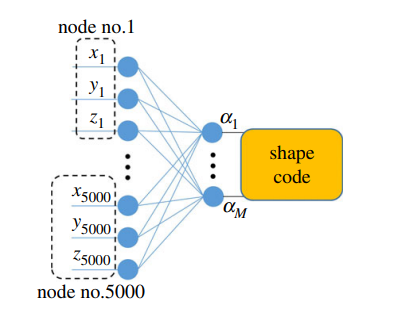
# How to do things: -

# UnsupervisedLearning.m file

## Function

It encodes the stress and shape data.

Shape encoding:



The data ShapeData.mat is a 2D array that has 15000 rows representing the Nodes and 729 columns representing the different shapes. The encoding is done using Principal Component Analysis (PCA). Steps are: -

* We divide the ShapeData for train and test.
* We find mean of ShapeData as MeanShape.
* The we centralize the ShapeData by subtracting MeanShape from it and dividing it by square root of number of training set.
* We find the eigen vector and eigen values of the data by using svd function, i.e., that performs Singular Value Decomposition (SVD).
* We have taken 3 columns of eigen vector to represent out data as it holds more than 80% of our data and stored it in PC
* We will normalize our training and testing data by subtracting MeanShape.
* Finally, we will matrix multiply transpose of PC with test/train data and divide it with the respective square root of eigen values and construct the encoded test and train data respectively.

Note: - Final size of encoded train/test will be 3 x number of train/test data respectively.

Major Doubt: -

* Are we reversing the process of PCA to find the original covariance matrix that represents our data?
* Why are we dividing centralized shape data with square root of number of train data?

Shape encoding:

Diagram

Description automatically generated

In the above diagram going from right to left is encoding. The data StressData.mat is a 4D array that has 4 channels containing stress at different planes of 5000 nodes for each 729 shapes. The encoding is done by using bidirectional neural network using convolution layers. The steps are: -

* We divide the StressData into train and test data, reshape them into 4D array of size 50 x 100 x 3 x number of train/test and save them as SData\_train/SData\_test respectively. Basically, we rearrange the order of dimension of stress data while removing the 3rd column and converted the 1D vector of Nodes into 2D array of size 50 x 100 for convo operation.
* Using im2patch function **each** **train** data is broken into 4D patches of dimension 10 x 20 x 3 x 25, containing 25 Nodes patches of size 10 x 20 for each channel.
* Each patch of train is concatenated in Data1 of size 600 x 25\*number of train data. Here, 600 rows have 200 (single patch) of each channel stress and 25 columns for each train data.
* Applying PCA, i.e., construct Data1 covariance matrix and find its eigen vectors and values using SVD.
* We construct W1 using the first 256 columns of eigen vector into the shape 10 x 20 x 3 x 256.
* Now we will evaluate the convolution of SData\_train with the weights W1 and 0 bias with the stride of 10 and 20, and store in Y1. This is the first layer of

## Inputs

* OutputDataFile: - We will save all the outputs in this file
* ShapeDataFile: - Load the ShapeData.mat
* StressDataFile: - Load the StressData.mat
* IdxList\_train: - This list contains the position of columns (out of 729) which will be our training set
* IdxList\_test: - This list contains the position of columns (out of 729) which will be our testing set

## Outputs

# Im2patch.m

It takes an 2D or 3D array. We will only be using 3D array algorithm to form the patches. The array has 3 channels which contains n x m Nodes. This function divides the n x m into further small patches of size L1 x L2 (here, L1 = 10 and L2 = 20) while taking the stride of S1 and S2 (here, S1 = L1 and S2 = L2). Thus, no repletion of data is happening. If S1 or S2 was less than L1 and L2 respectively then data would have repeated and if was more then data would have been lost.