RGPVNOTES.IN

Please do not share these notes on apps like WhatsApp or Telegram.

The revenue we generate from the ads we show on our website and app funds our services. The generated revenue **helps us prepare new notes and improve the quality of existing study materials**, which are available on our website and mobile app.

If you don't use our website and app directly, it will hurt our revenue, and we might not be able to run the services and **have to close them**. So, it is a humble request for all to **stop sharing the study material** we provide on various apps. Please **share the website's URL instead.**

Subject Name: Soft Computing **Subject Code: IT 701**

Subject Notes

Syllabus:

Unsupervised learning: Introduction, Fixed weight competitive nets, Kohonen SOM, Counter Propagation networks, (Theory, Architecture, Flow Chart, Training Algorithm and applications). Introduction to Convolutional neural networks (CNN) and Recurrent neural networks (RNN).

Unit-3

Course Objectives

1. The objective of this course is to understand Unsupervised learning and its type.
2. To help student to understand Convolutional neural network and Recurrent neural network.

Unsupervised Learning: Introduction

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

In ANNs following unsupervised learning, the input vectors of similar type are grouped without the use of training data to specify how a member of each group looks or to which group a number belongs. In the training process, the network receives the input patterns and organizes these patterns to form clusters. When a new input pattern is applied, the neural network gives an output response indicating the class to which the input pattern belongs.If for an input, a pattern class cannot be found then a new class is generated.

These are called unsupervised learning because unlike supervised learning above there is no correct answer and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

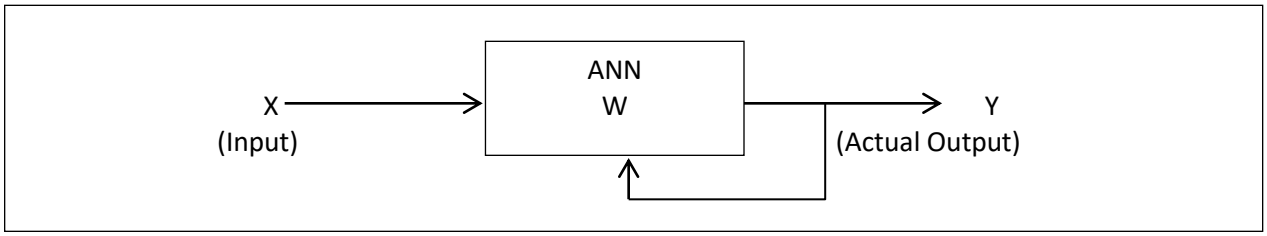

Figure 3.1: Unsupervised Learning

From the working of unsupervised learning it is clear that there is no feedback from the environment to inform what the outputs should be or whether the outputs are correct. In this case the network must itself discover patterns, regularities, features or categories from the input data and relations for the input data over the output.

Fixed weight competitive nets

Many neural nets use the idea of competition among neurons to enhance the contrast in activations of the neurons. In the most extreme situation, often called Winner-Take-All, only the neuron with the largest activation is allowed to remain "on." Typically. The neural implementation of this competition is not specified (and in computer simulations, the same effect can be achieved by a simple. non-neural sorting process). **A Hamming network** is a fixed-weight competitive ANN where the lower network feeds into a competitive network called maxnet.

Hamming Network

In most of the neural networks using unsupervised learning, it is essential to compute the distance and perform comparisons. This kind of network is Hamming network, where for every given input vectors, it would be clustered into different groups. Following are some important features of Hamming Networks −

1. Lippmann started working on Hamming networks in 1987.
2. It is a single layer network.
3. The inputs can be either binary {0, 1} of bipolar {-1, 1}.
4. The weights of the net are calculated by the exemplar vectors.
5. It is a fixed weight network which means the weights would remain the same even during training.

Max Net

This is also a fixed weight network, which serves as a subnet for selecting the node having the highest input. All the nodes are fully interconnected and there exists symmetrical weights in all these weighted interconnections.

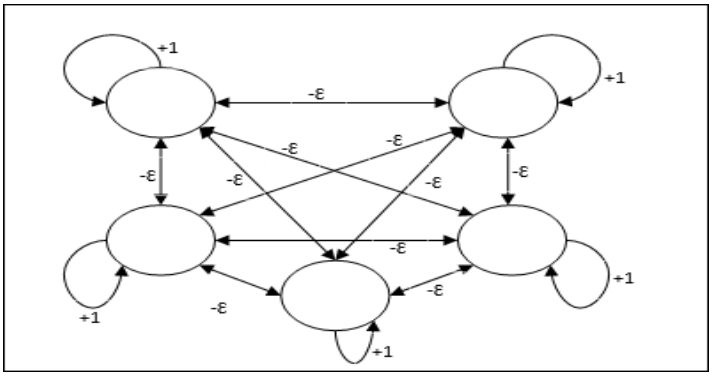

Figure 3.2: Architecture of Max Net

It uses the mechanism which is an iterative process and each node receives inhibitory inputs from all other nodes through connections. The single node whose value is maximum would be active or winner and the activations of all other nodes would be inactive. Max Net uses identity activation function with:

$$f(x) = \begin{cases} x & if\ x > 0 \\ 0 & if\ x \leq 0 \end{cases}$$

The task of this net is accomplished by the self-excitation weight of +1 and mutual inhibition magnitude, which is set like [0 < ε < 1/m] where **"m"** is the total number of the nodes.

Competitive Learning in ANN

It is concerned with unsupervised training in which the output nodes try to compete with each other to represent the input pattern. To understand this learning rule we will have to understand competitive net which is explained as follows:

Basic Concept of Competitive Network

This network is just like a single layer feed-forward network having feedback connection between the outputs. The connections between the outputs are inhibitory type, which is shown by dotted lines, which means the competitors never support themselves.

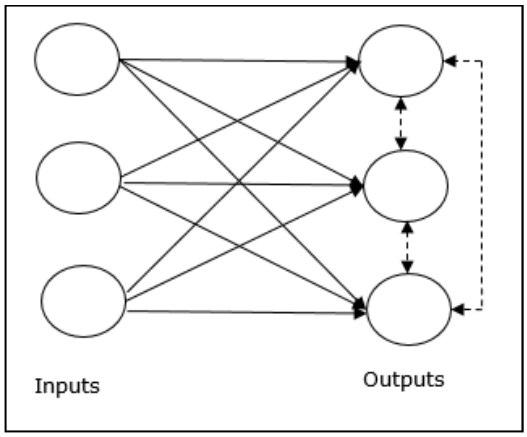

Figure 3.3: Single layer feed-forward network

Basic Concept of Competitive Learning Rule

As said earlier, there would be competition among the output nodes so the main concept isduring training, the output unit that has the highest activation to a given input pattern, will be declared the winner. This rule is also called Winner-takes-all because only the winning neuron is updated and the rest of the neurons are left unchanged.

Mathematical Formulation

Following are the three important factors for mathematical formulation of this learning rule –

- Condition to be a winner

  Suppose if a neuron y_k wants to be the winner, then there would be the following condition

$$y_k = \begin{cases} 1 & if\ v_k > v_j\ for\ all\ \ j,\ j \neq k \\ 0 & otherwise \end{cases}$$

  It means that if any neuron, say, y_k wants to win, then its induced local field the output of the summation unit say v_k, must be the largest among all the other neurons in the network.

- Condition of the sum total of weight

  Another constraint over the competitive learning rule is the sum total of weights to a particular output neuron is going to be 1. For example, if we consider neuron **k** then

$$\sum_k w_{kj} = 1 \quad for\ all\ k$$

- Change of weight for the winner

  If a neuron does not respond to the input pattern, then no learning takes place in that neuron. However, if a particular neuron wins, then the corresponding weights are adjusted as follows –

$$\Delta w_{kj} = \begin{cases} -\alpha(x_j - w_{kj}), & if\ neuron\ k\ wins \\ 0 & if\ neuron\ k\ losses \end{cases}$$

  Here α is the learning rate.

  This clearly shows that we are favoring the winning neuron by adjusting its weight and if a neuron is lost, then we need not bother to re-adjust its weight.

Kohonen Self-Organizing Map(Theory, Architecture, Flow Chart, Training Algorithm):

The Self-Organizing Map is one of the most popular neural network models. It belongs to the category of competitive learning networks. The Self-Organizing Map is based on unsupervised learning, which means that no human intervention is needed during the learning and that little needs to be known about the characteristics of the input data. The SOM can be used to detect features inherent to the problem and thus has also been called SOFM, the Self-Organizing Feature Map.

The Self-Organizing Map was developed by professor Kohonen. The SOM has been proven useful in many applications. The SOM algorithm is based on unsupervised, competitive learning. It provides a topology preserving mapping from the high dimensional space to map units. Map units, or neurons, usually form a two-dimensional lattice and thus the mapping is a mapping from high dimensional space onto a plane. The property of topology preserving means that the mapping preserves the relative distance between the points. Points that are near each other in the input space are mapped to nearby map units in the SOM. The SOM can thus serve as a cluster analyzing tool of high-dimensional data. Also, the SOM has the capability to generalize. Generalization capability means that the network can recognize or characterize inputs it has never encountered before. A new input is assimilated with the map unit it is mapped to.

Architecture of Kohonen SOM:

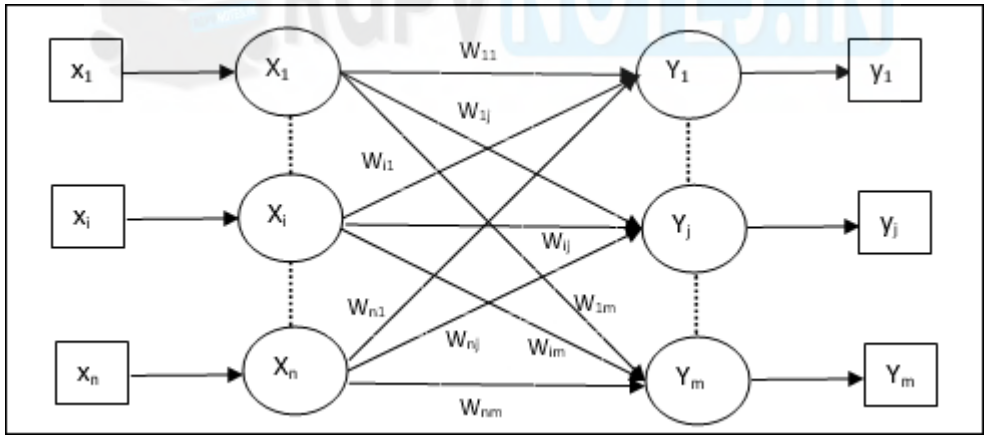

Figure 3.4: Architecture of Kohonen SOM

The architecture consists of two layers: input layer and output layer (cluster). There are “n” units in the input layer and “m” units in the output layer. Basically, here the winner unit is identified by using either dot product or Euclidean distance method and the weight updation using Kohonen learning rules is performed over the winning cluster unit.

Flowchart:

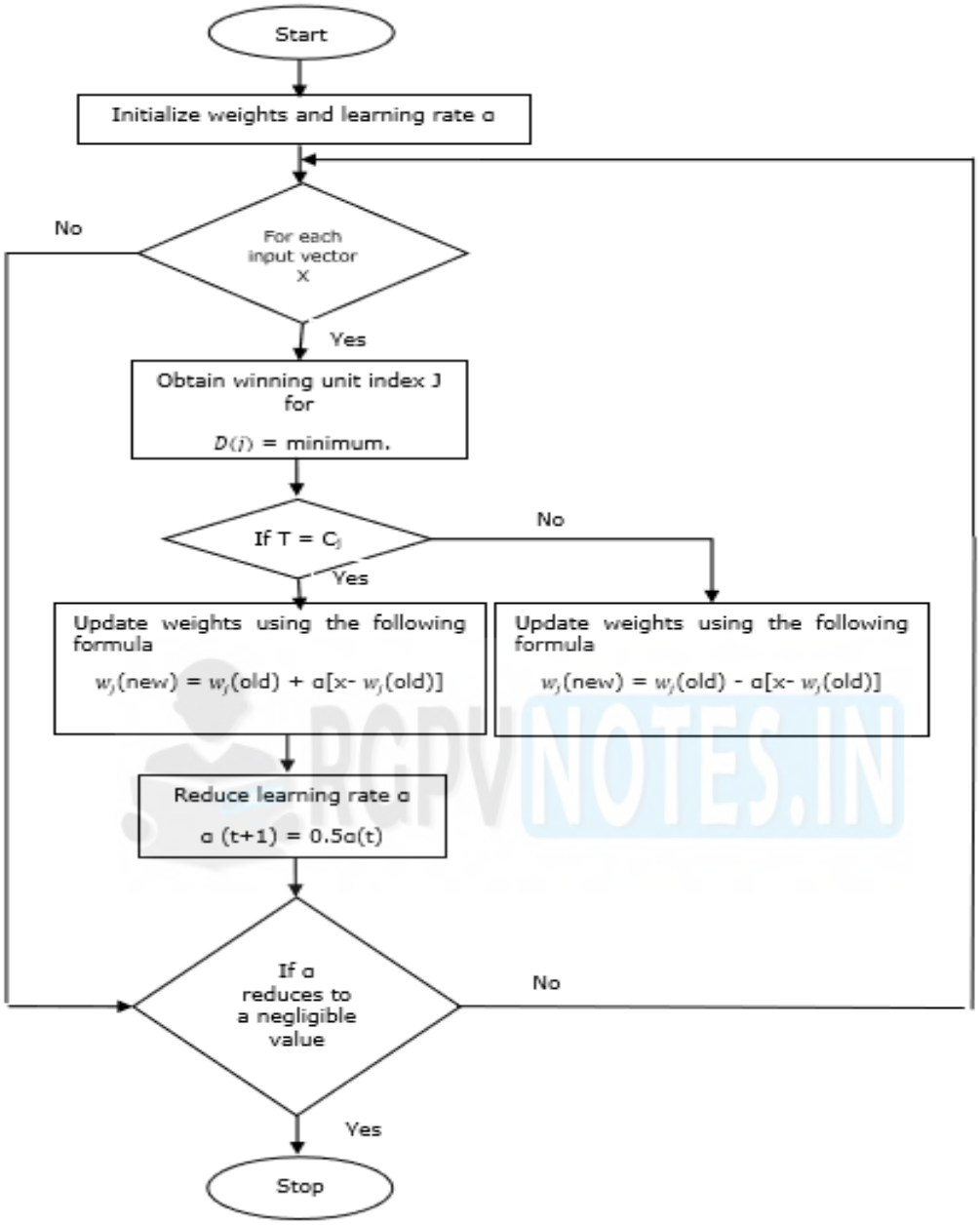

Figure 3.5: Flowchart of Kohonen SOM

Training Algorithm:

Step 1 − Initialize reference vectors, which can be done as follows −

- **Step 1**a − From the given set of training vectors, take the first "**m**" number of clusters training vectors and use them as weight vectors. The remaining vectors can be used for training.
- **Step 1** b − Assign the initial weight and classification randomly.
- **Step 1** c − Apply K-means clustering method.

Step 2 − Initialize reference vector α

Step 3 − Continue with steps 4-9, if the condition for stopping this algorithm is not met.

Step 4 − Follow steps 5-6 for every training input vector **x**.

Step 5 − Calculate Square of Euclidean Distance for **j = 1 to m** and **i = 1 to n**

$$D(j) = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_i - w_{ij})^2$$

Step 6 − Obtain the winning unit **J** where **D**j is minimum.

Step 7 − Calculate the new weight of the winning unit by the following relation –

if **T = C**$_j$then

$$w_j(new) = w_j(old) + \alpha[x - w_j(old)]$$

if **T ≠ C**$_j$ then

$$w_j(new) = w_j(old) - \alpha[x - w_j(old)]$$

Step 8 − Reduce the learning rate α.

Step 9 − Test for the stopping condition. It may be as follows −

- Maximum number of epochs reached.
- Learning rate reduced to a negligible value.

Counter Propagation Network:

Counter propagation networks were proposed by Hecht Nielsen in 1987. They are multilayer networks based on the combinations of the input, output and clustering layers. The application of counter propagation nets are data compression, function approximation and pattern association. The counterpropagation network is basically constructed from an instar-outstar model. This model is a three layer neural network that performs input-output data mapping producing an output vector y in response to an input vector x, on the basis of competitive learning.

A counterpropagation net is an approximation of its training input vector pairs by adaptively constructing a look-up-table. By this method, several data points can be compressed to a more manageablenumber of look-up-table entries. The accuracy of the function approximation and data compression is basedon the number of entries in the look-up-table, which equals the number of units in the cluster layer of the net.

There are two stages involved in the training process of a counterpropagation net. The input vectors areclustered in the first stage. Originally, tr is assumed that there is no topology included in the counter propagation network. However, on the inclusion of a linear topology,

the performance of the net can be improved. The clusters are performed using Euclidean distance method or dot product method. In the second stageof training, the weights from the cluster layer units to the output units are tuned to obtain the desired response.
There are twotypesofcounterpropagation nets: (i) Fullcounterpropagation net and (ii) forward-onlycounterpropagation net.

Flow Chart:

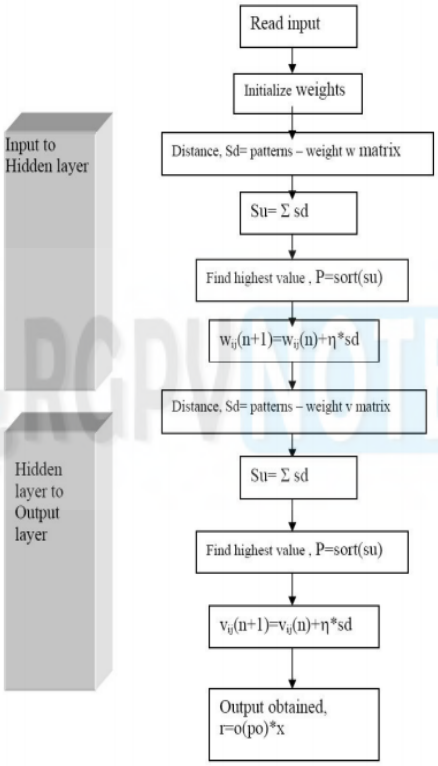

Figure 3.6 Flowchart of CPN

This figure shows three layers for CPN; an input layer that reads input patterns from the training set and forwards them to the network, a hidden layer that works in a competitive fashion and associates each input pattern with one of the hidden units and the output layer which is trained via a teaching algorithm that tries to minimize the mean square error (MSE) between the actual network output and the desired output associated with the current input vector. In some cases, a fourth layer is used to normalize the input vectors, but this normalization can be easily performed by the application before these vectors are sent to the Kohonen layer. Regarding the training process of the counter-propagation network, it can be described as a two-stage procedure; in the first stage, the process updates the weights of the synapses between the input and the Kohonen layer, while in the second stage the weights of the synapses between the Kohonen and the Grossberg layer are updated.

Applications:

The applications of counter propagation network are:

1. Datacompression specially for Image and audio
2. Function approximation
3. Pattern association

Full Counterpropagation Net:

Full counterpropagarion net (full CPN) efficiently represents a large number of vector pairs x:y by adaptivelyconstructing a look-up-table. The approximation here is x*:y*, which is based on the vector pairs x:y, possiblywith some distorted or missing elements in either vector or both vectors. The nerwork is defined to approximatea continuous function f, defined on a compact set A. The full CPN works best if the inverse function exists and is continuous.

The vectors x and y propagate through the network in a counterflow manner toyield output vectors x* and y*, which are the approximations of x andy, respectively. During competition,the winner can be determined either by Euclidean distance or by dot product method. In case of dot productmethod, the one with the largest net input is the winner. Whenever vectors are to be compared using thedot product metric, they should be normalized. Even though the normalization can be performed withoutloss of information by adding an extra component, yet to avoid the complexicy Euclidean distance methodcan be used. On the basis of this, direct comparison can be made between the full CPN and forward-onlyCPN.

For continuous function, the CPN is efficient as the back-propagation net; it is a universal continuousfunction approximator. In case of CPN, the number of hidden nodes required to achieve a particular levelof accuracy is greater than the number required by the back-propagation network. The greatest appeal ofCPN is its speed of learning. Compared to various mapping networks, ir requires only fewer steps of trainingto achieve best performance. This is common for any hybrid learning method that combines unsupervisedlearning (e.g., instar learning) and supervised learning (e.g., outsrar learning).

As already discussed, the training ofCPN occurs in two phases. In the input phase, the units in the clusterlayer and input layer are found to be active. In CPN, no topology is assumed for the cluster layer units; onlythe winning units are allowed to learn. The weight updarion learning rule on the winning cluster units is

$$V_{ij}(new) = V_{ij}(old) + \alpha [x_i - V_{ij}(old)], \quad i=1 \text{ to } n$$

$$W_{kj}(new) = W_{kj}(old) + \beta (y_k - W_{kj}(old)], \quad k=1 \text{ to } n$$

The above is standard Kohonen learning which consists of competition among the units and selection of
winner unit. The weight updation is performed for the winning unit.

Architecture:

The four major components of rhe instar-outstar model are the input layer, the instar, the competitive layerand the outstar. For each node i in the input layer, there is an input value xi. An

instar responds maximally tothe input vectors from a particular cluster. All the instars are grouped into a layer called the competitive layer.Each of the instar responds maximally to a group of input vectors in a different region of space. This layer ofinstars classifies any input vector because, for a given input, the winning instar with the strongest responseidentifies the region of space in which the input vector lies. Hence, it is necessary that the competitive layersingle outs the winning instar by setting its output ro a nonzero value and also suppressing the other outputs to zero. That is, it is a winner-take-all or a Maxnet-type network. An outstar model is found to have all thenodes in the output layer and a single node in the competitive layer. The outstar looks like the fan-out ofa node.

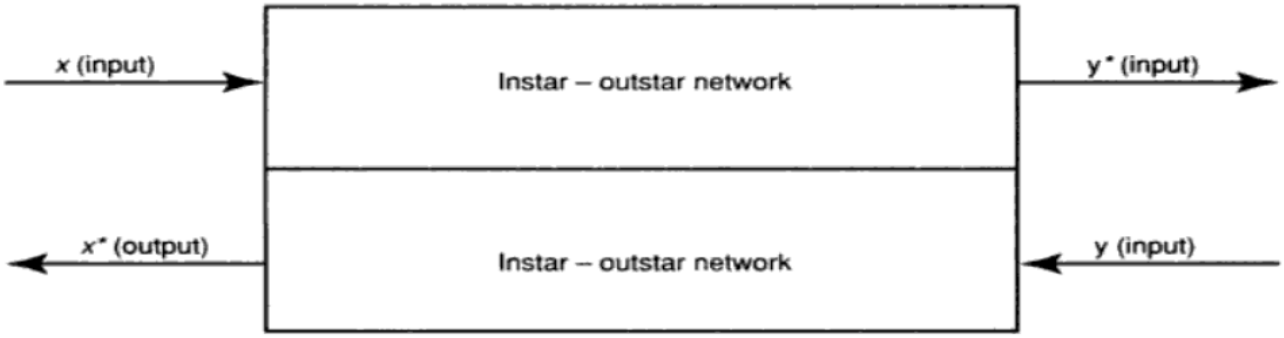

Figure 3.7: General Architecture of Full CPN

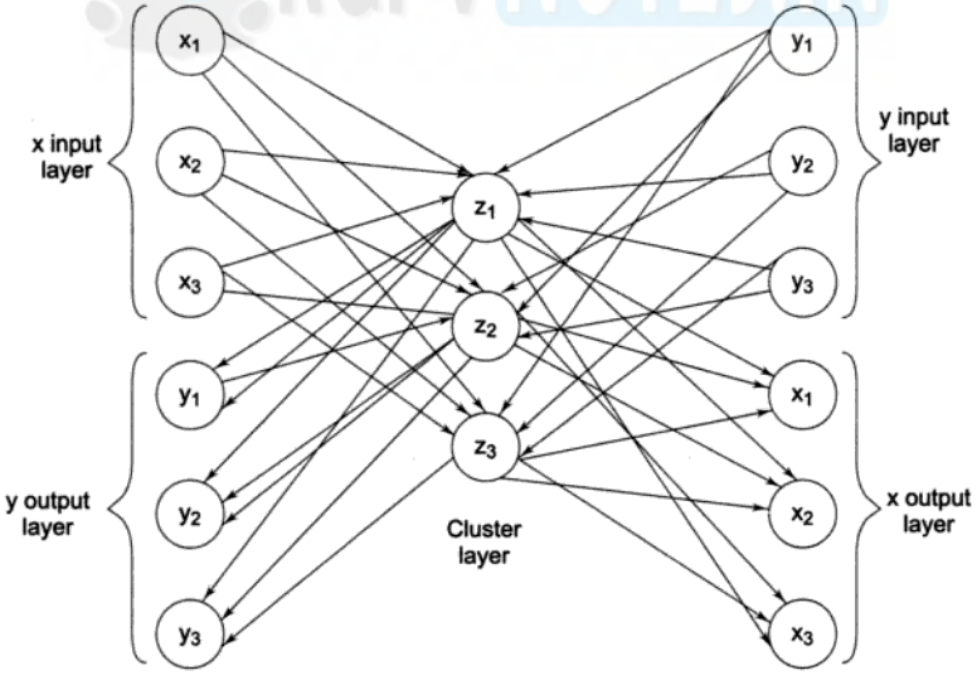

Figure 3.8: Architecture of Full CPN

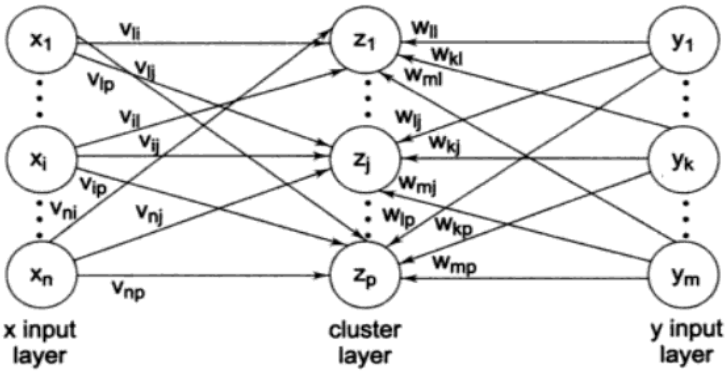

Figure 3.9: First Phase of Training of Full CPN

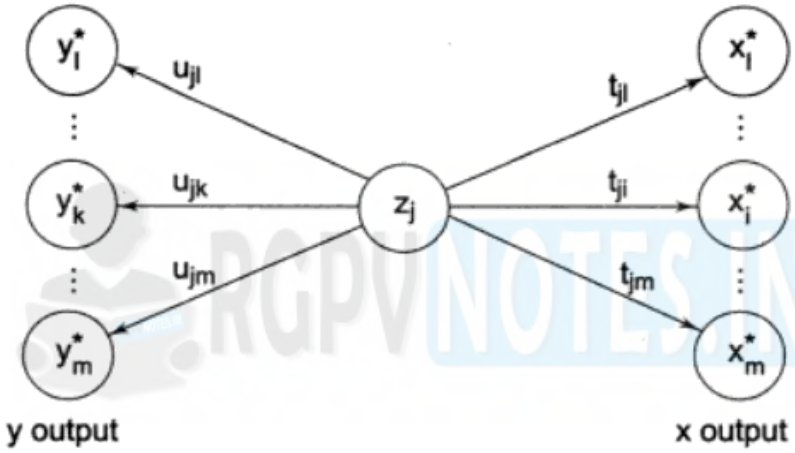

Figure 3.10: Second Phase of Training of Full CPN

Forward-Only Counterpropagation Network:

A simplified version of full CPN is the forward-only CPN. The approximation of the function y = f(x) butnot of x = f(y) can be performed using forward-only CPN, i.e., it may be used if the mapping from x toyis well defined but mapping from y to x is not defined. In forward-only CPN only the x-vectors are used to form the clusters on the Kohonen units. Forward-only CPN uses only the x vectors to form the clusters onthe Kohonen units during first phase of training.

In case of forward-only CPN, first input vectors are presented to the input units. The cluster layer unitscompete with each other using winner-take-all policy to learn the input vector. Once entire set of trainingvecrors has been presented, there exist reduction in learning rate and the vectors are presented again, performingseveral iterations. First, the weights between the input layer and cluster layer are trained. Then the weightsbetween the cluster layer and output layer are trained. This is a specific competitive network, with targetknown. Hence, when each input vector is presented to the input vector, its associated target vectors arepresented to the output layer. The winning cluster unit sends its signal to the output layer. Thus each ofthe output unit has a computed signal (Wjk) and the

target value {Yk}. The difference between these values iscalculated; based on this, the weights between the winning layer and output layer are updated.

Architecture:

It consists of three layers: input layer, cluster(competitive) layer and output layer. The architecture of forward-only CPN resembles the back-propagationnetwork, but in CPN there exists interconnections becween the units in the cluster layer (which are not connected). Once competition is completed in a forward-only CPN, only one unit will beactive in that layer and it sends signal to the output layer. As inputs are presented to the network, the desiredoutputs will also be presented simultaneously.

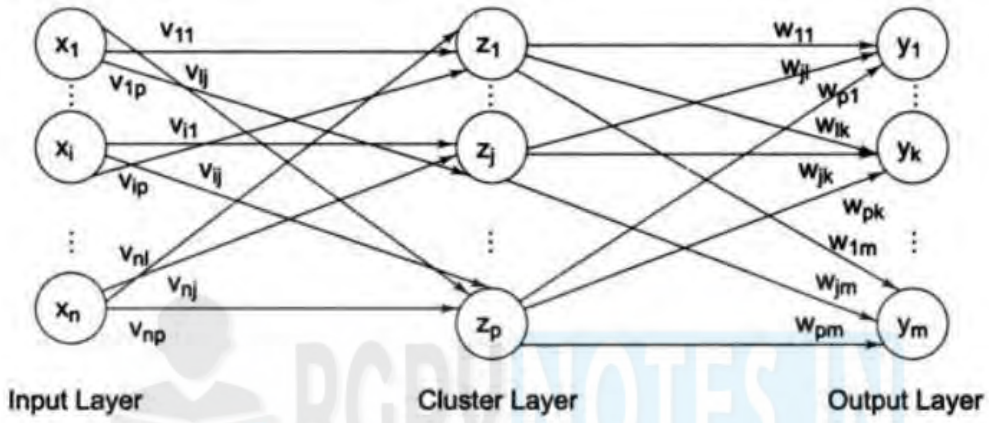

Figure 3.11: Architecture of Forward Only CPN

Introduction to Convolutional neural networks (CNN)

A convolutional neural network (CNN) is a particular implementation of a neural network used in machine learning that exclusively processes array data such as images, and is frequently used in machine learning applications targeted at medical images. For image classification we use Convolution Neural Network.CNN is a neural network with a special structure that was designed as a model of a human vision system (HVS). Thus, CNNs are most suitable for solving problems of computer vision, such as object recognition and classification of images and video data. They have also been used successfully for speech recognition and text translation.

Convolution Neural Networks or covnets are neural networks that share their parameters.It can be represented as a cuboid having its length, width (dimension of the image) and height (as image generally have red, green, and blue channels).Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different width, height, and depth. Instead of just R, G and B channels now we have more channels but lesser width and height, this operation is called Convolution. If patch size is same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.Now let's talk about a bit of mathematics which is involved in the whole convolution process:

- Convolution layers consist of a set of learnable filters (patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimension 34x34x3. Possible size of filters can be a * a * 3, where 'a' can be 3, 5, 7, etc but small as compared to image dimension.
- During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
- As we slide our filters we will get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

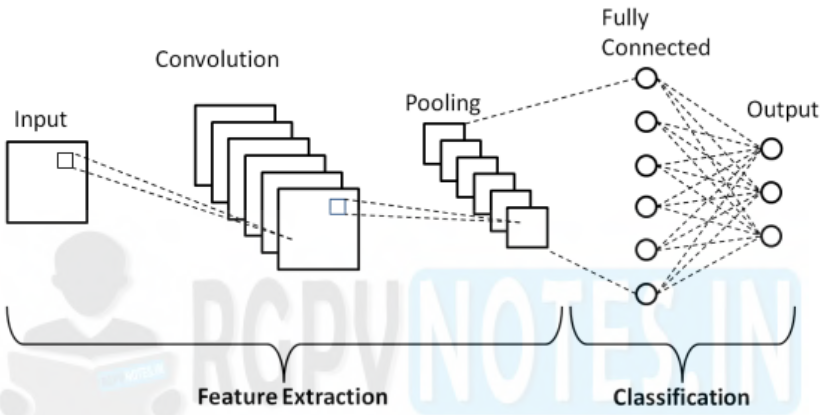

Figure 3.12 Basic convolutional neural network (CNN) architecture

Layers used to build Convolutional neural networks

A Convolutional neural networks is a sequence of layers, and every layer transforms one volume to another through differentiable function.

Types of layers:

Take an example by running a Convolutional neural network on of image of dimension 32 x 32 x 3.

1. **Input Layer:** This layer holds the raw input of image with width 32, height 32 and depth 3.
2. **Convolution Layer:** This layer computes the output volume by computing dot product between all filters and image patch. Suppose we use total 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12.
3. **Activation Function Layer:** This layer will apply element wise activation function to the output of convolution layer. Some common activation functions are RELU: max(0, x), Sigmoid: 1/(1+e^-x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimension 32 x 32 x 12.
4. **Pool Layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents from overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

5. **Fully-Connected Layer:** This layer is regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

Training process of the Convolution Network:

Step1: We initialize all filters and parameters / weights with random values
Step2: The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.

- Lets say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]
- Since weights are randomly assigned for the first training example, output probabilities are also random.

Step3: Calculate the total error at the output layer (summation over all 4 classes)

Total Error = ∑ ½ (target probability – output probability) ²

Step4: Use Backpropagation to calculate the *gradients* of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.

- The weights are adjusted in proportion to their contribution to the total error.
- When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].
- This means that the network has *learnt* to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.
- Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process - only the values of the filter matrix and connection weights get updated.

Step5: Repeat steps 2-4 with all images in the training set.

Recurrent Neural Network

Recurrent Neural Network(RNN) is a type of Neural Network where the **output from previous step are fed as input to the current step**. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is **Hidden state**, which remembers some information about a sequence.RNN have a **"memory"** which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

How Does Recurrent Neural Network work?

Just like traditional Artificial Neural Networks, RNN consists of nodes with three distinct layers representing different stages of the operation.

- The nodes represent the “Neurons” of the network.
- The neurons are spread over the temporal scale (i.e., sequence) separated into three layers.

The layers are:

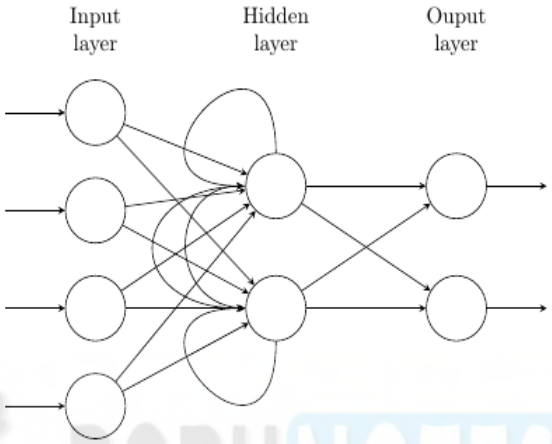

Figure 3.13: Layers of Recurrent Neural Network

1. Input layer represents information to be processed;
2. A hidden layer represents the algorithms at work;
3. Output layer shows the result of the operation;

Hidden layer contains a temporal loop that enables the algorithm not only to produce an output but to feed it back to itself.

This means the neurons have a feature that can be compared to short-term memory. The presence of the sequence makes them to “remember” the state (i.e., context) of the previous neuron and pass that information to themselves in the “future” to further analyze data.

Overall, the RNN neural network operation can be one of the three types:

1. One input to multiple outputs - as in image recognition, image described with words;
2. Several contributions to one output - as in sentiment analysis, where the text is interpreted as positive or negative;
3. Many to many - as in machine translation, where the word of the text is translated according to the context they represent as a whole;

The key algorithms behind RNN are:

1. Backpropagation Through Time to classify sequential input- linking one-time step to the next
2. Vanishing/Exploding gradients - to preserve the accuracy of the results
3. Long Short-Term Memory Units - to recognize the sequences in the data

RGPVNOTES.IN

Thank you for using our services. Please support us so that we can improve further and help more people.
https://www.rgpvnotes.in/support-us

If you have questions or doubts, contact us on
WhatsApp at +91-8989595022 or by email at hey@rgpvnotes.in.

For frequent updates, you can follow us on
Instagram: https://www.instagram.com/rgpvnotes.in/.