## Parametric assumptions for Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA):

Parametric assumptions, are the base assumptions that algorithms make about the shape of the data. For PCA, assumption that we were making was that the original data took on a shape that could be decomposed and represented by a single linear transformation (the matrix operation).
Model being non-parametric doesn't mean that there are no assumptions at all made by the model during training. While the algorithms forgo the assumption on the shape of the data, they still may make assumptions on other aspects of data, for example, values of the cells.
Parametric assumptions refer to the underlying assumptions about the distribution of the data in a statistical model. These assumptions are crucial because they impact the validity and reliability of the statistical inferences made from the data. Parametric methods assume a specific form or distribution for the data, and violating these assumptions can lead to biased or inefficient estimates.

For Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), let's discuss the key assumptions:

**Principal Component Analysis (PCA):**

1. **Linearity**: PCA assumes that the relationships among variables are linear. If the relationships are highly nonlinear, PCA might not be the most suitable technique.
2. **Independence**: The variables should be independent of each other. Correlated variables can still be analyzed with PCA, but the interpretation becomes more complex.
3. **Homoscedasticity**: Homoscedasticity assumes that the variance of the variables is roughly constant across all levels of the components. This ensures that all variables contribute equally to the principal components.

**Linear Discriminant Analysis (LDA):**

1. **Normality**: LDA assumes that the independent variables (features) are normally distributed within each class. This assumption is particularly important when sample sizes are small.
2. **Equality of Covariance Matrices:** LDA assumes that the covariance matrices of the independent variables are equal across all classes. This means that the spread of data points within each class is the same.

3. **Linearity**: LDA assumes that the relationships between the independent variables are linear. Nonlinear relationships may affect the performance of LDA.
4. **Homoscedasticity**: Similar to PCA, LDA assumes homoscedasticity, meaning that the variance within each class is the same for all classes.
5. **Independence**: LDA assumes that the observations within and between classes are independent.

It's important to note that while these assumptions are ideal for optimal performance of PCA and LDA, both techniques are often quite robust, and violations of these assumptions might not always lead to significant issues, especially with large sample sizes. However, it's advisable to be aware of these assumptions and, if possible, check for their validity in your specific dataset before applying PCA or LDA.

## Differences between feature learning and transformation:

| Aspect | Feature Learning | Feature Transformation |
|---|---|---|
| **Definition** | Involves automatically learning representations or features from raw data without explicitly designing them. | Involves modifying or converting the original features of the data to a new set of features, often based on design choices. |
| **Techniques** | Deep learning methods like autoencoders, CNNs, RNNs. | Standardization, normalization, polynomial feature expansion, PCA, etc. |
| **Flexibility** | Highly flexible, adapts to complex patterns and relationships. | Typically less flexible, as the practitioner designs the transformation rules. |
| **Unsupervised/Supervised** | Can be unsupervised or supervised. Learns patterns and structures with or without labeled data. | Primarily a manual process, often used in preprocessing before applying learning algorithms. |
| **Interpretability** | May lack interpretability due to automated feature discovery. | Often more interpretable, as the practitioner explicitly designs the transformation rules. |
| **Purpose** | Discover informative and discriminative features from raw data. | Modify or convert features to address issues like multicollinearity, scaling, or to create new features. |

| Aspect | Feature Learning | Feature Transformation |
|---|---|---|
| **Overlap** | Some overlap; feature learning techniques may inherently involve a form of transformation. | While conceptually distinct, there can be overlap, and both can be used together. |
| **Complementary** | Can be used in conjunction with feature transformation for preprocessing. | Feature transformation can be used to prepare data for feature learning. |

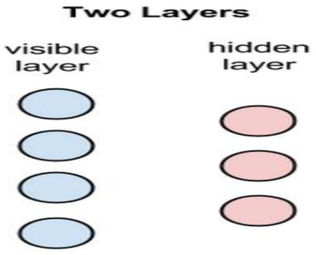
## Restricted Boltzmann Machines (RBM) architecture:

A simple deep learning architecture that is set up to learn a set number of new dimensions based on a probabilistic model that data follows.

RBM is a family of unsupervised feature learning algorithms that use probabilistic models to learn new features. Like PCA and LDA, can use RBMs to extract a new feature set from raw data and use them to enhance ML pipelines. The features that are extracted by RBMs tend to work best when followed by linear models such as linear regression, logistic regression, perceptron's, and so on. The unsupervised nature of RBMs is important as they are more similar to PCA algorithms than they are to LDA. They do not require a ground-truth label for data points to extract new features. This makes them useful in a wider variety of machine learning problems.

RBMs are shallow (two-layer) neural networks. Building blocks of a class of algorithms called Deep Belief Networks (DBN). There is a visible layer (the first layer), followed by a hidden layer (the second layer).

First visible layer of network has as many layers as input feature dimension. Number of nodes in hidden layer is a human-chosen number and represents number of features that we wish to learn.



**Not necessarily dimension reduction:**

In PCA and LDA, we had severe limits to the number of components we were allowed to extract.

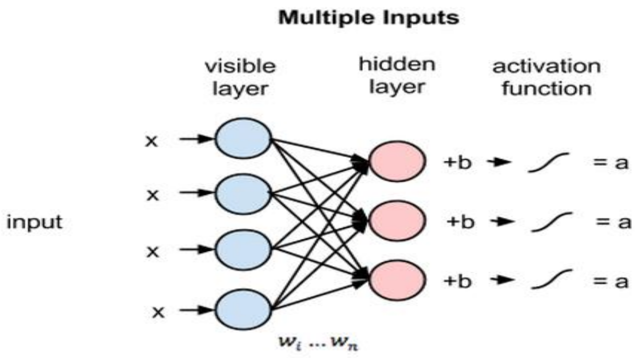
For PCA, we were capped by the number of original features, while LDA enforced the much stricter imposition that caps the number of extracted features to the number of categories in the ground truth minus one. The only restriction on the number of features RBMs are allowed to learn is that they are limited by the computation power of the computer running the network and human interpretation. RBMs can learn fewer or more features than we originally began with. The exact number of features to learn is up to the problem and can be gridsearched.

### RBM Architecture

RBM shows the movement of a single data point through the graph and through a single hidden node.

The visible layer has four nodes, representing the four columns of the original data. Each arrow represents a single feature of the data point moving through the four visible nodes in the first layer of the RBM. Each of the feature values is multiplied by a weight associated to that feature and are added up together. This calculation can also be summed up by a dot product between an input vector of data and a weight vector. The resulting weighted sum of the data is added to a bias variable and sent through an activation function (sigmoidal is popular). The result is stored in a variable called *a*.

In a real RBM, each of the visible nodes is connected to each of the hidden nodes. Because inputs from each visible node are passed to every single hidden node, an RBM can be defined as a symmetrical bipartite graph. The symmetrical part comes from the fact that the visible nodes are all connected with each hidden node.



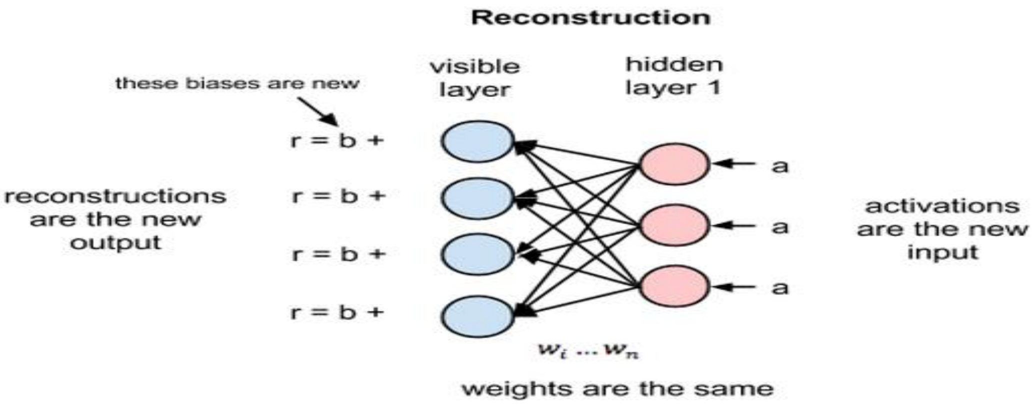
## Reconstruction of data in RBM:

A simple deep learning architecture that is set up to learn a set number of new dimensions based on a probabilistic model that data follows.

RBM is a family of unsupervised feature learning algorithms that use probabilistic models to learn new features. Like PCA and LDA, can use RBMs to extract a new feature set from raw data and use them to enhance ML pipelines. The features that are extracted by RBMs tend to work best when followed by linear models such as linear regression, logistic regression, perceptron's, and so on. The unsupervised nature of RBMs is important as they are more similar to PCA algorithms than they are to LDA. They do not require a ground-truth label for data points to extract new features. This makes them useful in a wider variety of machine learning problems.

### Data Reconstruction:

In this forward pass of the network, we can see how data goes forward through the network (from the visible layer to the hidden layer), but that doesn't explain how the RBM is able to learn new features from our data without ground truths. This is done through multiple forward and backward passes through the network between our visible and hidden layer. In the reconstruction phase, we switch the network around and let the hidden layer become the input layer and let it feed our activation variables (a) backwards into the visible layer using the same weights, but a new set of biases. The activated variables that are calculated during the forward pass are then used to reconstruct the original input vectors. To transform data, we simply pass it through the network and retrieve the activation variables and call those the new features.



## Word embedding and methods for Word embedding:

Word embedding is any of a set of language modelling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually it involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Methods to generate this mapping include

- neural networks
- dimensionality reduction on the word co-occurrence matrix,
- probabilistic models,
- explainable knowledge base method,
- explicit representation in terms of the context in which words appear

Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

What are word embeddings exactly?

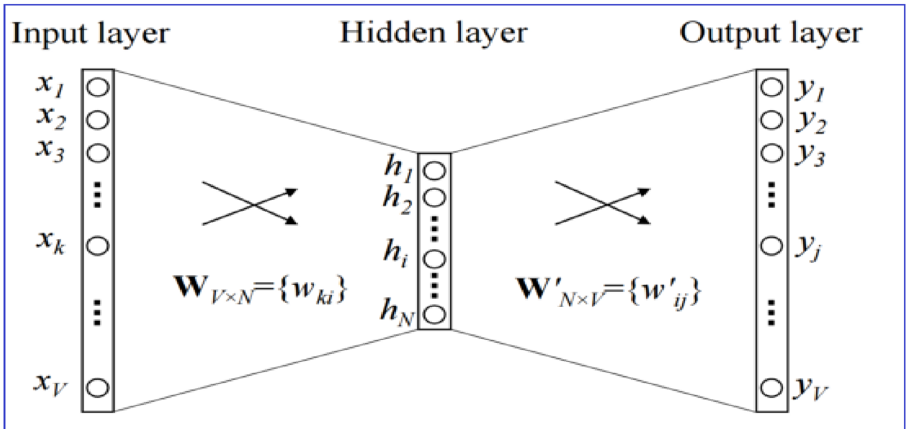
- Loosely speaking, they are vector representations of a particular word.
- Having said this, what follows is how do we generate them?
- More importantly, how do they capture the context?

Word2Vec is one of the most popular technique to learn word embeddings using shallow neural network. It was developed by Tomas Mikolov in 2013 at Google.

## Word2Vec is a method to construct Word embedding:

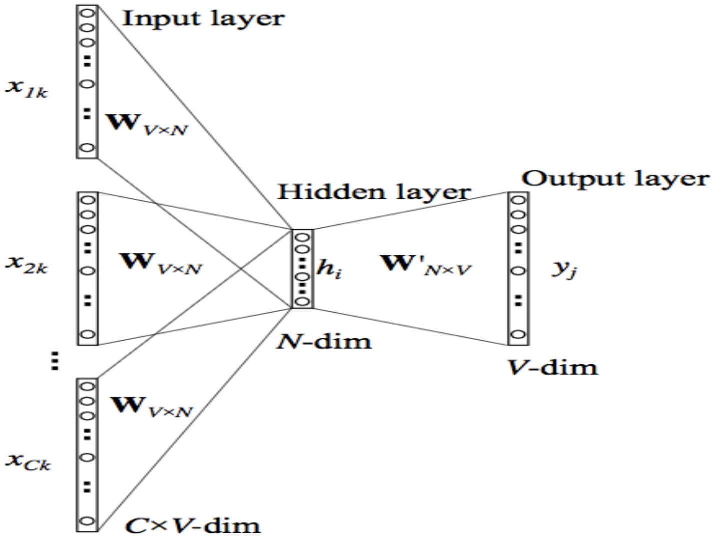
Word2Vec is a method to construct word embedding. It can be obtained using two methods (both involving Neural Networks):

1. Skip Gram and
2. Common Bag of Words (CBOW)



A simple CBOW model with only one word in the context

Earlier model used a single context word to predict the target. Model can use multiple context words to do the same.



**************************************************