CHARACTERISTICS OF SINGLE PERCEPTRON TRAINING.

Eng. Momchil Mihaylov Milev, Technical University - Sofia
Dr. Eng. Marin Hristov Hristov, Technical University - Sofia

Synopsis: The report summarizes results obtained and observations made while applying LMS and Single Perceptron Rule training algorithms to a Single Perceptron trained for a linear separability classifier. Experimental data presented demonstrates basic concepts of LMS and Perceptron rule training, general characteristics and conclusions drawn on the basis of both practical and analytical considerations of training models.

A single two input one output Processing Element (PE) simulates an artificial neuron linearly combining the weighted input signals (from its synaptic connections) and then applying a non-linear saturation function over the above sum to produce the neuron responce to its internal activity level. Such a PE input weightis can be adjusted (trained) by a number of various optimization algorithms iteratively minimizing an output error estimate function, so that the output responce matches with the desired pattern of responces to the input stimuli. This article briefly shows the essence, characteristics and experimental results for the case of two variations of such algorithms for training of a single PE - LMS algorithm and the Perceptron Rule for the McCulloc-Pitts "neuron". Both algorithms update input weights of the PE based on the error "signal" calculated at the output.The differences between these algorithms are related to the type of the PE and the statistical characteristics of the training data. Similarities as well as differences are considered briefly at the end of the presented study. The objective of the study is to show some of the essential characteristics of the above mentioned algorithms on the example of a single PE trained to classify 2-Dimensional input vector data into two separate classes.

Training data. Two sets (classes) of data are generated by drawing data from two independent random generators. Data is then classified (associated) in two classes: linearly separable data and data vectors which can not be linearly separated - linearly unseparable data. Random data is further filtered out to represent certain statistical characteristics, which for simlicity here are not listed.

A brief information on the Least Mean Square (LMS) training algorithm for the case of the "linear combiner" and a McCulloc-Pitt's "hard limiter" - the Perceptron Rule follows.

Least Mean Square Algorithm

The LMS algorithm also known as the "delta rule" or "Widrow-Hoff rule" is applied to a linear adaptive element to adjust its free parameters so that the "mean squared

error" between the output value of the Processing Element (PE) and its desired response is minimized.
The input-output relation of the "linear Processing Element" ("linear combiner") is given by:

$$y = \Sigma\, w_k\, x_k$$

where k the k-th vector component (input) and y is the output of the linear PE. The free-parameters w_k are adjusted according to the "error signal" defined as:

$$e = d - y$$

the difference between the desired output response d and its actual value y. The cost-function to be minimized is defined as

$$J = S\, E\{e^2\}$$

where $E\{.\}$ is the expected value the squared error over the ensemble set of data.
The LMS algorithm can operate under the assumption of non-stationery environment, in which case the ensemble averages are not available and replaced by "instantaneous estimates":

$$y(n) = \Sigma\, \varpi_j(n)\, x_k(n)$$

where ϖ_j is the "estimate" of the j-th weight vector component. The weight vector components are updated at each iteration by:

$$\varpi_k(n+1) = w_k(n) + \eta(n)\, e(n)\, x_k(n) \qquad \text{for } k = 1, 2, \ldots, p$$

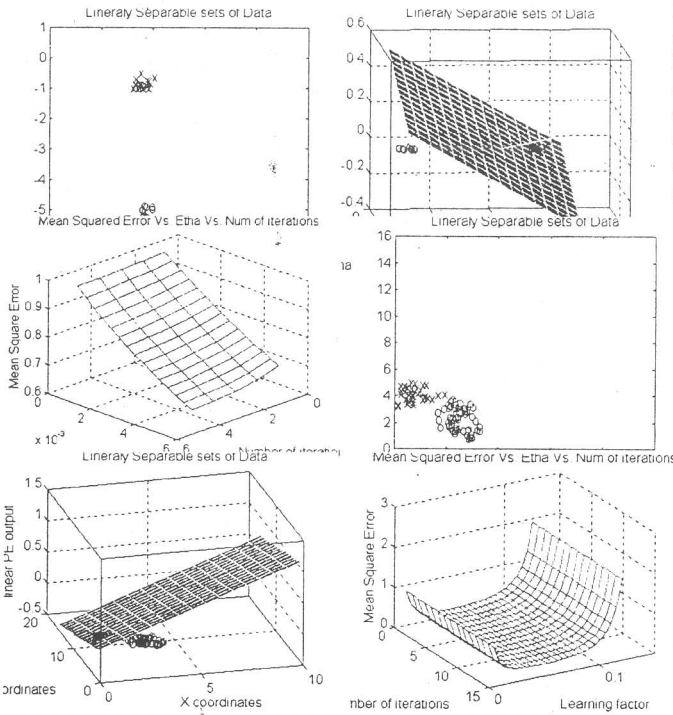
where p - is the number of inputs, and n -the iteration (time) index in adjusting the weights.
Data results produced
Several plots which follow depict the initial set of data - cluster classes A and B on a scatter-plot, a 3-D plot of The Mean Squared Error versa Etha - the learning rate parameter, and versa the Number of Iterations over the entire set of input vector presentations.

Case A Linearly Separable Data

- small number of data points are generated randomly and the "halo" vicinity is relatively small.
- LMS training is applied using time-varying "annealing" learning rate

$$\eta = \eta_o / (1 + n/\tau)$$

τ - "search-and-converge" time constant; n - current iteration index; $\eta_o = 2/\lambda max = 2/trace\{ X*X^T \}$ - "maximum" rate to satisfy stability ; X - input vector



PE output ("decision") surface plot is given. Mean Sqare Error ploted as a function of Number of iterrations over the training set of data and the 'lerning factor'is shown below.

Other case:

- more densely generated data points
- larger number of training data

Results observed:

On one hand, larger number of training data "improves" convergence of the weights in conjunction with "very small" values of the learning rate parameter, but increases overall training error over the entire training set, and if the learning rate is big enough to ensure "fast convergence", the number of iterations does not seem to contribute to system qualities.

On the other hand, if the learning rate is "large" the updates can lack convergence, in which case they will never reach the optimum solution; in the case of convergence conditions, but still "large" value of the learning rate, a non-decaying oscilations can occur "around" the optimum values of the weights.

Learning rate: Various schemes for the learning rate has been experimented:

- constant - η_0
- "exponentially decaying" $\eta = \eta_0/n$
- "search-then-converge" (annealing) schedule: $\eta = \eta_0/(1 + n/\tau)$

The last one has finally chosen with search constant $\tau = 10$, which means that during the first 10 iterations η does not significantly change from η_0 and only after starts decreasing to provide for better weight convergence as, by that time, hopefully the weight vector is "rambling" randomly "around" the optimum solution for W.

The initial rate is chosen to be $\eta_0 = 2 / \text{trace}\{ Rx\}$, where Rx is the auto-correlation matrix of X, thus accounting for the total "input power" and scaling the rate down properly to ensure convergence.(as by Widrow and Stearns, 1985; Haykin, 1991)

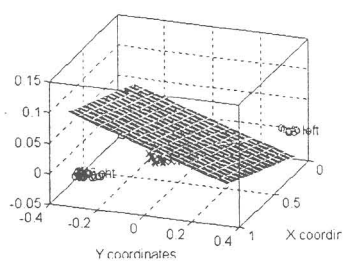
Steady-state-condition: the steady state mean square error was found to be: 0.53-0.72

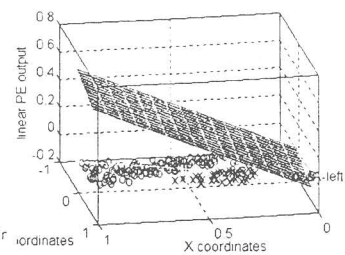
Case B: Linearly Inseparable Data
Results: any attempt for classification failed.
Reason: By definition the decision making "boundary" for the 2 input linear PE is:

$$x_1 w_1 + x_2 w_2 = 0,$$

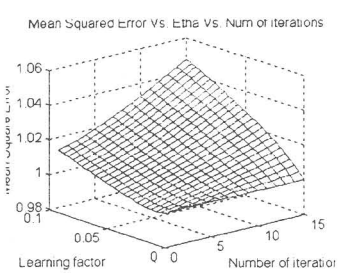
if "bias" weight is included: $x_1 w_1 + x_2 w_2 + \theta = 0$
i.e. it is a 'line' in the $\{x_1\ x_2\}$ plane. Thus, it is "genetically" incapable of separating linearly inseparable sets of data.

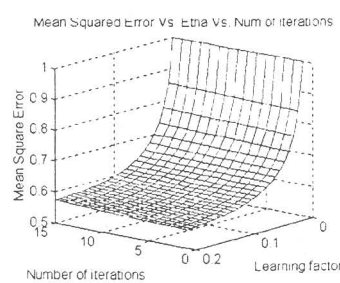
Example 1: "Dense" clustering, "large" (non-linear) separation (on left) and *Example 2 (on right)* "Sparse" clustering, "little" (non-linear) separation.





Mean Square Error plots for the above extracts of training data follow.
Notes: Again, note that in the first example the MSE surface versa learning rate parameter is in such a position/curvature that requires "small" rates of learning to minimize the error.





As we can see from the MSE plot of the second example, this is not the case: because of the "large" sparseness of input data the algorithm is supposed to attempt "larger" steps to converge the weights into a single solution for a given desired classification.

Perceptron Training

The "**Perceptron Rule**" is carried out by the "same" code as for LMS algorithm. Described by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\ [\ d(n) - y(n)]\ \mathbf{x}(n)$$

where:
$\mathbf{x}(n) = [-1\ \ x_1\ \ x_2\ \ x_3\ \ \ldots\ \ x_m]^T$ is the input vector

$\mathbf{w}(n) = [\theta \ w_1 \ w_2 \ w_3 \ \ldots \ w_m]^T$ is the weights vector

θ - offset 'bias' or treshold.

$d(n) = \text{sign}(\mathbf{w}^T \mathbf{x}(n))$

+1 if $\mathbf{x}(n)$ belongs to Class 1

-1 if $\mathbf{x}(n)$ belongs to Class 2

The differences from LMS are :

- in computation of d(n)
- adjusting θ (the same as the rest of the weights)
- convergence condition is met when:

$w(n_o) = w(n_o+1) = w(n_o+2) = \ldots$

(three consecutive weight vectors are the same)

Note: experimental data on training a 'signum' PE on linear separable and linearly inseparable data is not shown for briefness.

Observation: Because of the quantized output response, the Perceptron with a threshold non-linear function as its output reaches convergence and zero-error in only a few steps and is much more suitable for classification purposes of this kind (lenear separability classification) than the linear output combiner. Mean Squared Error of the hard-limited PE, is observed to be almost independent of the number of iterations and/or learning rate as long as it satisfies convergence condition: $0 < \eta \leq 1$

Summary and Conclusions

The above considered algorithms for adaptation of the free parameters of a some basic models of processing elements (neurons) show some commonalities and and differences. The commonalities include the adaptation process which takes the computed output error and by the scale of the learning rate parameter produce an adjustment to those free parameters such as they "improve" the "overall system performance". This means that the free parameters of the system are adapted on iteration-by-iteration basis, and the rule applied is one from or the other of an "error correction rule". In doing this, although the adjustment equations look pretty much the same, the two algorithms are different in that the LMS algorithm is applied to a linear adaptive element, while the perceptron rule is applied to a non-linear adaptive element. Furthermore, both algorithms make use of the "instantaneous error" at each iteration step (as opposed to other training algorithms like "the method of steepest descend" which is involving ensemble averaging of mean squared error over all of the presented training data.) Next, the two algorithms differ in that the Perceptron Rule "expects" that the input signals will not change during "training" and either each input vector will be repeatedly presented until it is classified correctly or the whole set of the same input vectors will be iterated until convergence is reached. The LMS algorithm is not limited to operating in that mode and neither it is limited to stationary environments - it has the ability to track the time varying input statistics. Also, in general, it operates in an "unknown" environment with which no particular form of cost function is associated or assumed. At the end, one of the most important considerations for both algorithms is the conflicting requirement for the value of the learning rate parameter:

- small values of η to produce "accurate" and stable weight updates, and
- large values of η to ensure the ability to track time varying input data and provide fast convergence.

References:

[1] Neural Networks comprehensive foundation.Haykin, 1991
[2] Neural Networks for patter recognition, C.Bishop, 1995