Preliminary Notes on Perceptrons

This is the study of feedforward networks as found at the "front end" of sensory systems. Roughly,

\[ y = f \left( \sum_{j=1}^{N_c} w_j x_j - b \right) \]

Consider case of step function, which is not so far from type 2 neurons or Hopf bifurcation.

McCulloch & Pitts considered mapping Boolean functions onto neurons in this way. Take the case of "AND".

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The perceptron has two calls, so
\[ w_1 x_1 + w_2 x_2 - b = y \] implies:

1. \( b < 0 \)
2. \( w_1 < b \)
3. \( w_2 < b \)
4. \( w_1 + w_2 > b \)

OR function \( \rightarrow \) just move dividing line
so \( b = \frac{1}{2} \)
NOR function \( \rightarrow \) cannot be realized

In general, line is \( \sum w_j x_j = b \), or \( \overrightarrow{w} \cdot \overrightarrow{x} = b \) a hyperplane.

**Perceptron Learning** (Minimize \( \langle x - y \overrightarrow{w} \overrightarrow{x} - b^2 \rangle \))

For binary input-output, we can have
an update rule
\[ \overrightarrow{w} = \overrightarrow{w} + y \overrightarrow{x} \]

Start with \( \overrightarrow{w} = (0, 0) \)
Put on \( \overrightarrow{x} = (1, 1) \)
\[ \overrightarrow{w} = (0, 0) + 1(1, 1) = (1, 1) \]
Other values for \( \overrightarrow{x} \) do not contribute

\[ b = b + y \overrightarrow{w} \overrightarrow{x} - \langle y \rangle \]
\[ w = w + 1 (1, 1) (1) - y_2 \]
\[ w = 3/2 \]
All of this becomes a bit easier if we switch to \((y = \pm 1)\) notation:

\[
\begin{align*}
\{ & y = \pm 1 \} \\
& x = \pm 1 \\
& y = \frac{1}{N} \sum_{i=1}^{N} \omega_i x_i = \frac{1}{2} [w_1 x_1 + w_2 x_2]
\end{align*}
\]

Consider \(N\) sets of Boolean functions.

We also map threshold onto input:

\[
\text{Input}(n) = [x_1(n), x_2(n), \ldots, x_N(n)]
\]

\[
\text{Output}(n) = y(n) = \pm 1
\]

Training constitutes \(\{ \tilde{x}(n), y(n) \}\) sets for learning of

\[
\text{Class 1: } w^T(n) x(n) \geq 0
\]

\[
\text{Class 2: } w^T(n) x(n) \leq 0
\]

Calculate \(\tilde{y}(n) = w^T(n) x(n)\), \(\uparrow\) from training set

\[
\text{Calculate current weights}
\]

\[
\text{Update rule}
\]

\[
\tilde{w}(n+1) = \tilde{w}(n) + \frac{1}{2} [y(n) - \tilde{w}(n) \cdot x(n)] x(n)
\]

weights on \(n^\text{th}\) iteration

\(\uparrow\) on response to \(n^\text{th}\) \((y, \tilde{x})\) pair.
\[ y(n) = \hat{y}(n) \text{ implies } \hat{w}(n+1) = \hat{w}(n) \]
\[ y(n) \neq \hat{y}(n) \text{ implies } \begin{cases} \hat{w}(n+1) = \hat{w}(n) + \hat{x}(n) & \text{Class 1} \\ \hat{w}(n+1) = \hat{w}(n) - \hat{x}(n) & \text{Class 2} \end{cases} \]

where we have two training sets.

Set of class 1 \((y(n), \hat{x}(n))\) \(y(n) = 1\) \(\forall n\)

Set of class 2 \((y(n), \hat{x}(n))\) \(y(n) = -1\) \(\forall n\)

Note \(\hat{w} = [b, w_1, w_2, \ldots]\)

\(\hat{x} = [-1, x_1, x_2, \ldots]\)

\(\hat{w} \cdot \hat{x} = \hat{w}^T \hat{x} = w_1 x_1 + w_2 x_2 + \ldots - b\)

\[ \text{Categorization by Perceptron} \]

\[ \begin{array}{c}
\text{Correct categorization:} \\
\hat{w}(n+1) = \hat{w}(n)
\end{array} \]

\[ \begin{array}{c}
\text{Wrong categorization:} \\
\hat{w}(n+1) = \hat{w}(n) + y(n) \hat{x}(n)
\end{array} \]

Learning makes weights an average over \(\hat{x}(n)\) examples.
Convergence theorem.

Need to show that correction grows faster than error. Do this in 3 steps.

1) Consider correction first. Since we make n errors, leading to n updates.

That is, \( \tilde{w}(n) \tilde{x}(n) \leq 0 \) for set of class 2

yet \( y(n) = +1 \)

\[ \therefore \tilde{w}(n+1) = \tilde{w}(n) + \tilde{x}(n) \]

\[ \therefore \tilde{w}(n) = \tilde{w}(n-1) + \tilde{x}(n) \]

iterate down to \( \tilde{x}(1) \)

\[ \|w\| = \frac{1}{n} \sum_{k=1}^{n} x(k) + \tilde{w}(1) \]

\[ \uparrow \]

no loss to take \( \tilde{w}(1) = 0 \)

Consider a potential class 1 solution

denoted by \( \bar{w}_1 \)

Then \( \bar{w}_1^T \tilde{w}(n+1) = \frac{1}{n} \sum_{k=1}^{n} \bar{w}_1^T x(k) \)

\[ \Delta = \min_{\tilde{x}(n) \in \text{set 1}} \frac{\bar{w}_1^T \tilde{x}(n)}{\bar{x}(n)} \]

\[ \|w\| \geq n \Delta \]

Cauchy-Schwarz: \[ \| \bar{w}_1^T \tilde{x}(n) \|^2 \geq (n \Delta)^2 \]

but \[ \| \bar{w}_1 \| \| \tilde{x}(n) \|^2 \geq \| \bar{w}_1^T \tilde{w}(n+1) \|^2 \]
\[ \| \tilde{w}_2 \|_2^2 / \| \tilde{w}(n+1) \|_2^2 > C(n^2) \]

\[ \| \tilde{w}(n+1) \|_2^2 > \frac{L^2}{\| \tilde{w}_2 \|_2^2} - \frac{n^2}{\| \tilde{w}_2 \|_2^2} \]

Weight correction scales as square of the number of iterations.

2) Now look at growth of error.

As above, the change in weight upon update for \( k=1, \ldots, n \) is

\[ \tilde{w}(k+1) = \tilde{w}(k) + \tilde{x}(k) \]

\[ \| \tilde{w}(k+1) \|_2^2 = \| \tilde{w}(k) + \tilde{x}(k) \|_2^2 \]

\[ = \| \tilde{w}(k) \|_2^2 + \| \tilde{x}(k) \|_2^2 + 2 \tilde{w}^T(k) \tilde{x}(k) \]

\[ \leq \| \tilde{w}(k) \|_2^2 + \| \tilde{x}(k) \|_2^2 \]

Now iterate from \( k=n \) downward

\[ \| \tilde{w}(n+1) \|_2^2 - \| \tilde{w}(n) \|_2^2 \leq \| \tilde{x}(n) \|_2^2 \]

\[ \sum_{k=1}^{n} \| \tilde{x}(k) \|_2^2 \leq \frac{n}{1} \| \tilde{w}(n) \|_2^2 \]
\[ \| \omega(n+1) \|^2 \leq \sum_{k=1}^{n} \| x(k) \|^2 \]

\[ \beta = \max_{x(n) \in \text{set 1}} \| x(n) \|^2 \]

\[ \beta n \leq \text{linear growth of error} \]

We now have two constants.

1) gave \[ \| \omega(n+1) \|^2 > \frac{\alpha^2}{\| w_1 \|^2} n^2 \]

2) gave \[ \| \omega(n+1) \|^2 \leq \beta n \]

These are equal, i.e., continuation beats \textit{error for a finite value of} \( n \):

\[ \frac{\alpha^2}{\| w_1 \|^2} n^2 = \beta n \]

or \( n = \frac{\beta}{\alpha^2} \| w_1 \|^2 = \text{some number} \)

minimum learning steps

\[ \text{Finis!} \]

Even if the example does not follow the same to be split exactly with a hyperplane, the perceptron will converge to limit solutions.