ADAPTIVE SWITCHING CIRCUITS

Bernard Widrow
and
Marcian E. Hoff

A. Introduction

The modern science of switching theory began with work by Shannon[1] in 1938. The field has developed rapidly since then, and at present a wealth of literature exists[2] concerning the analysis and synthesis of logical networks which might range from simple interlock systems to telephone switching systems to large-scale digital computing systems.

An example illustrating the use of switching theory is that of the design of an interlock system for the control of traffic in a railroad switch yard. The first step is the preparation of a "truth table", an exhaustive listing of all input possibilities (the positions of all incoming and outgoing trains), and what the desired system output should be (what the desired control signals should be) for each input situation. The next step is the construction of a Boolean function, and the following steps are algebraic reduction and design of the logical control system.

The design of the traffic control system is an example wherein the truth table must be followed precisely and reliably. Errors would be destructive. The design of the arithmetic element of a digital computer is another example wherein the truth table must be followed precisely.

There are other situations in which some errors are inevitable, however, and here errors are usually costly but not catastrophic. These situations call for statistically optimum switching circuits. A common performance objective is the minimization of the average number of errors. An example is that of prediction of the next bit in a correlated stochastic binary number sequence. The predictor output is to be a logical combination of a finite number of previous input sequence bits. An optimum system is a sequential switching circuit that predicts with a minimum number of errors.

Suppose that a record of the binary sequence is printed on tape and cut up into pieces (with indication of the positive direction of time preserved), say 25 bits long. Place all pieces where the most recent event is ONE in one pile, and the remainder in another pile. Delete the most recent bit on each piece of tape. If the statistical scheme could be discovered by which the pieces of tape are classified, this would lead to a prediction scheme. It is apparent that prediction is a certain kind of classification.

Assuming statistical regularity, a reasonable way to proceed might be to form a truth table, and let the data from each piece of tape be an entry in the table. It might be expected that with the data of 100 pieces of tape, a fairly good predictor could be developed. The truth table would have only 100 entries however, out of a total of 2^{24}. The "best" way to fill in the remainder of the truth table depends upon the nature of the sequence statistics and the error cost criteria. Filling in the table is a difficult and a crucial part of the problem. Even if the truth table were filled in, however, the designer would have the difficult task of realizing a logical network to satisfy a truth table with 2^{24} entries.

An approach to such problems is taken in this paper which does not require an explicit use of the truth table. The design objective is the minimization of the average number of errors, rather than a minimization of the number of logical components used. The nature of the logical elements is quite unconventional. The system design procedure is adaptive, and is based upon an iterative search process. Performance feedback is used to achieve automatic system synthesis, i.e., the selection of the "best" system from a restricted but useful class of possibilities. The designer "trains" the system to give the correct responses by "showing" it examples of inputs and respective desired outputs. The more examples "seen", the better is the system performance. System competence will be directly and quantitatively related to amount of experience.

B. A Neuron Element

In Fig. 1, a combinatorial logical circuit is shown which is a typical element in the adaptive switching circuits to be considered. This element bears some resemblance to a "neuron" model introduced by von Neuman[3], whence the name.

The binary input signals on the individual lines have values of +1 or -1, rather than the usual values of 1 or 0. Within the neuron, a linear combination of the input signals is formed. The weights are the gains $a_1, a_2, \ldots$, which could have both positive and negative values. The output signal is +1 if this weighted sum is greater than a certain threshold, and -1 otherwise. The threshold level is determined by the setting of a_0, whose input is permanently connected to a +1 source. Varying a_0 varies a constant added to the linear combination of input signals.

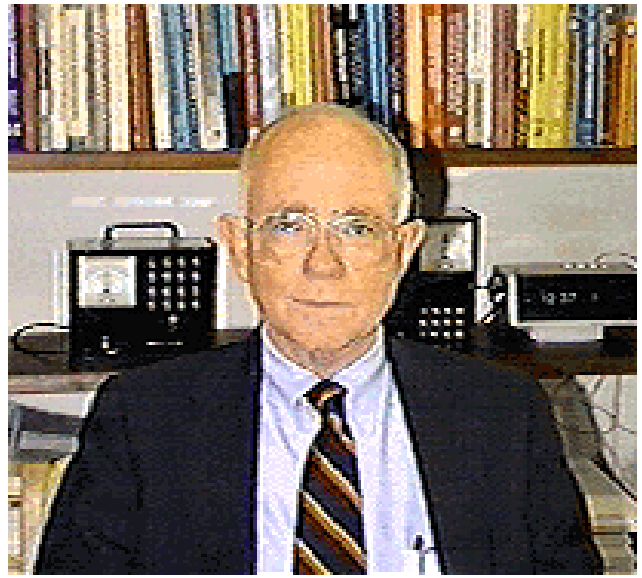
For fixed gain settings, each of the 2^5 possible input combinations would cause either a +1 or -1 output. Thus, all possible inputs are classified into two categories. The input-output relationship is determined by choice of the gains $a_0, \ldots a_5$. In the adaptive neuron, these gains are set during the "training" procedure.

In general, there are 2^{2^5} different input-output relationships or truth functions by which the five input variables can be mapped into the single output variable. Only a subset of these, the linearly separated truth functions[4], can be realized by all possible choices of the gains of the neuron of Fig. 1. Although this subset is not all-inclusive*, it

†Department of Electrical Engineering, Stanford University.

††Doctoral student in the Department of Electrical Engineering, Stanford University.

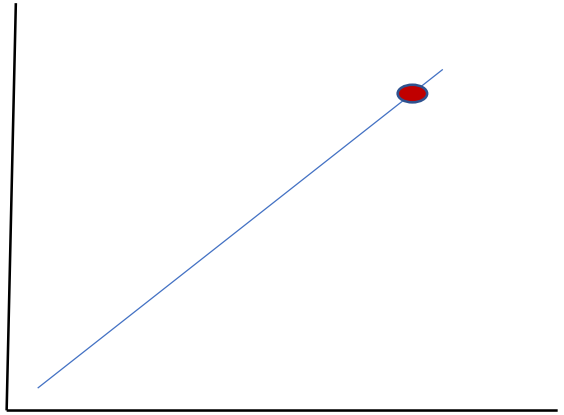
*It becomes a vanishingly small fraction of all possible switching functions as the number of inputs gets large.

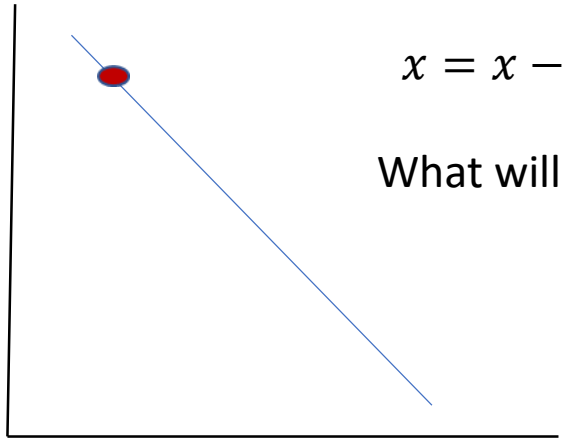


Bernard Widrow

Ted Hoff

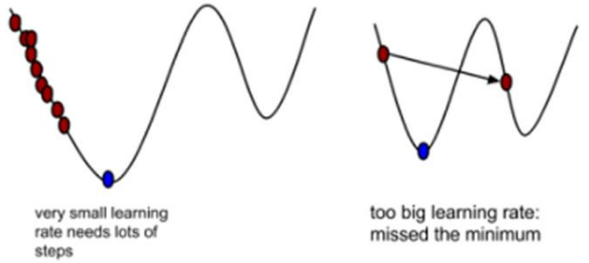
B. Widrow and M.E. Hoff, Jr., ``Adaptive Switching Circuits,'' *IRE WESCON Convention Record*, 4:96-104, August 1960.

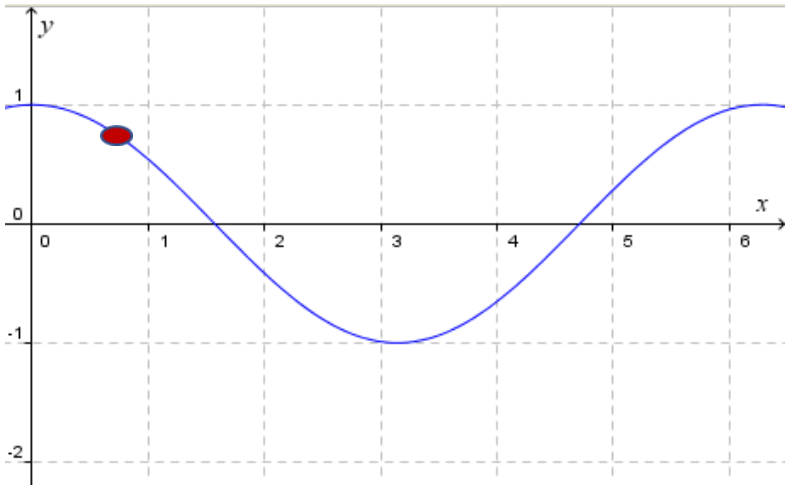




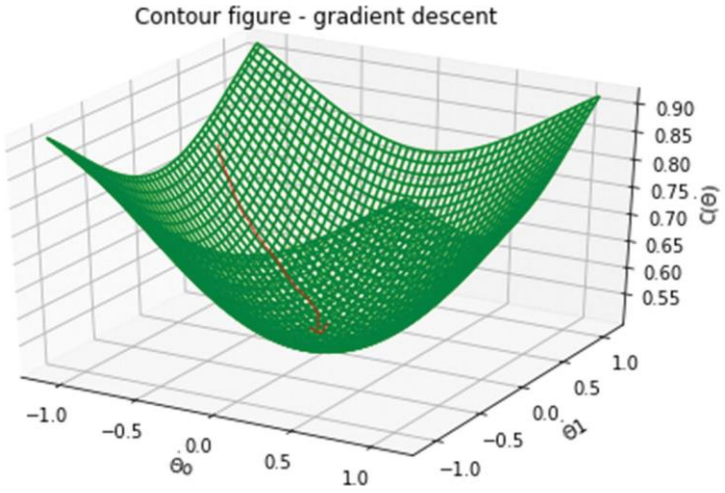
$$x = x - \alpha m$$

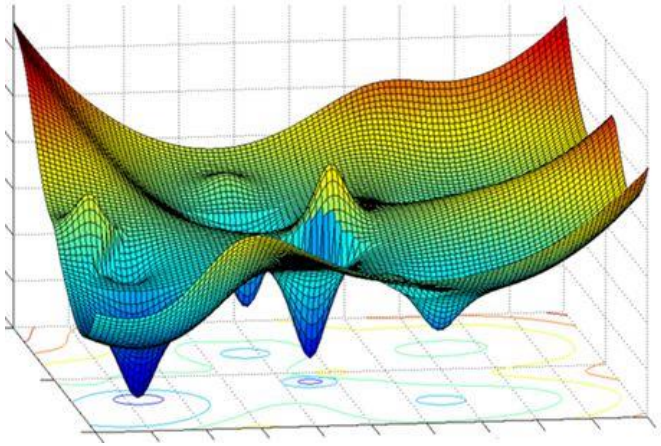
What will be the impact of α ?





$$x = x - \alpha \frac{dy}{dx}$$



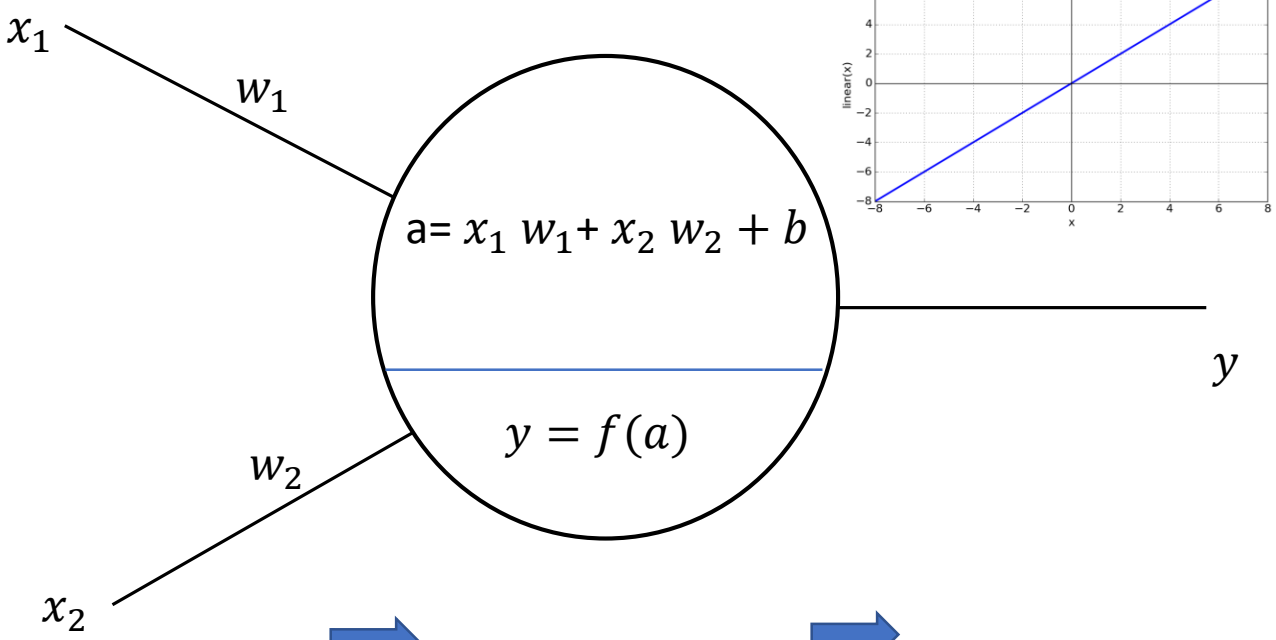


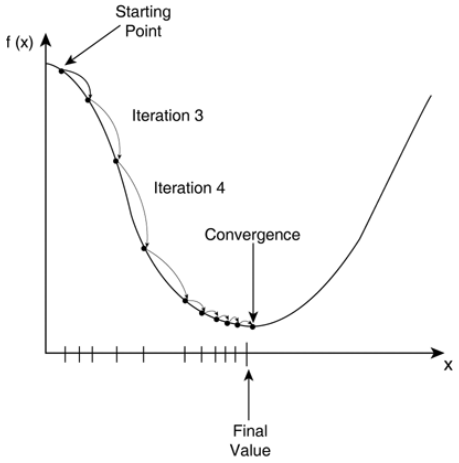
$$x_1 = x_1 - \alpha \nabla Y \qquad x_2 = x_2 - \alpha \nabla Y$$

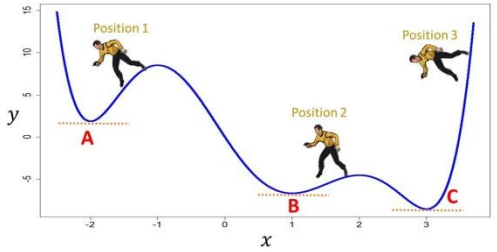
$$\nabla Y = \left\langle \frac{dy}{dx_1}, \frac{dy}{dx_2} \right\rangle \longrightarrow$$

Collection of partial derivatives into a vector

Single neuron with linear activation function







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

$y = a$

$t =$ target output

$$w_1 = w_1 - \alpha \frac{\partial E}{\partial w1}$$

$$\frac{\partial E}{\partial w1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w1}$$

$$\frac{\partial E}{\partial w1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial a}\frac{\partial a}{\partial w1}$$

Chain rule

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

$$\frac{\partial y}{\partial a} = 1$$

$$\frac{\partial a}{\partial w1} = \frac{\partial(x_1 w_1 + x_2\ w_2 + b)}{\partial w1} = x_1$$

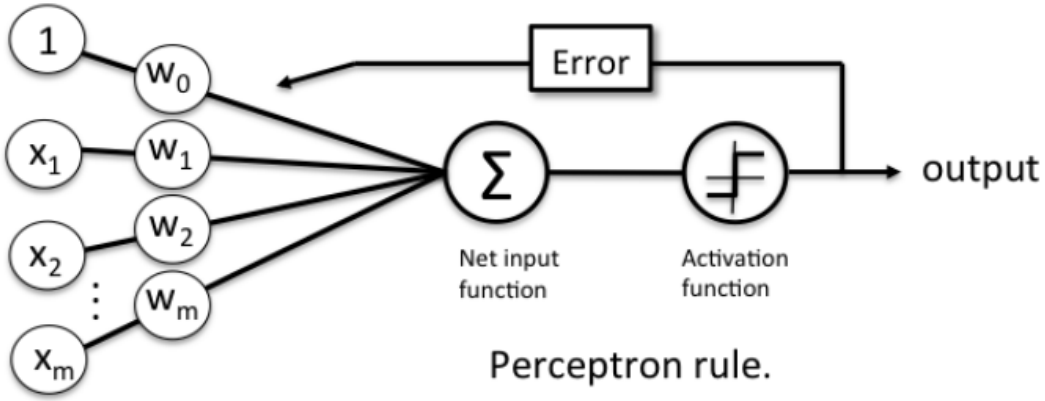
$$\frac{\partial E}{\partial w1} = -(t-y)\ x_1$$

$$w_1 = w_1 + \alpha(t-y)\ x_1$$

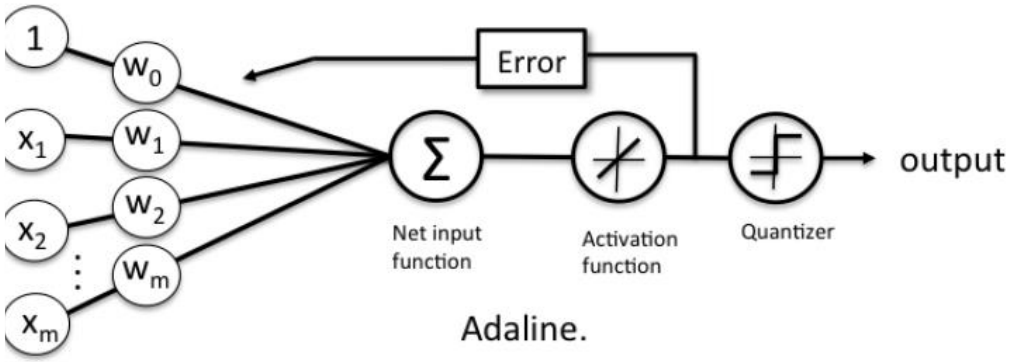
$$w_2 = w_2 + \alpha(t-y)\ x_2$$

$$b = b + \alpha(t-y)$$

$$w_i = w_i + \alpha\ (t - y_i)\ x_i$$



Perceptron rule.



Adaline.

What Adaline and the Perceptron have in common

- they are classifiers for binary classification
- both have a linear decision boundary
- both can learn iteratively, sample by sample (the Perceptron naturally, and Adaline via stochastic gradient descent)
- both use a threshold function

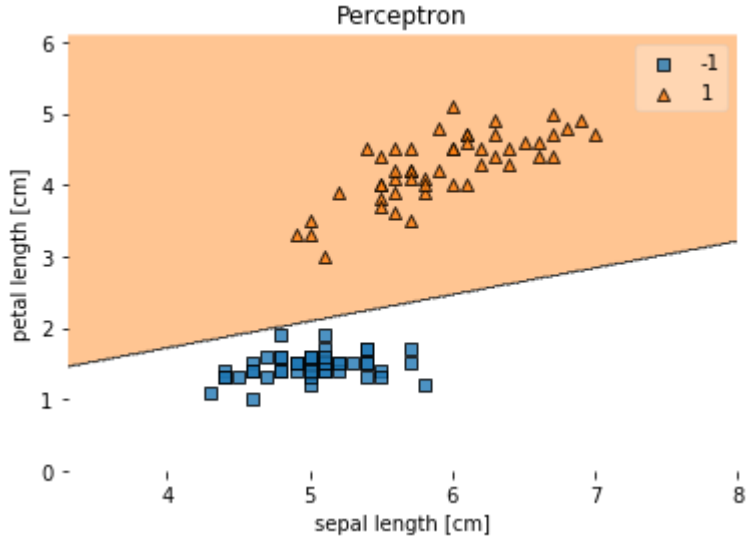
The differences between the Perceptron and Adaline

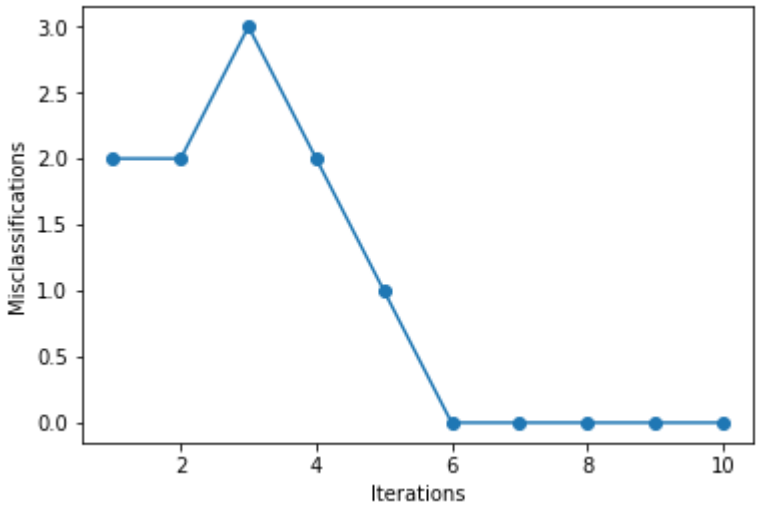
- the Perceptron uses the class labels to learn model coefficients
- Adaline uses continuous predicted values (from the net input) to learn the model coefficients, which is more "powerful" since it tells us by "how much" we were right or wrong

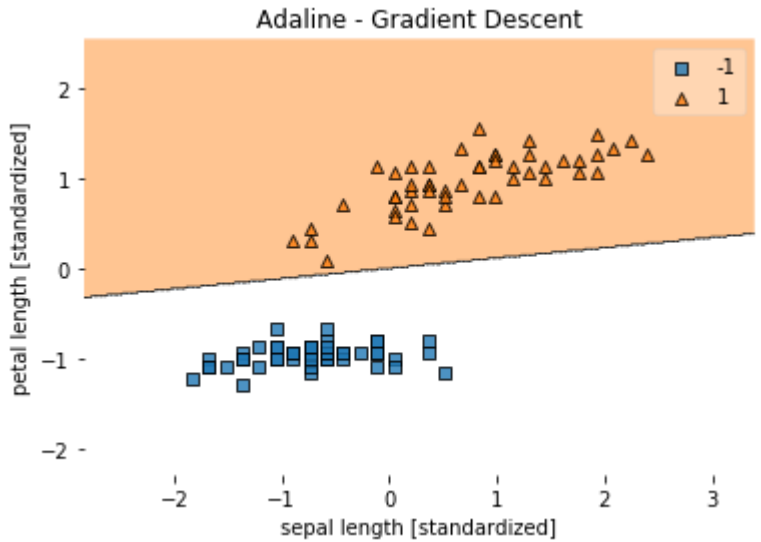
https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html

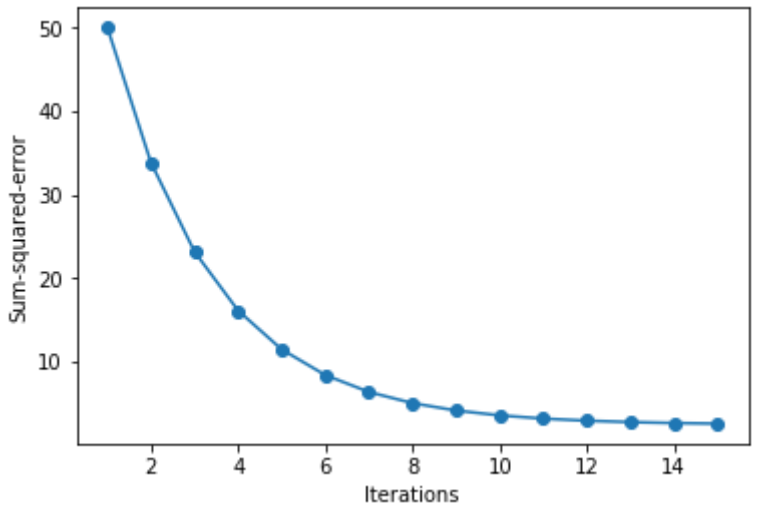
Iris virginica

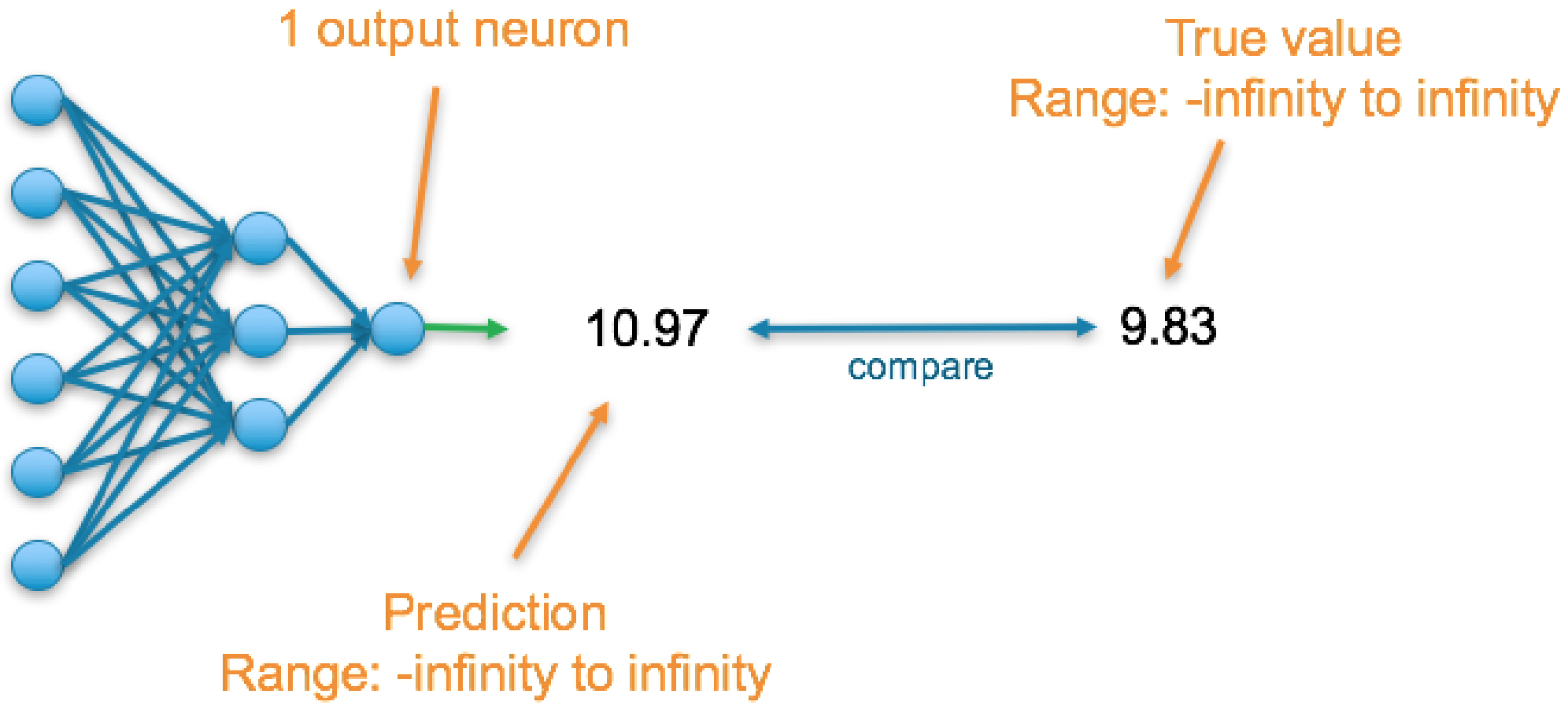
Iris versicolor


Perceptron
-1
1
petal length [cm]
sepal length [cm]


Misclassifications
Iterations


Adaline - Gradient Descent
-1
1
petal length [standardized]
sepal length [standardized]


Sum-squared-error
Iterations



$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

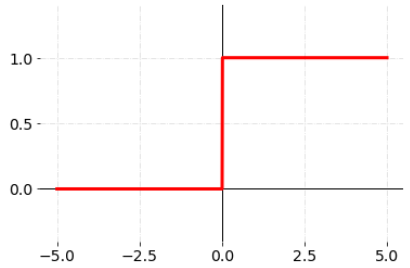
Where $\hat{y}$ is the predicted value and y is the true value

M-P model

- Not Automated
- Parameters computed manually.
- Data should be linearly separable.

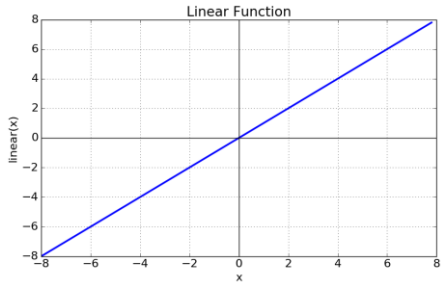
Perceptron

- Automated
- Perceptron Learning rule
- Data should be linearly separable
- Small margins (decision boundary)
- Used step activation function



Adaline

- Automated
- Parameter computing – using gradient
- Data should be linearly separable.
- Large margins (decision boundary)
- Linear activation function (identity function)
- Loss function : MSE
- Data should be linearly separable
- Output : not bounded [-inf to +inf]



Nonlinear activation functions are preferred as they allow the nodes to learn more complex structures in the data.