Adalines
  - Select the smallest linear output magnitude
  - Perform a “trial adaption” by adding a $\Delta s$ perturbation of suitable amplitude to invert its binary output
  - If the output error is reduced, remove $\Delta s$, and change the weight of the select Adaline using $\alpha$-LMS algorithm
  - Perturb and update other Adalines in the first layer with sufficiently small $s_k$ output
  - After exhausting possibilities with the first layer, move on to next layer and proceed in a like manner
  - Random select a new training pattern and repeat the procedure.
Objective of adaptation: reduce error averaged over the training set rather than reducing a given proportion of the error in each presentation of training data.

The most common error: mean-square-error (MSE)

\[ W_{k+1} = W_k + \mu(-\nabla_k) \]

\( \mu \): controls stability and convergence speed.

In practice, it works with on data at a time. It minimizes MSE approximately.
The steepest descent rule is linear if weight changes are proportional to the linear error.

Linear Combiner:
\[ \epsilon_k^2 = (d_k - X_k^T W_k)^2 = d_k^2 - 2d_k X_k^T W_k + W_k X_k X_k^T W_k \]

MSE:
\[ E[\epsilon_k^2] = E[d_k^2] - 2E[d_k X_k^T] W_k + W_k^T E[X_k X_k^T] W_k \]

\[ P^T \triangleq E[d_k X_k^T], \quad R \triangleq E[X_k X_k^T] \]

\[ \nabla_k = \frac{\partial E[\epsilon_k^2]}{\partial W_k} = -2P + 2RW_k \]

Set the gradient to zero: \( W^* = R^{-1}P \) (Wiener weight vector)

Need to compute \( R^{-1} \) and \( P \)?
Steepest Descent Rules

µ-LMS Algorithm

- Obtain accurate estimate of $W^*$ without computing $R^{-1}$ and $P$
- Use instantaneous gradient $\hat{\nabla}_k = \frac{\partial \epsilon_k^2}{\partial W_k}$. It is an unbiased estimate of the true gradient.
- Weight update: $W_{k+1} = W_k + \mu(-\hat{\nabla}_k) = W_k + 2\mu \epsilon_k X_k$
- $\mu$ controls stability and convergence speed. Training data should be in random order.
\( \alpha \)-LMS: \( W_{k+1} = W_k + \alpha \frac{\epsilon_k X_k}{|X_k|^2} \), \( \mu \)-LMS: \( W_{k+1} = W_k + 2\mu \epsilon_k X_k \)

- \( \alpha \)-LMS is self-normalization, with the parameter \( \alpha \) determining the fraction of the instantaneous error to be corrected with each adaption.
- \( \mu \)-LMS is constant-coefficient linear algorithm.
- In practice, \( \alpha \)-LMS usually converges faster than \( \mu \)-LMS.
- \( \mu \)-LMS converges to minimum MSE solution (Wiener solution), while \( \alpha \)-LMS converges to a biased solution.
- Normalized training set: \( \tilde{X}_k \triangleq \frac{X_k}{|X_k|} \), \( \tilde{d}_k \triangleq \frac{d_k}{|X_k|} \).
- \( \alpha \)-LMS achieves minimum MSE solution for normalized training set.
Sigmoid Adaline: extend the Adaline to include the use of a sigmoid in place of the signum.
- $y_k = sgm(s_k)$
- Use instantaneous gradient: $\hat{\nabla}_k = -2\tilde{\epsilon}_k sgm'(s_k)X_k$
- Update weights: $W_{k+1} = W_k + \mu(-\hat{\nabla}_k) = W_k + 2\tilde{\epsilon}_k sgm'(s_k)X_k$
Madaline Rule III (MRIIII) for Sigmoid Adaline

- Motivation: the backpropagation algorithm requires accurate implementation of sigmoid function hardware. Need another way to compute the gradient, which does not rely on the accurate function hardware.

- Adding a small perturbation signal $\Delta s$ to $s_k$, record the effect on $y_k$ and $\epsilon_k$.

\[
\hat{\nabla}_k = \frac{\partial \epsilon^2_k}{\partial W_k} = \frac{\partial \epsilon^2_k}{\partial s_k} \frac{\partial s_k}{\partial W_k} = \frac{\partial \epsilon^2_k}{\partial s_k} X_k \approx \left( \frac{\Delta \epsilon^2_k}{\Delta s} \right) X_k \approx 2\tilde{\epsilon}_k \left( \frac{\Delta \epsilon_k}{\Delta s} \right) X_k
\]

- $W_{k+1} = W_k - \mu \left( \frac{\Delta \epsilon^2_k}{\Delta s} \right) X_k$ or

Weights update: $W_{k+1} = W_k - 2\mu \tilde{\epsilon}_k \left( \frac{\Delta \epsilon_k}{\Delta s} \right) X_k$

- Backpropagation and MRIIII are mathematically equivalent if the perturbation $\Delta s$ is small. MRIIII is robust even with the analog implementation.
Intuition: propagate the error backward from the output layer to the first layer

For an input pattern vector $X$:

- Sweep forward through the system to get an output respond vector $Y$
- Compute the errors in each output
- Sweep the effects of the errors backward through the network to associate a “square error derivative” $\delta$ with each Adaline
- Compute a gradient from each $\delta$
- Update the weights of each Adaline based upon the corresponding gradient

More details are in the next talk.
MRIII for Networks

- Same idea as MRIII for sigmoid Madaline
- Measure the sum square output response error
  \[ \epsilon^2 = (d_1 - y_1)^2 + (d_2 - y_2)^2 = \epsilon_1^2 + \epsilon_2^2 \]
- \[ \frac{\Delta(\epsilon^2)}{\Delta s} = \frac{\Delta(\epsilon_1^2 + \epsilon_2^2)}{\Delta s} \approx \frac{\partial \epsilon^2}{\partial s} \]
- \[ \hat{\nabla}_k = \frac{\partial \epsilon_k^2}{\partial W_k} = \frac{\partial \epsilon_k^2}{\partial s_k} \frac{\partial s_k}{\partial W_k} = \frac{\partial \epsilon_k^2}{\partial s_k} X_k \approx \frac{\Delta \epsilon_k^2}{\Delta s} X_k \]
- \[ W_{k+1} = W_k - \mu \frac{\Delta \epsilon_k^2}{\Delta s} X_k \]
Invariance of Neural Network

- Neural network should be invariant to translation, rotation and scale change of the input pattern.
Invariance to up-down, left-right translation

Retina

All retinal signals go to all ADALINES

Slab output
Invariance to up-down, left-right translation
Invariance to up-down, left-right translation

- The roles of the various Adalines interchange.
- The “key” weights \((W_1)\) can be randomly chosen or manufactured.
- The rotation and scale invariance can be obtained in the same way.

\[
\begin{align*}
(W_1) & \quad T_{R1}(W_1) & \quad T_{R2}(W_1) & \quad T_{R3}(W_1) \\
T_{D1}(W_1) & \quad T_{R1}T_{D1}(W_1) & \quad T_{R2}T_{D1}(W_1) & \quad T_{R3}T_{D1}(W_1) \\
T_{D2}(W_1) & \quad T_{R1}T_{D2}(W_1) & \quad T_{R2}T_{D2}(W_1) & \quad T_{R3}T_{D2}(W_1) \\
T_{D3}(W_1) & \quad T_{R1}T_{D3}(W_1) & \quad T_{R2}T_{D3}(W_1) & \quad T_{R3}T_{D3}(W_1)
\end{align*}
\]
Summary

Artificial Neural Networks of The Perceptron, Madaline, and Backpropagation Family

Presented by Xuanchong Li 41 / 41