REPORT – 4

PCA (PRINCIPAL COMPONENT ANALYSIS)

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set

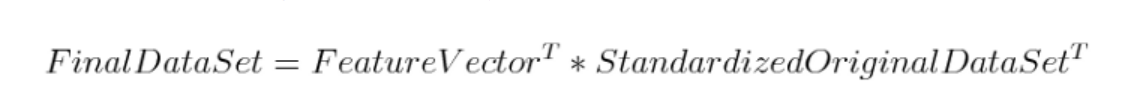
**HOW DO YOU DO A PCA?**

1. Standardize the range of continuous initial variables
2. Compute the covariance matrix to identify correlations
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
4. Create a feature vector to decide which principal components to keep
5. Recast the data along the principal components axes
6. Standardization - The reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges.
7. Covariance matrix computation - The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Icon

   Description automatically generated with medium confidence
8. To identify Principal Components - Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the principal components of the data. Before getting to the explanation of these concepts, let’s first understand what do we mean by principal components.

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components.

1. FEATURE VECTOR- the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only p eigenvectors (components) out of n, the final data set will have only p dimensions.
2. RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES



RECONSTRUCTION FROM COMPRESSED REPRESENTATION –

We can get a approximate value of the original data set by the following formula

Text

Description automatically generated

Choosing value of k ( number of Principal Components )

Text, whiteboard

Description automatically generated

Text

Description automatically generated with medium confidence

Where Sii is square of distance from the PC

NEURAL NETWORKS

Forward propagation

aAs the name suggests, the input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.

In order to generate some output, the input data should be fed in the forward direction only. The data should not flow in reverse direction during output generation otherwise it would form a cycle and the output could never be generated. Such network configurations are known as feed-forward network. The feed-forward network helps in forward propagation.

At each neuron in a hidden or output layer, the processing happens in two steps:

Preactivation: it is a weighted sum of inputs i.e. the linear transformation of weights w.r.t to inputs available. Based on this aggregated sum and activation function the neuron makes a decision whether to pass this information further or not.

Activation: the calculated weighted sum of inputs is passed to the activation function. An activation function is a mathematical function which adds non-linearity to the network. There are four commonly used and popular activation functions — sigmoid, hyperbolic tangent(tanh), ReLU and Softmax.

The hypothesis used here is same as logistic regression

In neural networks, we use the same logistic function as in classification, \frac{1}{1 + e^{-\theta^Tx}}1+*e*−*θTx*1​, yet we sometimes call it a sigmoid (logistic) **activation** function. In this situation, our "theta" parameters are sometimes called "weights".

Our input nodes (layer 1), also known as the "input layer", go into another node (layer 2), which finally outputs the hypothesis function, known as the "output layer".

We can have intermediate layers of nodes between the input and output layers called the "hidden layers."

In this example, we label these intermediate or "hidden" layer nodes a^2\_0 \cdots a^2\_n*a*02​⋯*an*2​ and call them "activation units."

*a*(*j*)*i*="activation" of unit *i* in layer *j*

Θ(*j*)=matrix of weights controlling function mapping from layer *j* to layer *j*+1

Each layer gets its own matrix of weights, \Theta^{(j)}Θ(*j*).

The dimensions of these matrices of weights is determined as follows:

\text{If network has $s\_j$ units in layer $j$ and $s\_{j+1}$ units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s\_{j+1} \times (s\_j + 1)$.}If network has *sj*​ units in layer *j* and *sj*+1​ units in layer *j*+1, then Θ(*j*) will be of dimension *sj*+1​×(*sj*​+1).

The +1 comes from the addition in \Theta^{(j)}Θ(*j*) of the "bias nodes," x\_0*x*0​ and \Theta\_0^{(j)}Θ0(*j*)​. In other words the output nodes will not include the bias nodes while the inputs will.

Text, letter

Description automatically generated