Perceptron Networks and Applications

M. Ali Akcayol
Gazi University
Department of Computer Engineering

Content

- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Perceptrons

- In supervised learning algorithms, the desired result is known for samples in the training data.
- The learning algorithms are simpler for the networks consisting of only one node in one layer.
- The modification of the weights is very simple.
- The perceptrons have simple description but limited capabilities.
- A perceptron is defined to be a machine that learns using examples.
- A perceptron also is defined as a stochastic gradient-descent algortihm that separate a set of n-dimensional space linearly.

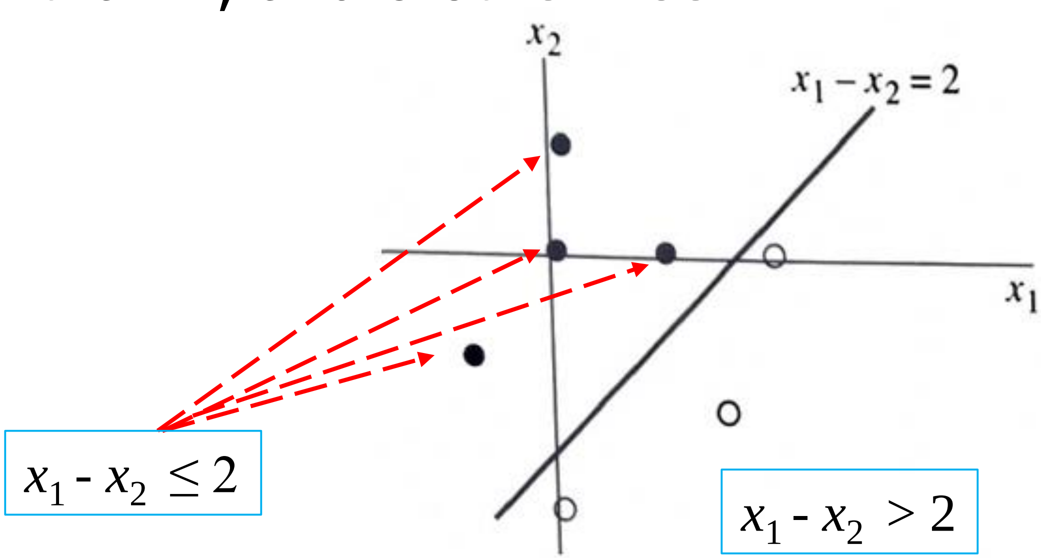
Perceptrons

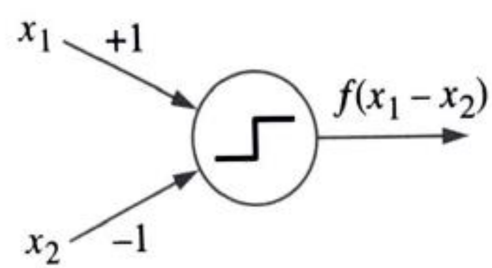
- A perceptron has a single output whose values determine that each input pattern belongs to which one of two classes.
- A perceptron can be represented by a single node.
- The perceptron applies a step function to the net weighted sum of its inputs.
- The input pattern is considered to belong to one class or the other.
- The output class is decided depending on whether the node output is 0 or l.

Perceptrons

Example

- Consider two-dimensional samples (0,0), (0,1), (1,0), (-1,-1) that belong to one class, and samples (2.1,0), (0, -2.5), (1.6, -1.6) that belong to another class.
- These classes are linearly separable.
- The node function is a step function.
- The output of the node is 1 if the net weighted input is greater than 2, and 0 otherwise.





Content

- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Linear separability

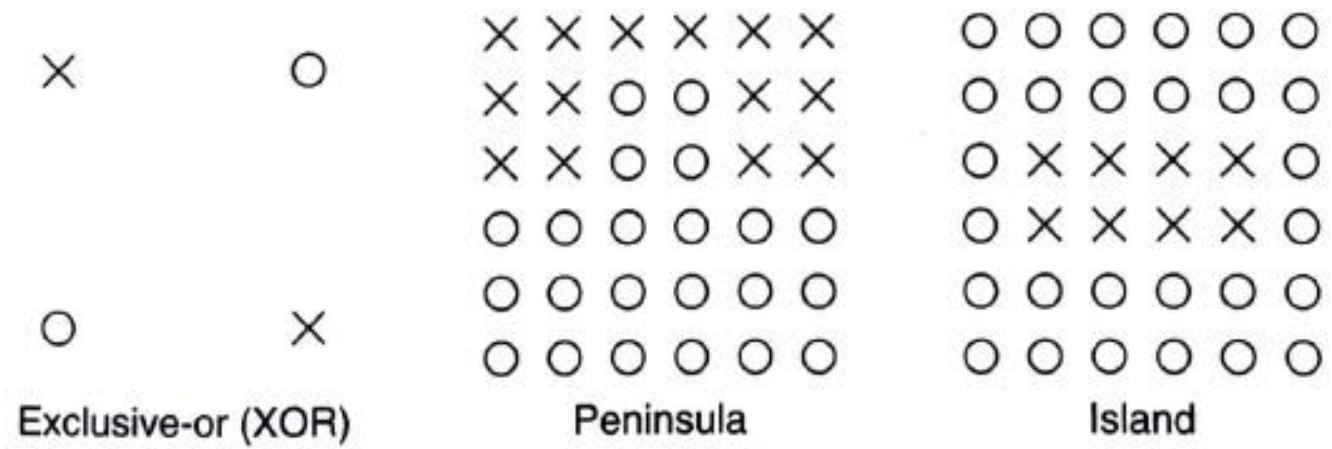
- If there exists a line that separates all samples of one class from the other class, such classification problems are said to be 'linearly separable'.
- The line's equation is

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

- If there is perceptron with weights w_0 , w_1 , w_2 for connections from inputs 1, x_1, x_2 , the perceptron can separate samples of two classes.
- If the samples are NOT linearly separable, i.e., no straight line can possibly separate samples belonging to two classes, then there cannot be any simple perceptron that achieves this task.
- This is the fundamental limitation of simple perceptrons.

Linear separability

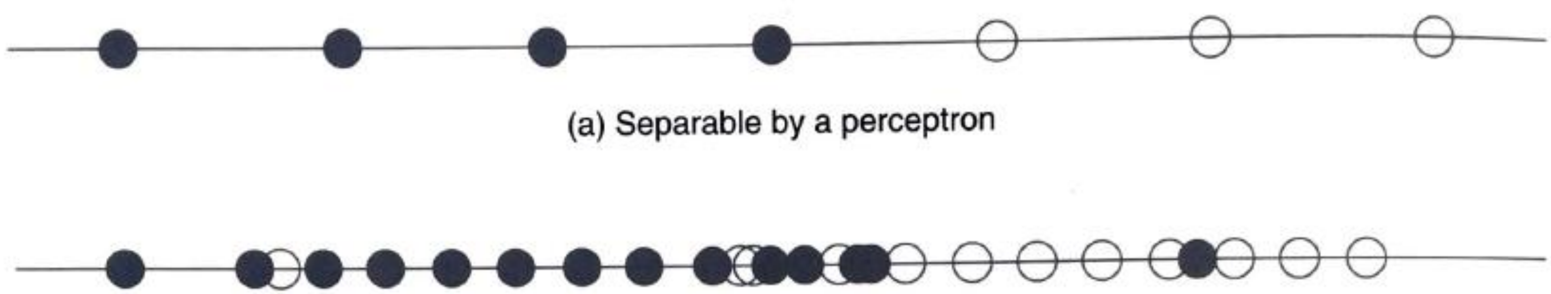
- Examples of linearly non separable classes are:



- Most real-life classification problems are linearly nonseparable.

Linear separability

- If there is only one input dimension x, then the two-class problem can be solved using a perceptron if and only if there is some value x_0 of x such that all samples of one class occur for $x > x_0$, and all samples of the other class occur for $x < x_0$.

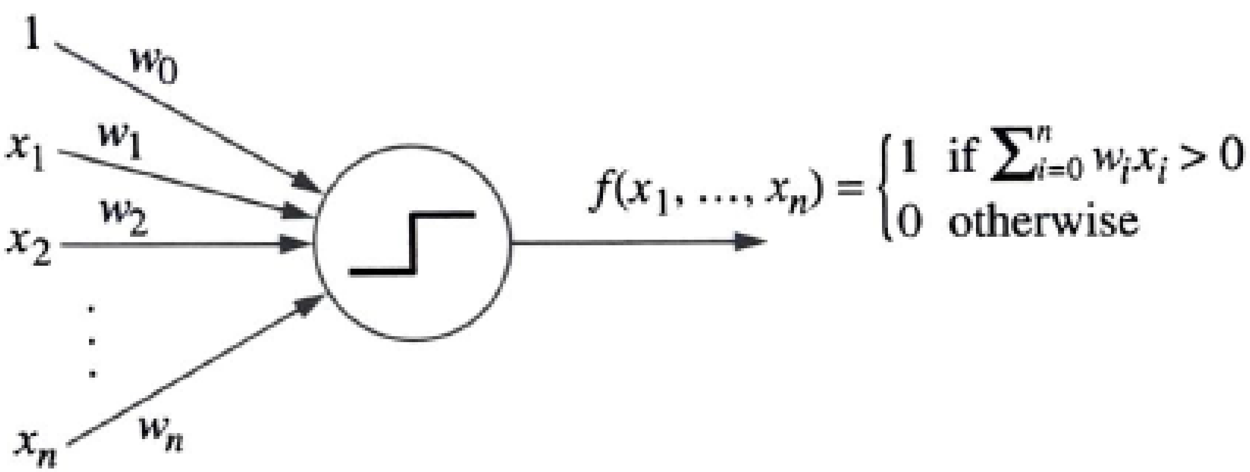


(a) Separable by a perceptron

(b) Not separable by a perceptron

Linear separability

- If there are three input dimensions, a two-class problem can be solved using a perceptron if and only if there is a plane that separates samples of different classes.
- As in the two-dimensional case, coefficients of terms correspond to the weights of the perceptron.
- A generic perceptron for n-dimensional space.



- For this perceptron, hyperplane is $\sum_{i=0}^{n} w_i x_i = 0$.

Linear separability

- For spaces of higher number of input dimensions, the geometric presentations need to be extended.
- Hyperplanes can separate samples of different classes in n-dimensional space.
- Each hyperplane in n dimensions is defined by the equation

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = 0$$

- Each hyperplane divides the n-dimensional space into two regions:

  1- $w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > 0$

  2- $w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n < 0$

- Training algorithms used to obtain the weights of a suitable perceptron.

Content

- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Perceptron training algorithm

- Perceptron training algorithm can be used to obtain appropriate weights of a perceptron that separates two classes.
- Using weight values, the equation of the hyperplane that divide the solution space can be derived.
- The developed perceptron can be used to classify new samples.
- Dot product or scalar product of two vectors, $\boldsymbol{w}$ and $\boldsymbol{x}$, is defined as follows,

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$$

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

$$\boldsymbol{w} \cdot \boldsymbol{x} = (w_1 x_1 + w_2 x_2 + \ldots + w_n x_n)$$

- Euclidean length $\|\boldsymbol{v}\|$ of a vector $\boldsymbol{v}$ is defined as,

$$\|\boldsymbol{v}\| = (\boldsymbol{v} \cdot \boldsymbol{v})^{1/2}$$

Perceptron training algorithm

- The presentation of the learning is simplified by using perceptron output values ∈ $\{-1, 1\}$ instead of $\{0, 1\}$.
- Weight values are randomly chosen between 0 and 1.
- It is assumed that the perceptron with weight vector w has output 1 if $w.x > 0$, and output -1 otherwise.
- If the network output differs from the desired output, the weights must be changed, otherwise cannot be changed.
- If a sample (i) belongs to class 0, but $w.i > 0$, then the weight vector needs to be modified.
- After each modification, the sample would have a better chance in the following iteration.

Perceptron training algorithm

- If $\boldsymbol{i}$ belongs to a class (desired node output is -1) but $\boldsymbol{w.i} > \boldsymbol{0}$, then the weight vector needs to be modified to $\boldsymbol{w} + \Delta\boldsymbol{w}$ so that $(\boldsymbol{w} + \Delta\boldsymbol{w}).\boldsymbol{i} < \boldsymbol{w.i}$
- $\Delta\boldsymbol{w} = -\eta.\boldsymbol{i}$, where $\eta > 0$.
- After modification of the weight, $\boldsymbol{i}$ would have a better chance of being classified correctly in the following iteration.

Algorithm Perceptron;
  Start with a randomly chosen weight vector $\boldsymbol{w}_0$;
  Let $k = 1$;
  while there exist input vectors that are misclassified by $\boldsymbol{w}_{k-1}$, **do**
    Let $\boldsymbol{i}_j$ be a misclassified input vector;
    Let $\boldsymbol{x}_k = class(\boldsymbol{i}_j).\boldsymbol{i}_j$, implying that $\boldsymbol{w}_{k-1} \cdot \boldsymbol{x}_k < 0$;
    Update the weight vector to $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \eta \boldsymbol{x}_k$;
    Increment k;
  end-while;

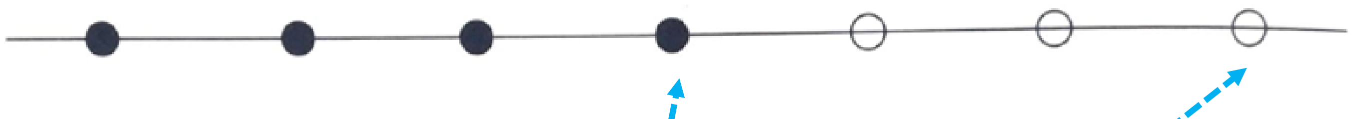
Perceptron training algorithm

- If $\boldsymbol{i}$ belongs to a class (desired node output is 1) but $\boldsymbol{w.i} < \boldsymbol{0}$, then the weight vector needs to be modified to $\boldsymbol{w} + \boldsymbol{\Delta w}$ so that $(\boldsymbol{w} + \boldsymbol{\Delta w}).\boldsymbol{i} > \boldsymbol{w.i}$
- Let $\boldsymbol{i}_1, \boldsymbol{i}_2, \ldots, \boldsymbol{i}_p$ denote the training set, containing $\boldsymbol{p}$ input vectors.
- We define a function that maps each sample to either +1 ($\boldsymbol{C}_1$) or -1 ($\boldsymbol{C}_0$).
- Samples are presented repeatedly to train the weights.

Perceptron training algorithm

Example

- Let there be 7 one-dimensional input patterns as shown below.



- The 7 input paterns can be separable linearly.
- Samples {0.0, 0.17, 0.33, 0.50} belong to one class (desired output 0), and samples {0.67, 0.83, 1.0} belong to the other class (desired output 1).
- For the initial randomly chosen value of $\boldsymbol{w_1} = \mathbf{-0.36}$, and $\boldsymbol{w_0} = \mathbf{-1.0}$, {0.83, 0.67, 1.0} are misclassified.

Perceptron training algorithm

Example – cont.

- For the input value 0.83, output is (0.83)(-0.36) – 1.0 = -1.2
- Then the sample has calculated class 0, which is an error (it would be 1).
- For η = 0.1, new weights are calculated as,

$$w_1 = -0.36 + (0.1)(0.83) = -0.28$$

$$w_0 = -1 + (0.1)(1) = -0.9$$

- For the new weights, some samples are still misclassified.
- The weights are modified iteratively and the final weight values are,

$$\mathbf{w_1 = 0.3}$$
$$\mathbf{w_0 = -0.2}$$

Perceptron training algorithm

Example – cont.

- The progress of the training process.

Presentation No.	2	3	4	5	6	7	8	9
Sample	0.33	0.67	0.17	1.00	0.50	0.00	0.83	0.33
Desired output	−1	1	−1	1	−1	−1	1	−1
Net input	−0.99	−1.10	−0.80	−1.00	−0.80	−0.70	−0.80	−0.60
w_1	−0.28	−0.21	−0.21	−0.11	−0.11	−0.11	−0.03	−0.03
w_0	−0.90	−0.80	−0.80	−0.70	−0.70	−0.70	−0.60	−0.60
Number misclassified	3	3	3	3	3	3	3	3

Presentation No.	10	...	17	...	24	25	26	27
Sample	0.67	...	0.67	...	0.67	0.17	1.00	0.50
Desired output	1	...	1	...	1	−1	1	−1
Net input	−0.62	...	−0.15	...	−0.01	−0.04	0.25	0.01
w_1	0.04	...	0.29	...	0.35	0.35	0.30	0.30
w_0	−0.50	...	−0.20	...	−0.10	−0.10	−0.10	−0.20
Number misclassified	3	...	1	...	2	2	2	0

- What is the reason of the oscillations on weight values?

Perceptron training algorithm

- There are some important questions:
  - How long should we execute this training procedure?
  - What is the termination criterion (if the given samples are not linearly separable)
  - What is the appropriate choice of the learning rate?
  - How can the perceptron training algorithm be applied to problems in which the inputs are non-numeric values (color, label, name, …)?
  - Is there a guarantee that the training algorithm will always succeed whenever the samples are linearly separable?
  - Can the perceptron training algorithm work reasonably well when samples are not linearly separable?

Content

- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Termination criterion

- For many ANN learning algorithms, the termination criterion is ”stop when the goal is achieved”.
- For any kind of classifier, the goal is the correct classification of all samples.
- So the perceptron training algorithm runs until all samples are correctly classified.
- For perceptron, termination is assured if η sufficiently small and samples are linearly separable.
- If η is not appropriate or samples are not linearly separable, the algorithm runs indefinitely.
- How can we detect that this may be the case?

Termination criterion

- The amount of progress achieved in the recent past can be used to terminate the training.
- For linear classifier, if the number of correct classification has not changed in large of steps, the samples may not be linearly separable.
- The same problem may be occurred with the inappropriate choice of η.
- The different values of η may yield improvement for training phase.

Termination criterion

- In some problems, two classes overlap (not linearly separable).
- If the performance requirements allow some amount of misclassification, we can modify the termination criterion.
- For example, it may known that at least 6% of the samples will be misclassified (or user satisfied with 6%), the termination criterion should be modified.
- We can then terminate the training algorithm as soon as 94% of the samples are correctly classified.

Content

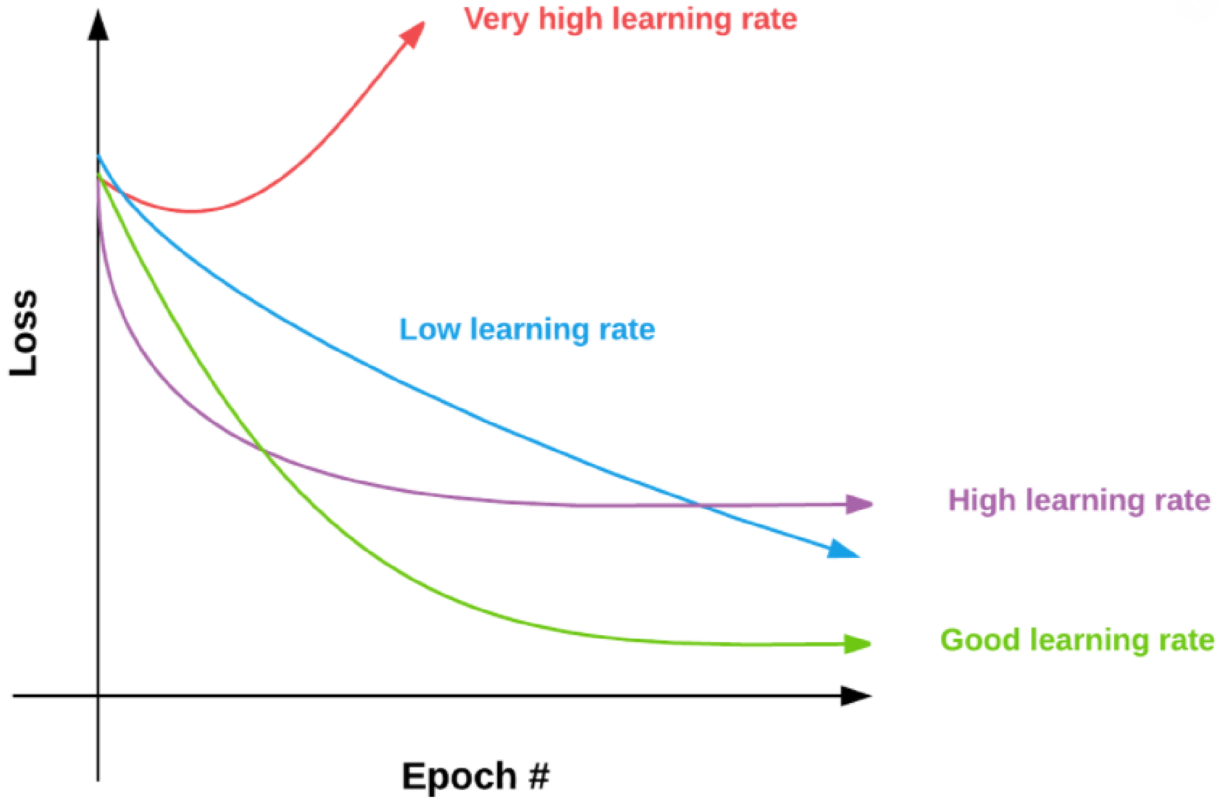
- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Choice of learning rate

- The examination of extreme cases can help derive a good choice for $\boldsymbol{\eta}$.
- If $\boldsymbol{\eta}$ is too large (e.g. 1.000.000), then the components of $\boldsymbol{\Delta w} = \pm\boldsymbol{\eta x}$ can have very large magnitudes.
- If $\boldsymbol{\eta}$ is too large, each weight update swings perceptron outputs completely in one direction as a result, the perceptron considers all samples to be in the same class.
- The system oscillates between extremes.
- If $\boldsymbol{\eta}$ is very small (e.g. $\boldsymbol{\eta} = \mathbf{0}$) the weights are never going to be modified.
- If $\boldsymbol{\eta}$ equals some too small value, the change in the weights in each step going to be too small. This makes the algorithm exceedingly slow.

Choice of learning rate

- If η is too large, the progress will start very fast, but eventually jump around the optimal solution and will never settle down.
- If η is too small, the training will eventually converge to the best state, but this will take a long time.
- To find a fairly good learning rate, the network should be trained by using various learning rates.



Choice of learning rate

- What is an appropriate choice for $\boldsymbol{\eta}$, which is neither too small nor too large?
- A common choice is $\boldsymbol{\eta} = \mathbf{1}$, leading to the simple weight change computational rule of $\boldsymbol{\Delta w} = \pm \boldsymbol{x}$,
  so that $(\boldsymbol{w} + \boldsymbol{\Delta w}).\boldsymbol{x} = \boldsymbol{w}.\boldsymbol{x} \pm \boldsymbol{x}.\boldsymbol{x}$
- If $|\boldsymbol{w}.\boldsymbol{x}| > |\boldsymbol{x}.\boldsymbol{x}|$, the sample $\boldsymbol{x}$ may not be correctly classified.
- In order to ensure that the sample $\boldsymbol{x}$ correctly classified, $(\boldsymbol{w} + \boldsymbol{\Delta w}).\boldsymbol{x}$ and $\boldsymbol{x}.\boldsymbol{x}$ have opposite signs.

$$|\Delta \boldsymbol{w} \cdot \boldsymbol{x}| > |\boldsymbol{w} \cdot \boldsymbol{x}|$$

$$\eta |\boldsymbol{x} \cdot \boldsymbol{x}| > |\boldsymbol{w} \cdot \boldsymbol{x}|$$

$$\eta > \frac{|\boldsymbol{w} \cdot \boldsymbol{x}|}{|\boldsymbol{x} \cdot \boldsymbol{x}|}$$

Content

- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Non-numeric inputs

- In some problems, the input dimensions are non-numeric.
- For example, input dimension may be ”color”.
- Its values may range over the set {red, blue, green, yellow}.
- We may not establish a relationships between colors on an axis.
- The simplest way is to generate four new dimensions (”red”, ”blue”, ”green”, ”yellow”).
- We can replace each original attribute-value pair by a binary vector.
- For instance, color = ”green” is represented by the input vector (0, 0, 1, 0), ”blue” is (0, 1, 0, 0).
- The disadvantage of this approach is a drastic increase in the number of dimensions.

Non-numeric inputs

Example

- The day of the week (Sunday/Monday/ . . .) is an important variable in predicting the amount of electric power consumed in a city.
- However, there is no obvious way of sequencing weekdays.
- So it is not appropriate to use a single variable whose values range from 1 to 7.
- Instead, seven different variables should be chosen and each input sample has a value of 1 for one of these coordinates, and a value of 0 for others.
- For instance, ”Tuesday” is represented as (0, 0, 1, 0, 0, 0, 0), ”Monday” is (0, 1, 0, 0, 0, 0, 0).

Content

- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Adalines

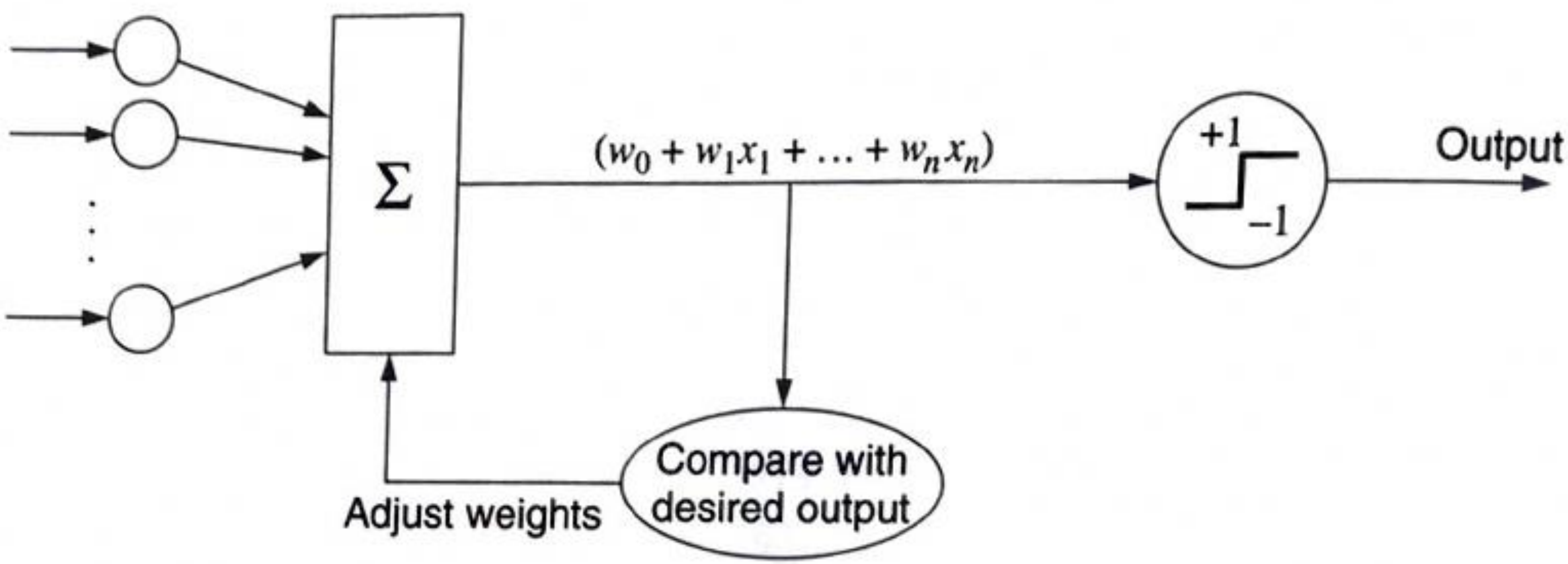
- The fundamental principle underlying the perceptron learning algorithm is to modify weights to reduce the number of misclassifications.
- Perfect classification using a linear element may not be possible for all problems.
- Minimizing the mean squared error (MSE) instead of the number of misclassified samples may be used while training.
- An adaptive linear element or Adaline, proposed by Widrow (1959, 1960), is a simple perceptron-like system.

Adalines

- Adaline accomplishes classification by modifying weights in such a way as to diminish the MSE at each iteration.
- This can be accomplished using gradient descent.
- MSE is a quadratic function whose derivative exists everywhere.
- Unlike the perceptron, this algorithm implies that weight changes are made to reduce MSE.
- Even when a sample is correctly classified by the network, the weights may change.

Adalines

- In the training process, when a sample is presented to the network, the linear weighted net input is computed.
- Computed net value is compared with the desired output.
- Generated error signal used to modify each weight in the Adaline.
- The weight change rule use partial derivative with respect to weights.



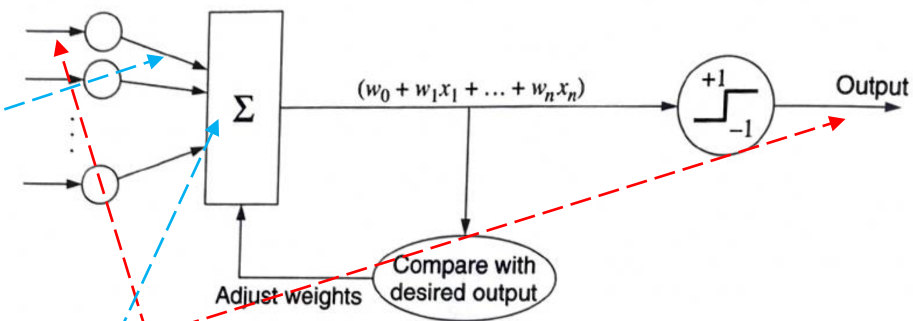
Adalines

- Let $\boldsymbol{i}_j = (i_0, i_1 \dots, i_n)$ be an input vector for which $\boldsymbol{d}_j$ is the desired output value.
- Let $\text{net}_j = \sum_l w_l i_l$ be the net input to the node.
- $\boldsymbol{w} = (w_0, \dots, w_n)$ is the presented value of the weight vector.
- The squared error is $E = (d_j - \text{net}_j)^2$

$$\frac{\partial E}{\partial w_k} = 2(d_j - \text{net}_j)\frac{\partial}{\partial w_k}(\text{-net}_j)$$

$$= (d_j - \text{net}_j)\frac{\partial}{\partial w_k}\left(-\sum_{l=0}^{n} w_l i_{l,j}\right)$$

$$= -(d_j - \text{net}_j)\, i_{k,j}.$$



- The weight update rule is

$$\Delta w_k = \eta\left(d_j - \sum_l w_l i_{l,j}\right) i_{k,j} i = \eta(d_j - net_j) i_j$$

Adalines

- Adaline Least-Mean-Squares (LMS) training algorithm

```
Algorithm LMS-Adaline;
  Start with a randomly chosen weight vector w_0;
  Let k = 1;
  while mean squared error is unsatisfactory and computational
  bounds are not exceeded, do
    Let i_j = (i_0, i_1, ..., i_n) be an input vector
    (chosen randomly or in some sequence)
    for which d_j is the desired output value, with i_0 = 1;
    Update the weight vector to
    w_k = w_{k-1} + η(d_j − Σ_l w_{k-1,l} i_l) i_j
    Increment k;
  end-while.
```

- The weight vector $\boldsymbol{w}$ is changed when the input vector $\boldsymbol{i}_j$ is presented to the Adaline.

Adalines

- A modification on this LMS rule has been made by Widrow and Hoff.
- The weight change magnitude independent of the magnitude of the input vector.
- α-LMS (or Widrow-Hoff delta rule) training rule is

$$\Delta w = \eta(d_j - \text{net}_j)\frac{i_j}{||i_j||}$$

where, d_j is the desired output for the jth input i_j ,
$||\ i\ ||$ denotes the length of vector i.

$$\text{net}_j = \sum_{l=0}^{n} w_l i_{l,j}$$

where, l is the length of the input vector.

Content

- Perceptrons
- Linear separability
- Perceptron training algorithm
- Termination criterion
- Choice of learning rate
- Non-numeric inputs
- Adalines
- Multiclass discrimination

Multiclass discrimination

- So far, we have considered dichotomies, or two-class problems.
- Many important real-life problems require partitioning data into three or more classes.
- For example, the character recognition problem consists of distinguishing between samples of 29 (for Turkish alphabet) different classes.
- A layer of perceptrons or Adalines may be used to solve some such multiclass problems.
- Four perceptrons can put together to solve a four-class classification problem.

Multiclass discrimination

- Each weight $w_{i,j}$ indicates the strength of the connection jth input to the ith node.
- A sample is considered to belong to the ith class if and only if the ith output $o_i = 1$, and every other output $o_k = 0$, for $k \neq i$.
- This network is trained in the same way as perceptrons.
- If all outputs are zeroes or if more than one output value equals 1, the network is considered to have failed in the classification task.
- All outputs can have values in between 0 and 1, a 'maximum-selector' can be used to select the highest-value output.

Homework

- Prepare a report on the use of artificial neural networks in the speech-to-text and text-to-speech applications.