Dimensionality Reduction

1. Principal Component Analysis:

It is a statistical procedure that uses an orthogonal transformation which converts a set of correlated variables to a set of uncorrelated variables. PCA is a most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

1.1 **Steps Involved in PCA**

1. Standardize the data. (with mean =0 and variance = 1)
2. Compute the Covariance matrix of dimensions.
3. Obtain the Eigenvectors and Eigenvalues from the covariance matrix (we can also use correlation matrix or even Single value decomposition, however in this post will focus on covariance matrix).
4. Sort eigen values in descending order and choose the top k Eigenvectors that correspond to the k largest eigen values (k will become the number of dimensions of the new feature subspace k≤d, d is the number of original dimensions).
5. Construct the projection matrix W from the selected k Eigenvectors.
6. Transform the original data set X via W to obtain the new k-dimensional feature subspace Y.

1.2 Implementation and Demonstration:

1. Standardization

When there are different scales used for the measurement of the values of the features, then it is advisable to do the standardization to bring all the feature spaces with mean = 0 and variance = 1.

The reason why standardization is very much needed before performing PCA is that PCA is very sensitive to variances. Meaning, if there are large differences between the scales (ranges) of the features, then those with larger scales will dominate over those with the small scales. So, transforming the data to the same scales will prevent this problem. That is where we use standardization to bring the features with mean value 0 and variance 1.

$$\textit{Standardized value of } x_i = \frac{x_i - \textit{mean of } x}{\textit{std Deviation of } x}$$

2. **Eigen decomposition — Computing Eigenvectors and Eigenvalues**

The eigenvectors and eigen values of a covariance (or correlation) matrix represent the “core” of a PCA:

- The Eigenvectors (principal components) determine the directions of the new feature space, and the eigenvalues determine their magnitude.
- In other words, the eigenvalues explain the variance of the data along the new feature axes. It means the corresponding eigenvalue tells us that how much variance is included in that new transformed feature.
- To get eigenvalues and Eigenvectors we need to compute the covariance matrix. So in the next step let's compute it.

3. Selecting The Principal Components

- The typical goal of a PCA is to reduce the dimensionality of the original feature space by projecting it onto a smaller subspace, where the eigenvectors will form the axes.
- However, the eigenvectors only define the directions of the new axis, since they have all the same unit length 1.

4. Construct the projection matrix W from the selected k eigenvectors

- Projection matrix will be used to transform the Iris data onto the new feature subspace or we say newly transformed data set with reduced dimensions.
- It is a matrix of our concatenated top k Eigenvectors.

2. Artificial Neural Networks:

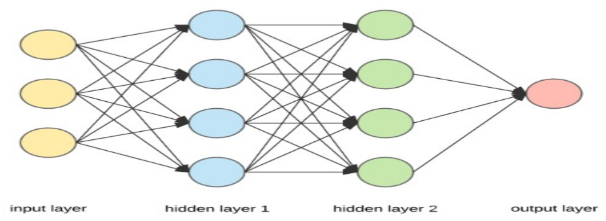
"Artificial Neural Networks or ANN is an information processing paradigm that is inspired by the way the biological nervous system such as brain process information. It is composed of large number of highly interconnected processing elements(neurons) working in unison to solve a specific problem."

2.1 NEURAL NETWORK REPRESENTATIONS

Artificial Neural Network is computing system inspired by biological neural network that constitute animal brain. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

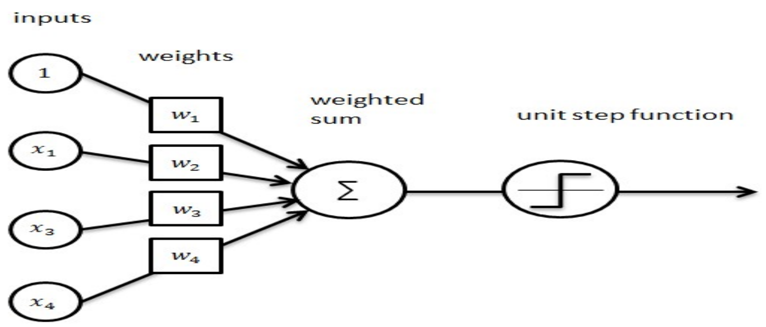
The Neural Network is constructed from 3 type of layers:

1. Input layer — initial data for the neural network.
2. Hidden layers — intermediate layer between input and output layer and place where all the computation is done.
3. Output layer — produce the result for given inputs.



There are 3 yellow circles on the image above. They represent the input layer and usually are noted as vector ***X.*** There are 4 blue and 4 green circles that represent the hidden layers. These circles represent the "activation" nodes and usually are noted as ***W*** or ***θ***. The red circle is the output layer or the predicted value (or values in case of multiple output classes/types).

Each node is connected with each node from the next layer and each connection (black arrow) has particular weight. Weight can be seen as impact that that node has on the node from the next layer. So if we take a look on one node it would look like this



2.2 APPROPRIATE PROBLEMS FOR NEURAL NETWORK LEARNING:

ANN learning is well-suited to problems in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras and microphones. It is also applicable to problems for which more symbolic representations are often used, such as the decision tree learning. It is appropriate for problems with the following characteristics:

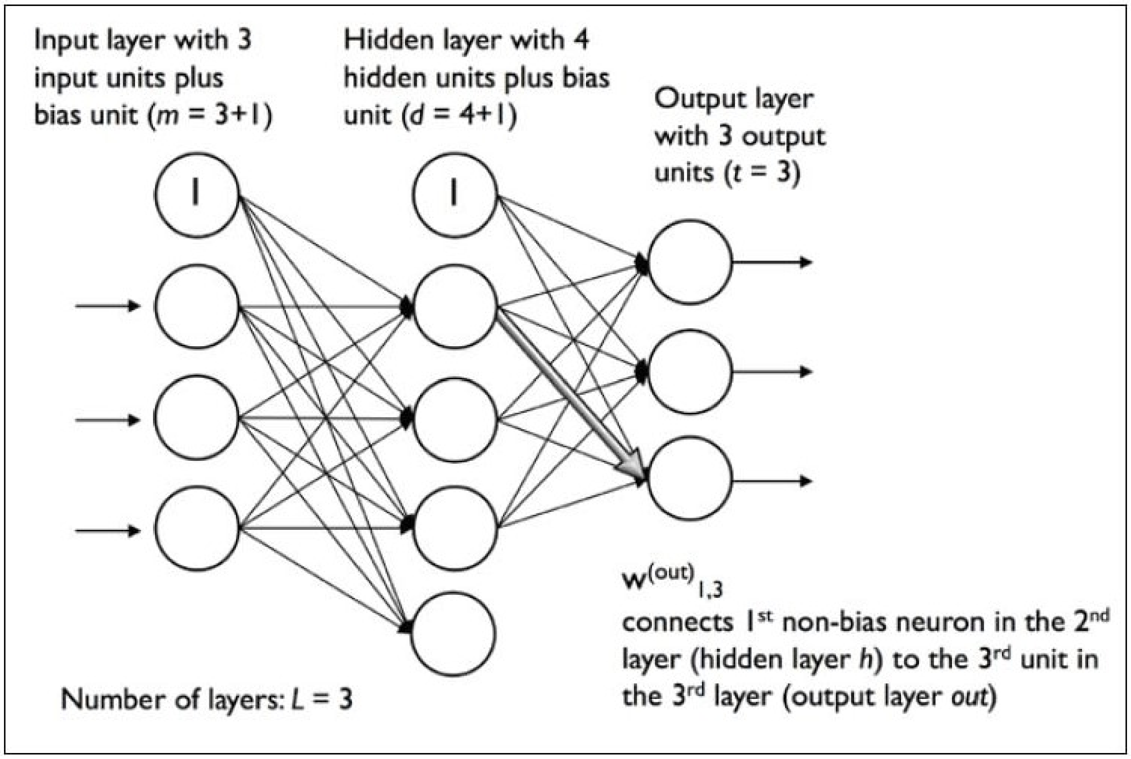
- ***Instances are represented by many attribute-value pairs.***
  The target function to be learned is defined over instances that can be described by a vector of predefined features, such as the pixel values in the ALVINN example. These input attributes may be highly correlated or independent of one another. Input values can be any real values.
- ***The target function output may be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes.***
  For example, in the ALVINN system the output is a vector of 30 attributes, each corresponding to a recommendation regarding the steering direction. The value of each output is some real number between 0 and 1, which in this case corresponds to the confidence in predicting the corresponding steering direction. We can also train a single network to output both the steering command and suggested acceleration, simply by concatenating the vectors that encode these two output predictions.
- ***The training examples may contain errors.***
  ANN learning methods are quite robust to noise in the training data.
- ***Long training times are acceptable.***
  Network training algorithms typically require longer training times than, say, decision tree learning algorithms. Training times can range from a few seconds to many hours, depending on factors such as the number of weights in the network, the number of training examples considered, and the settings of various learning algorithm parameters.
- ***Fast evaluation of the learned target function may be required.***
  Although ANN learning times are relatively long, evaluating the learned network, in order to apply it to a subsequent instance, is typically very fast. For example,

ALVINN applies its neural network several times per second to continually update its steering command as the vehicle drives forward.

- ***The ability of humans to understand the learned target function is not important.*** The weights learned by neural networks are often difficult for humans to interpret. Learned neural networks are less easily communicated to humans than learned rules.

2.3 Multi Layer ANN:

A fully connected multi-layer neural network is called a Multilayer Perceptron (MLP).


Input layer with 3 input units plus bias unit (m = 3+1)
Hidden layer with 4 hidden units plus bias unit (d = 4+1)
Output layer with 3 output units (t = 3)
1
1
$w^{(out)}_{1,3}$
connects 1st non-bias neuron in the 2nd layer (hidden layer h) to the 3rd unit in the 3rd layer (output layer out)
Number of layers: L = 3

It has 3 layers including one hidden layer. If it has more than 1 hidden layer, it is called a deep ANN. An MLP is a typical example of a feedforward artificial neural network. In this figure, the i^{th} activation unit in the l^{th} layer is denoted as $a_i^{(l)}$. The number of layers and the number of neurons are referred to as hyperparameters of a neural network, and these need tuning. Cross-validation techniques must be used to find ideal values for these. The weight adjustment training is done via backpropagation. Deeper neural networks are better at processing data. However, deeper layers can lead to vanishing gradient problem. Special algorithms are required to solve this issue.

Notations

In the representation below:

$$a^{(in)} = \begin{bmatrix} a_0^{(in)} \\ a_1^{(in)} \\ \vdots \\ a_m^{(in)} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^{(in)} \\ \vdots \\ x_m^{(in)} \end{bmatrix}$$

- $a_i^{(in)}$ refers to the i^{th} value in the input layer
- $a_i^{(h)}$ refers to the i^{th} unit in the hidden layer
- $a_i^{(out)}$ refers to the i^{th} unit in the output layer
- $a_0^{(in)}$ is simply the bias unit and is equal to 1; it will have the corresponding weight w_0
- The weight coefficient from layer l to layer l+1 is represented by $w_{k,j}^{(l)}$

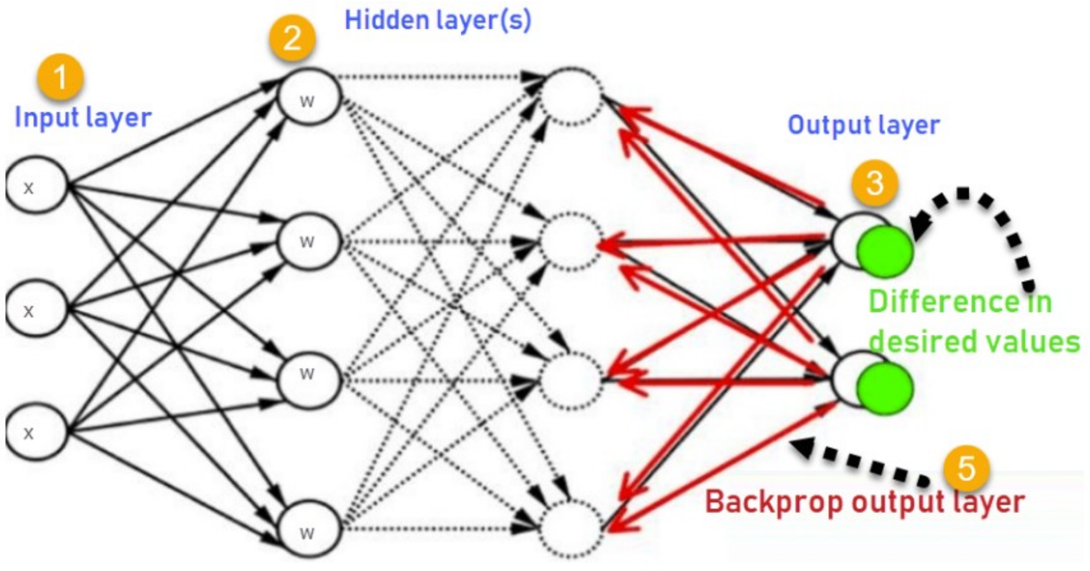
2.4 Backpropagation Algorithm:

Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

How Backpropagation Works: Simple Algorithm

Consider the following diagram



How Backpropagation Works

1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

   Error_B= Actual Output – Desired Output

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

   Keep repeating the process until the desired output is achieved

Advantages of Backpropagation are:

- Backpropagation is fast, simple and easy to program
- It has no parameters to tune apart from the numbers of input
- It is a flexible method as it does not require prior knowledge about the network
- It is a standard method that generally works well
- It does not need any special mention of the features of the function to be learned.