Practical: 1

AIM: To understand the perceptron learning rule that can be applied for supervised learning of neural network.

Description/Program:
In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time.

Perceptrons are trained on examples of desired behavior. The desired behavior can be summarized by a set of input, output pairs

p1t1, p2t2, p3t3, p4t4...pntn

where **p** is an input to the network and **t** is the corresponding correct (target) output. The objective is to reduce the error **e**, which is the difference **t-a** between the neuron response **a**, and the target vector **t**. The perceptron learning rule calculates desired changes to the perceptron's weights and biases given an input vector **p**, and the associated error **e**. The target vector t must contain values of either **-1** or **1**, as perceptrons (with signum activation functions) can only output such values.

As each iteration goes on, the perceptron has a better chance of producing the correct outputs. The perceptron rule is proven to converge on a solution in a finite number of iterations if a solution exists.
If a bias is not used, learning algorithm works to find a solution by altering only the weight vector **w** to point toward input vectors to be classified as **1**, and away from vectors to be classified as **-1**. This results in a decision boundary that is perpendicular to **w**, and which properly classifies the input vectors.

There are three conditions that can occur for a single neuron once an input vector **p** is presented and the network's response **a** is calculated:

CASE 1. If an input vector is presented and the output of the neuron is correct (**a** = **t**, and **e** = **t** - **a** = **0**), then the weight vector **w** is not altered.
CASE 2. If the neuron output is **-1** and should have been **1** (**a** = **-1** and **t** = **1**, and **e** = **t** - **a** = **2**), the input vector **p** is added to the weight vector **w**. This makes the weight vector point closer to the input vector, increasing the chance that the input vector will be classified as a **1** in the future.
CASE 3. If the neuron output is **1** and should have been **-1** (**a** = **1** and **t** = **-1**, and **e** = **t** - **a** = **-2**), the input vector **p** is subtracted from the weight vector **w**. This makes the weight vector point farther away from the input vector, increasing the chance that the input vector is classified as a **-1** in the future

The perceptron learning rule can be written more succinctly in terms of the error **e** = **t** - **a**, and the change to be made to the weight vector **w**:

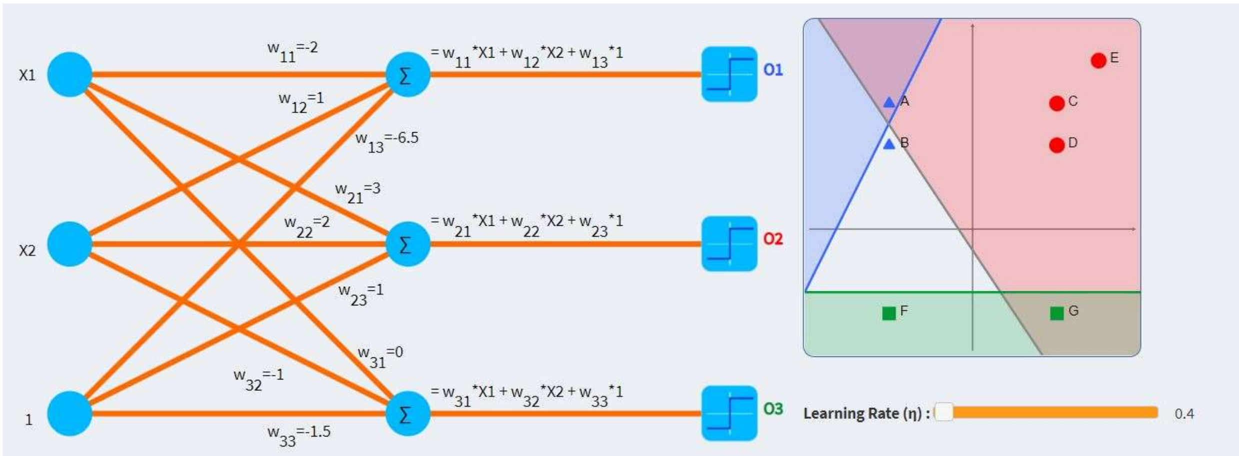
CASE 1. If **e** = **0**, then make a change in w equal to **0**.
CASE 2. If **e** = **1**, then make a change in w equal to $\mathbf{2p^T}$.
CASE 3. If **e** = **-1**, then make a change in w equal to $\mathbf{-2p^T}$.
According to **Perceptron** Learning Rule,

$$\mathbf{W_{new} = W_{old} + \Delta w}$$

w

here ($\Delta \mathbf{w} = \mathbf{e} * \mathbf{p}^T$) or $\Delta \mathbf{W_i} = \eta\ (\mathbf{D_i} - \mathbf{O_i})\ \mathbf{X}$

Execution:



Calculations:

$O = sgn(W \times X)$

$$O = sgn\left(\begin{bmatrix} -2 & 1 & -6.5 \\ 3 & 2 & 1 \\ 0 & -1 & -1.5 \end{bmatrix} X \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix} \right) = sgn\left(\begin{bmatrix} 0.5 \\ 1 \\ -4.5 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

$$O = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, D = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

According to **Perceptron Learning Rule** : $\Delta W_i = \eta (D_i - O_i) X$

Hence, $W_{i,new} = W_{i,old} + \eta (D_i - O_i) X$

The calculations for weight vector for each classifier neuron are as shown below:

For i = 1, $D_1 = 1$, $O_1 = 1$

$$W_{1,new} = \begin{bmatrix} -2 \\ 1 \\ -6.5 \end{bmatrix} + 0.4((1)-(1)) \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -2 \\ 1 \\ -6.5 \end{bmatrix}$$

In the carousel, the cards with green background indicate the corresponding **weight vector has not changed** whereas the cards with red background indicate the corresponding **weight vector has changed.**

Hence, the new weight vectors are (Refer to the carousel above) :

$W_{1,new} = \begin{bmatrix} -2 & 1 & -6.5 \end{bmatrix}$ $W_{2,new} = \begin{bmatrix} 4.6 & -0.4 & 0.2 \end{bmatrix}$ $W_{3,new} = \begin{bmatrix} 0 & -1 & -1.5 \end{bmatrix}$

Thus, the new weight matrix becomes:

$$\begin{bmatrix} -2 & 1 & -6.5 \\ 4.6 & -0.4 & 0.2 \\ 0 & -1 & -1.5 \end{bmatrix}$$

The weight matrix has **changed** and hence the graph will also change.

For i = 2, $D_2 = -1$, $O_2 = 1$

$$W_{2,new} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} + 0.4\,((-1) - (1)) \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.6 \\ -0.4 \\ 0.2 \end{bmatrix}$$

For i = 3, $D_3 = -1$, $O_3 = -1$

$$W_{3,new} = \begin{bmatrix} 0 \\ -1 \\ -1.5 \end{bmatrix} + 0.4\,((-1) - (-1)) \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -1.5 \end{bmatrix}$$

Calculations:

$O = \mathrm{sgn}(W \times X)$

$$O = \mathrm{sgn}\left(\begin{bmatrix} -2 & 1 & -6.5 \\ 4.6 & -0.4 & 0.2 \\ 0 & -1 & -1.5 \end{bmatrix} X \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix} \right) = \mathrm{sgn}\left(\begin{bmatrix} -0.5 \\ -9.8 \\ -3.5 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

$$O = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, D = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

According to **Perceptron Learning Rule** : $\Delta W_i = \eta\,(D_i - O_i)\,X$

Hence, $W_{i,new} = W_{i,old} + \eta\,(D_i - O_i)\,X$

The calculations for weight vector for each classifier neuron are as shown below:

For i = 1, $D_1 = 1$, $O_1 = -1$

$$W_{1,new} = \begin{bmatrix} -2 \\ 1 \\ -6.5 \end{bmatrix} + 0.4\,((1) - (-1)) \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -3.6 \\ 2.6 \\ -5.7 \end{bmatrix}$$

In the carousel, the cards with green background indicate the corresponding **weight vector has not changed** whereas the cards with red background indicate the corresponding **weight vector has changed.**

Hence, the new weight vectors are (Refer to the carousel above) :

$W_{1,new} = \begin{bmatrix} -3.6 & 2.6 & -5.7 \end{bmatrix}$ $W_{2,new} = \begin{bmatrix} 4.6 & -0.4 & 0.2 \end{bmatrix}$ $W_{3,new} = \begin{bmatrix} 0 & -1 & -1.5 \end{bmatrix}$

Thus, the new weight matrix becomes:

$$\begin{bmatrix} -3.6 & 2.6 & -5.7 \\ 4.6 & -0.4 & 0.2 \\ 0 & -1 & -1.5 \end{bmatrix}$$

The weight matrix has **changed** and hence the graph will also change.

For i = 2, D_2 = -1, O_2 = -1

$$W_{2,new} = \begin{bmatrix} 4.6 \\ -0.4 \\ 0.2 \end{bmatrix} + 0.4((-1)-(-1)) \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.6 \\ -0.4 \\ 0.2 \end{bmatrix}$$

For i = 3, D_3 = -1, O_3 = -1

$$W_{3,new} = \begin{bmatrix} 0 \\ -1 \\ -1.5 \end{bmatrix} + 0.4((-1)-(-1)) \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -1.5 \end{bmatrix}$$

Calculations:

O = sgn(W x X)

$$O = sgn\left(\begin{bmatrix} -3.6 & 2.6 & -5.7 \\ 4.6 & -0.4 & 0.2 \\ 0 & -1 & -1.5 \end{bmatrix} X \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \right) = sgn\left(\begin{bmatrix} -5.1 \\ 8.2 \\ -4.5 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

$$O = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, D = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

According to **Perceptron Learning Rule** : $\Delta W_i = \eta (D_i - O_i) X$

Hence, $W_{i,new} = W_{i,old} + \eta (D_i - O_i) X$

The calculations for weight vector for each classifier neuron are as shown below:

For i = 1, D_1 = -1, O_1 = -1

$$W_{1,new} = \begin{bmatrix} -3.6 \\ 2.6 \\ -5.7 \end{bmatrix} + 0.4((-1)-(-1)) \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -3.6 \\ 2.6 \\ -5.7 \end{bmatrix}$$

In the carousel, the cards with green background indicate the corresponding **weight vector has not changed** whereas the cards with red background indicate the corresponding **weight vector has changed.**

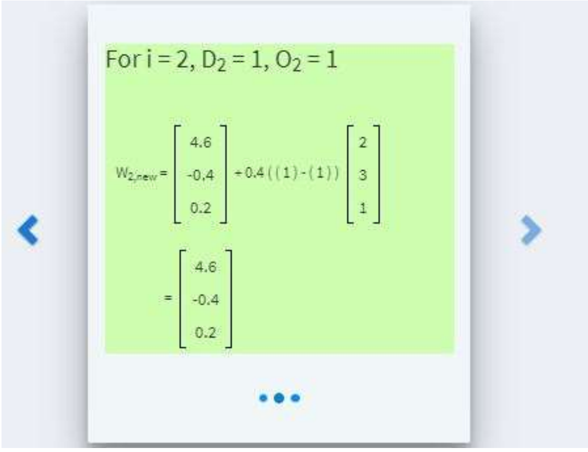
Hence, the new weight vectors are (Refer to the carousel above) :

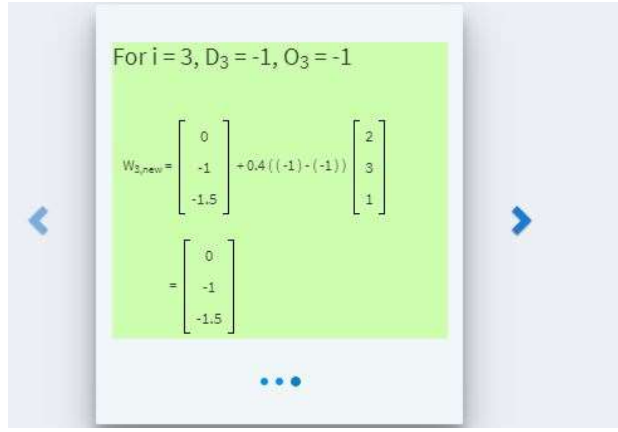
$W_{1,new} = \begin{bmatrix} -3.6 & 2.6 & -5.7 \end{bmatrix}$ $W_{2,new} = \begin{bmatrix} 4.6 & -0.4 & 0.2 \end{bmatrix}$ $W_{3,new} = \begin{bmatrix} 0 & -1 & -1.5 \end{bmatrix}$

Thus, the new weight matrix becomes:

$$\begin{bmatrix} -3.6 & 2.6 & -5.7 \\ 4.6 & -0.4 & 0.2 \\ 0 & -1 & -1.5 \end{bmatrix}$$

The weight matrix **hasn't changed** and hence the graph remains unchanged.


For i = 2, D_2 = 1, O_2 = 1
$W_{2,new} = \begin{bmatrix} 4.6 \\ -0.4 \\ 0.2 \end{bmatrix} + 0.4((1)-(1)) \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$
$= \begin{bmatrix} 4.6 \\ -0.4 \\ 0.2 \end{bmatrix}$


For i = 3, D_3 = -1, O_3 = -1
$W_{3,new} = \begin{bmatrix} 0 \\ -1 \\ -1.5 \end{bmatrix} + 0.4((-1)-(-1)) \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$
$= \begin{bmatrix} 0 \\ -1 \\ -1.5 \end{bmatrix}$

Practical: 2

AIM: Natural Language Processing- Word Analysis.

Description/Program:
Analysis of a word into root and affix(es) is called as Morphological analysis of a word. It is mandatory to identify root of a word for any natural language processing task. A root word can have various forms. For example, the word 'play' in English has the following forms: 'play', 'plays', 'played' and 'playing'. Hindi shows more number of forms for the word 'खेल' (khela) which is equivalent to 'play'. The forms of 'खेल'(khela) are the following:

खेल(khela), खेला(khelaa), खेली(khelii), खेलूंगा(kheluungaa), खेलूंगी(kheluungii), खेलेगा(khelegaa), खेलेगी(khelegii), खेलते(khelate), खेलती(khelatii), खेलने(khelane), खेलकर(khelakar)

For Telugu root ఆడడం (Adadam), the forms are the following::

Adutaanu, AdutunnAnu, Adenu, Ademu, AdevA, AdutAru, Adutunnaru, AdadAniki, Adesariki, AdanA, Adinxi, Adutunxi, AdinxA, AdeserA, Adestunnaru, ...

Thus we understand that the morphological richness of one language might vary from one language to another. Indian languages are generally morphologically rich languages and therefore morphological analysis of words becomes a very significant task for Indian languages.

Types of Morphology

Morphology is of two types,

1. Inflectional morphology

Deals with word forms of a root, where there is no change in lexical category. For example, 'played' is an inflection of the root word 'play'. Here, both 'played' and 'play' are verbs.
2. Derivational morphology

Deals with word forms of a root, where there is a change in the lexical category. For example, the word form 'happiness' is a derivation of the word 'happy'. Here, 'happiness' is a derived noun form of the adjective 'happy'.

Morphological Features:

All words will have their lexical category attested during morphological analysis. A noun and pronoun can take suffixes of the following features: gender, number, person, case
For example, morphological analysis of a few words is given below:

Language	input:word	output:analysis
Hindi	लडके (ladake)	rt=लड़का(ladakaa), cat=n, gen=m, num=sg, case=obl
Hindi	लडके (ladake)	rt=लड़का(ladakaa), cat=n, gen=m, num=pl, case=dir
Hindi	लड़कों (ladakoM)	rt=लड़का(ladakaa), cat=n, gen=m, num=pl, case=obl
English	boy	rt=boy, cat=n, gen=m, num=sg
English	boys	rt=boy, cat=n, gen=m, num=pl

A verb can take suffixes of the following features: tense, aspect, modality, gender, number, person

Languageinput:wordoutput:analysis

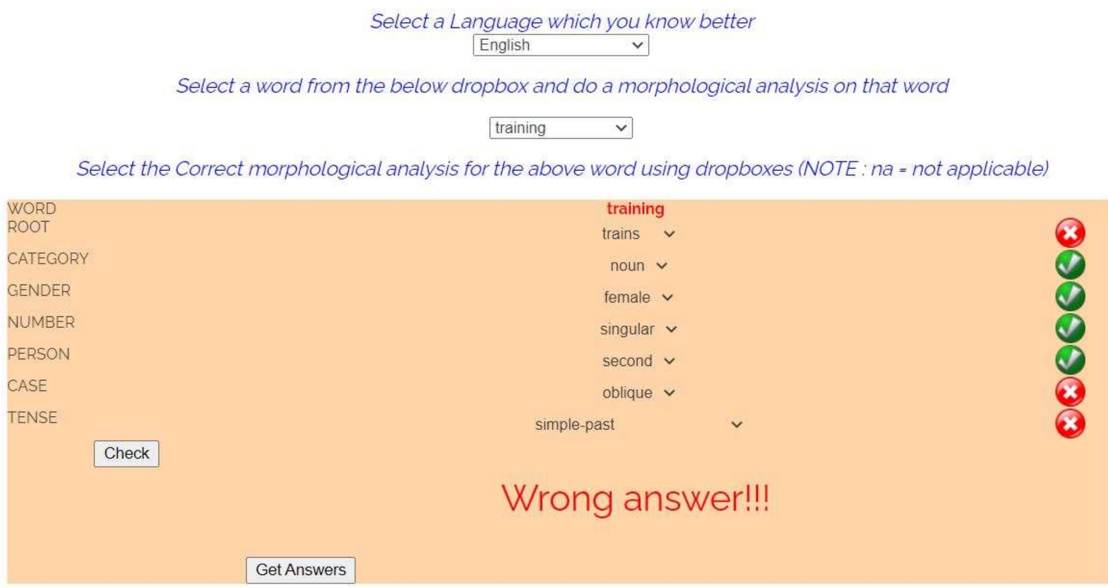
Hindi	हँसी(hansii)	rt=हँस(hans), cat=v, gen=fem, num=sg/pl, per=1/2/3 tense=past, aspect=pft
English	toys	rt=toy, cat=n, num=pl, per=3

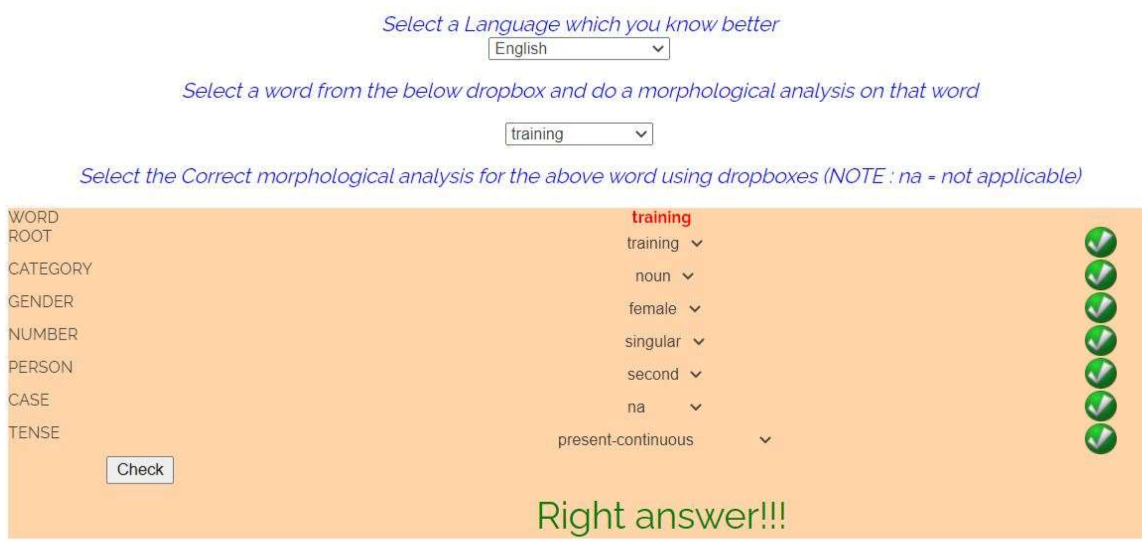
'rt' stands for root. 'cat' stands for lexical category. Thev value of lexicat category can be noun, verb, adjective, pronoun, adverb, preposition. 'gen' stands for gender. The value of gender can be masculine or feminine.
'num' stands for number. The value of number can be singular (sg) or plural (pl).
'per' stands for person. The value of person can be 1, 2 or 3

The value of tense can be present, past or future. This feature is applicable for verbs.
The value of aspect can be perfect (pft), continuous (cont) or habitual (hab). This feature is not applicable for verbs.

'case' can be direct or oblique. This feature is applicable for nouns. A case is an oblique case when a postposition occurs after noun. If no postposition can occur after noun, then the case is a direct case. This is applicable for hindi but not english as it doesn't have any postpositions. Some of the postpsitions in hindi are: का(kaa), की(kii), के(ke), को(ko), में(meM).

Execution:


Select a Language which you know better
English
Select a word from the below dropbox and do a morphological analysis on that word
training
Select the Correct morphological analysis for the above word using dropboxes (NOTE : na = not applicable)
WORD
training
ROOT
trains
CATEGORY
noun
GENDER
female
NUMBER
singular
PERSON
second
CASE
oblique
TENSE
simple-past
Check
Wrong answer!!!
Get Answers


Select a Language which you know better
English
Select a word from the below dropbox and do a morphological analysis on that word
training
Select the Correct morphological analysis for the above word using dropboxes (NOTE : na = not applicable)
WORD
training
ROOT
training
CATEGORY
noun
GENDER
female
NUMBER
singular
PERSON
second
CASE
na
TENSE
present-continuous
Check
Right answer!!!