Thoughts about training

- *Hebb learning*: The Hebb learning is based on the premise that if two neurons were active simultaneously, then the strength of the connection between then should be increased. This was later extended to include increasing the weights when two neurons simultaneously disagreed too.
- *Perceptron learning rule*: The perceptron is also called the Rosenblatt neuron. This is considered to be more powerful than Hebb training. One can show that the perceptron weights will converge if a solution exists. Perceptrons use a threshold output function. The updates occur only when there is an error.
- *Widrow-Hoff Learning Rule*: Also known as the LMS, Delta or batch learning rule. The delta rule adjusts the weights to reduce the difference between *s* to the output unit and the desired output. This results in the smallest mean square error. This improvement ensures that the training gives you a more generalized system (inputs which are similar but not exact to the training vectors).

A Simple Training Rule

- If y_T is the target value, y_A the actual value and $d = y_T - y_A$ is the difference (the error) then we have the following three conditions.
- We note that if the threshold function is used, d can take three values $d = 0, -1, +1$.
    - $d = 0$ implies that there is no error. i.e.
    $$\mathbf{w}^{'} = \mathbf{w}$$
    - $d = -1$ implies that $y_T = 0$ and $y_A = 1$ then
    $$\mathbf{w}^{'} = \mathbf{w} - \eta\mathbf{x}$$
    - $d = 1$ implies that $y_T = 1$ and $y_A = 0$ then
    $$\mathbf{w}^{'} = \mathbf{w} + \eta\mathbf{x}$$
- Combining we obtain $\mathbf{w}^{'} = \mathbf{w} + \eta d\mathbf{x}$

The Perceptron

- The perceptron learning rule is more powerful than the Hebb learning rule.
- If there exists a weight matrix that gives us the correct classification, the perceptron iterative learning procedure can be shown to converge to the correct weights.
- Perceptrons can differentiate patterns only if they are linearly separable.
- The output of the perceptron $y = F(s)$ is computed using the activation function.

$$F(s) = \begin{cases} 1 & \text{if } s > \theta \\ 0 & \text{if } -\theta \leq s \leq \theta \\ -1 & \text{if } s < -\theta \end{cases}$$

The Perceptron ...

- For each training input, the net (s) would calculate the response of the output unit and then compare it with the target to find the error.
- The following formular is used to update when an error is found.

$$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i$$

  where $t = \{-1, 0, -1\}$ the target value and α the learning rate.
- The following items indicate some instances where the perceptron fails to update.
  - The computed output $y = 1$ while the target $t = 0$. Even though there is an error the contribution $tx_i = 0$ which indicates no updates. Similarly for $x_i = 0$.
  - $y = 0$ and $t = 1$. Here the weight would be updated in the direction of the target vector.

The Perceptron Algorithm

Step 0 Initialize weights and bias. Set learning rate $\alpha \in (0, 1]$.

Step 1 While stopping condition is false, do Step 2–6.

Step 2 For each training pair $\mathbf{s} : t$, do Step 3–5.

Step 3 Set activations of input units: $x_i = s_i$.

Step 4 Compute the response of output unit:

$$y\_in = b + \sum_i x_i w_i;$$

$$y = \begin{cases} 1 & \text{if } y\_in > \theta \\ 0 & \text{if } -\theta \leq y\_in \leq \theta \\ -1 & \text{if } y\_in < -\theta \end{cases}$$

The Perceptron Algorithm ...

Step 5 Update weights and bias if an error occurred for this pattern.

```
If $y \neq t$,
    $w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i$,
    $b(\text{new}) = b(\text{old}) + \alpha t$.
else
    $w_i(\text{new}) = w_i(\text{old})$,
    $b(\text{new}) = b(\text{old})$.
```

Step 6 Test stopping condition: If no weight changed in Step 2, stop; else, continue.

Algorithm Description

- The given algorithm is suitable for either binary or bipolar input vectors, bipolar targets, fixed θ, and adjustable bias.
- The θ does not play the same role as in the previous case, and hence both bias and the threshold is needed.
- The algorithm is not particularly sensitive to the choice of initial weights and the initial learning rate.
- Note that the patterns are updated only if there in an error. That is if correct responses are given less leaning occurs.
- We see that instead of one separating line now we have two net $> \theta$ (line separating the positive response from the region of sero response) and net $< -\theta$ (line separating the zero response from the negative response). The separation of the two lines is decided by θ.
- Hence the previous analysis of interchangeability of the bias and the threshold does not apply here.
- The band formed with the two lines is called the undecided region.

AND Function: Binary inputs, Bipolar targets

- Assume $\alpha = 1$ and $\theta = 0.2$. The weight change is $\Delta\mathbf{w} = t(x_1, x_2, 1)$ if an error occurs.
- Presenting the first input (the truth table is taken from the previous example)

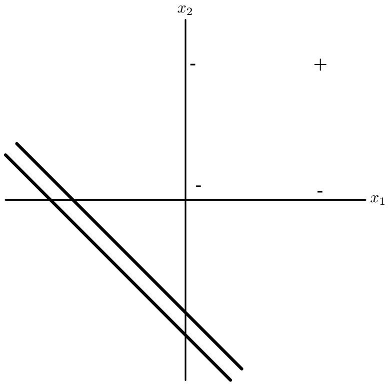
Input $(x_1, x_2, 1)$	**Net**	**Out**	**Target**	**Weight Changes** $(\Delta w_1, \Delta w_2, \Delta b)$	**Weights** (w_1, w_2, b)
					(0,0,0)
(1,1,1)	0	0	1	(1,1,1)	(1,1,1)

- The separating lines are

$$x_1 + x_2 + 1 = 0.2;$$
$$x_1 + x_2 + 1 = -0.2.$$

AND Function: Binary inputs, Bipolar targets ...

- The classification is correct for the first input and it is illustrated below



AND Function: Binary inputs, Bipolar targets ...

- Presenting the second input we have the following:

Input $(x_1, x_2, 1)$	**Net**	**Out**	**Target**	**Weight Changes** $(\Delta w_1, \Delta w_2, \Delta b)$	**Weights** (w_1, w_2, b)
					(1,1,1)
(1,0,1)	2	1	-1	(-1,0,-1)	(0,1,0)

- The separating lines now become

$$x_2 = 0.2 \quad \text{and,}$$
$$x_2 = -0.2$$

AND Function: Binary inputs, Bipolar targets ...

- At the end of presenting all the training data it is called a single epoch.
- The same training data is presented at the next epoch until we obtain a reasonable answer.
- For this example that answer comes at the 10th epoch. The boundary lines at this point are given by:

$$x_2 = -\frac{2}{3}x_1 + \frac{7}{5}$$
$$x_2 = -\frac{2}{3}x_1 + \frac{19}{15}$$

- In general the bias determines the separation of the two boundary lines. Hence it cannot be arbitrarily chosen.

Additional Comments

- If we change the above problem to bipolar inputs and targets with $\alpha = 1$ and $\theta = 0$, the results would be more refined.
- It is better to have a procedure that will continue to learn and improve even after the classifications are all correct.
- In the previous problem the weight updates cease as soon as all training patterns are classified correctly.
- However the result is obtained much faster than the binary input case.
- The perceptron rule can be shown to converge unlike other neural training s. The procedure involves showing that the number of steps used to correctly classify two patterns is bounded. Hence finite.
- The rule states that *if a solution exists* it will be found.

The Adaptive Linear Neuron (ADALINE)

- This typically uses bipolar activations for its input signals and target output.
- The architecture has weights that are adjustable.
- The ADALINE is trained using the Delta Rule (also known as the LMS or Widrow-Hoff rule)
- Even though the above rule can be used for many output units, the ADALINE has only one output unit.
- During training the activation function is the identity function. Hence for all derivations involving training uses the output of the neuron $y_j = s_j$.
- After training the threshold function is used during the pattern classification.

The Delta Rule

Delta Rule for Single Output Unit

- The delta rule changes the weights of he neural connection so as to minimize the difference between the net input to the output unit *S*, and the target value *t*.
- The aim is to minimize the error for al training patterns however this is accomplished by minimizing the error for each patterns one at a time.
- The delta rule for adjusting the *I*th weight for each pattern is

$$\Delta w_I = \alpha(t - s)x_I.$$

- It is derived in the following manner. The mean squared error for a particular training pattern is given by

$$E = \frac{1}{2}(t - s)^2$$

The Delta Rule ...

- E is a function of all the weights w_i, $i = \overline{1, n}$.
- The gradient $\partial E / \partial w_I$ gives the direction of most rapid increase in E, hence the error can be decreased by adjusting the weight w_I in the direction of $-\partial E / \partial w_I$, i.e.

$$\Delta w_I = -\alpha \frac{\partial E}{\partial w_I}.$$

  where α is the learning rate (descending rate).

Delta rule for several outputs

- The delta rule for adjusting the weight from the Ith input to the Jth output unit for each pattern is

$$\Delta w_{IJ} = \alpha (t_J - s_J) x_I$$

The Delta Rule ...

- The mean square error for a particular training pattern is

$$E = \frac{1}{2}\sum_{j=1}^{m}(t_j - s_j)^2.$$

- In the multiple output neuron case we again consider each weight separately for the error reduction.
- The local error will reduce most rapidly for a given learning rate by adjusting the weights according to

$$\Delta w_{IJ} = -\alpha\frac{\partial E}{\partial w_{IJ}}$$

The Learning Algorithm

```
Step 0  Initialize weights. Set learning rate $\alpha$.
Step 1  While stopping condition is false, do Step 2–6.
Step 2  For each bipolar training pair $\mathbf{s} : t$, do Step 3–5.
Step 3  Set activations of input units, $i = \overline{1, n}$: $x_i = s_i$.
Step 4  Computer net input to output unit:
            $$y\_in = b + \sum_i x_i w_i.$$
Step 5  Update bias and weights, $i = \overline{1, n}$:
            $$b(\text{new}) = b(\text{old}) + \alpha(t - s),$$
            $$w_i(\text{new}) = w_i(\text{old}) + \alpha(t - s)x_i$$
Step 6  Test for stopping condition: If the largest weight change
        that occurred in Step 2 is smaller than a specified
        tolerance, then stop; otherwise continue.
```

Some Comments

- The learning rate α has to be chosen appropriately:
  - A Small value will make the learning process extremely slow.
  - A large value will result in the learning process not converging.
- An appropriate value would be $\alpha = 0.1$.
- For a single neuron, a practical range for the learning rate α is $0.1 \leq n\alpha \leq 1.0$. Were n is the number of input neurons.
- The initial weights are set to small random values.

The Application Algorithm

Step 0 Initialize weights according to ADALINE algorithm.

Step 1 For each bipolar input vector **x**, do steps 2–4.

Step 2 Set activation of the input units to **x**.

Step 3 Compute net inputs *y_in* to output unit.

Step 4 Apply the activation function:

$$y = \begin{cases} 1 & \text{if } y\_in \geq 0; \\ -1 & \text{if } y\_in < 0. \end{cases}$$

- Some simple examples are given in Fausett pp. 83–86.
- It is seen that in most cases the resulting separating line is also obtained via the Perceptron rule.