# Artificial Intelligence (CSE 5290) – Grad Project Fall 16

Denver Lob & Sreya Bhar

*Instructor: Dr. Debasis Mitra*

* **Principal component analysis**

**Introduction :**

Principal component analysis (PCA) is a statistical procedure that uses an [orthogonal transformation](https://en.wikipedia.org/wiki/Orthogonal_transformation) to convert a set of observations of possibly correlated variables into a set of values of [linearly uncorrelated](https://en.wikipedia.org/wiki/Correlation_and_dependence) variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible [variance](https://en.wikipedia.org/wiki/Variance) (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is [orthogonal](https://en.wikipedia.org/wiki/Orthogonal) to the preceding components. The resulting vectors are an uncorrelated [orthogonal basis set](https://en.wikipedia.org/wiki/Orthogonal_basis_set). PCA is sensitive to the relative scaling of the original variables. [1]

Principal Component Analysis (PCA) is a simple yet popular and useful linear transformation technique that is used in numerous applications, such as stock market predictions, the analysis of gene expression data, and many more. The main goal of a PCA analysis is to identify patterns in data; PCA aims to detect the correlation between variables. If a strong correlation between variables exists, the attempt to reduce the dimensionality only makes sense. In a nutshell, this is what PCA is all about: Finding the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace while retaining most of the information. [2]

Principal component analysis (PCA) is a standard tool in modern data analysis - in diverse fields from neuroscience to computer graphics - because it is a simple, non-parametric method for extracting relevant information from confusing data sets. With minimal effort PCA provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified structures that often underlie it. [3]

### **PCA and Dimensionality Reduction**

Often, the desired goal is to reduce the dimensions of a d-dimensional dataset by projecting it onto a (k)-dimensional subspace (where k<d) in order to increase the computational efficiency while retaining most of the information. Later, eigenvectors (the principal components) of a dataset are computed and collected in a projection matrix. Each of those eigenvectors is associated with an eigenvalue which can be interpreted as the “length” or “magnitude” of the corresponding eigenvector. If some eigenvalues have a significantly larger magnitude than others that the reduction of the dataset via PCA onto a smaller dimensional subspace by dropping the “less informative” eigen pairs is reasonable. [2]

**PCA Approach**

* Standardize the data.
* Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Vector Decomposition.
* Sort eigenvalues in descending order and choose the k- eigenvectors that correspond to the k-largest eigenvalues where k is the number of dimensions of the new feature subspace (k ≤ d).
* Construct the projection matrix W from the selected k- eigenvectors.
* Transform the original dataset X via W to obtain a k-dimensional feature subspace Y.

## ***1 – Eigen decomposition - Computing Eigenvectors and Eigenvalues***

The eigenvectors and eigenvalues 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. [2]

### **Covariance Matrix**

The classic approach to PCA is to perform the eigen decomposition on the covariance matrix Σ, which is a d×d matrix where each element represents the covariance between two features. The covariance between two features is calculated as follows:

Σjk = (1/n−1)∑i=1 to N (xij−x¯j)(xik−x¯k)

We can summarize the calculation of the covariance matrix via the following matrix equation:

Σ = (1/n−1) ((X−x¯)T (X−x¯))

where  x¯ is the mean vector x¯=∑k=1 to n xi

The mean vector is a d-dimensional vector where each value in this vector represents the sample mean of a feature column in the dataset.

Next, an eigen decomposition is performed on the covariance matrix. [2]

### **Correlation Matrix**

Especially the correlation matrix typically used instead of the covariance matrix. However, the eigen decomposition of the covariance matrix yields the same results as a eigen decomposition on the correlation matrix, since the correlation matrix can be understood as the normalized covariance matrix. [2]

### **Singular Vector Decomposition**

While the eigen decomposition of the covariance or correlation matrix may be more intuitive, most PCA implementations perform a Singular Vector Decomposition (SVD) to improve the computational efficiency. So, an SVD is performed to confirm that the result are indeed the same. [2]

## ***2 - Selecting Principal Components***

### **Sorting Eigen pairs**

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.

In order to decide which eigenvector(s) can be dropped without losing too much information for the construction of lower-dimensional subspace, we need to inspect the corresponding eigenvalues: The eigenvectors with the lowest eigenvalues bear the least information about the distribution of the data; those are the ones can be dropped.
In order to do so, the common approach is to rank the eigenvalues from highest to lowest in order choose the top k eigenvectors. [2]

### **Explained Variance**

After sorting the eigen pairs, the next question is “how many principal components are we going to choose for our new feature subspace?” A useful measure is the so-called “explained variance,” which can be calculated from the eigenvalues. The explained variance tells us how much information (variance) can be attributed to each of the principal components. [2]

###  **Projection Matrix**

It’s about time to get to the really interesting part: The construction of the projection matrix that will be used to transform the Iris data onto the new feature subspace. Although, the name “projection matrix” has a nice ring to it, it is basically just a matrix of our concatenated top k eigenvectors. The 4-dimensional feature space can be reduced to a 2-dimensional feature subspace, by choosing the eigenvectors with the highest eigenvalues to construct our d× k-dimensional eigenvector matrix W. [2]

##  ***3 - Projection onto the New Feature Space***

In this last step the dimensional projection matrix W will be used to transform our samples onto the new subspace via the equation:

Y=X×W, where Y is a matrix of our transformed samples.

Now, what we will get after applying the linear PCA transformation is a lower dimensional subspace (from 3D to 2D in this case), where the samples are “most spread” along the new feature axes. [2]

**Quick Summary of PCA :**

1. Organize data as an m×n matrix, where m is the number of measurement types and n is the number of samples.
2. Subtract off the mean for each measurement type.
3. Calculate the Singular Value Decomposition or the eigenvectors of the covariance. [3]

**DISCUSSION**

Principal component analysis (PCA) has widespread applications because it reveals simple underlying structures in complex data sets using analytical solutions from linear algebra. A primary benefit of PCA arises from quantifying the importance of each dimension for describing the variability of a data set. In particular, the measurement of the variance along each principle component provides a means for comparing the relative importance of each dimension. An implicit hope behind employing this method is that the variance along a small number of principal components (i.e. less than the number of measurement types) provides a reasonable characterization of the complete data set. This statement is the precise intuition behind any method of dimensional reduction – a vast arena of active research. Although PCA “works” on a multitude of real world problems, any diligent scientist or engineer must ask when does PCA fail? Before we answer this question, let us note a remarkable feature of this algorithm. PCA is completely nonparametric: any data set can be plugged in and an answer comes out, requiring no parameters to tweak and no regard for how the data was recorded. From one perspective, the fact that PCA is non-parametric (or plug-and-play) can be considered a positive feature because the answer is unique and independent of the user. From another perspective the fact that PCA is agnostic to the source of the data is also a weakness. [3]

### **Limitation of PCA :**

The results of PCA depend on the scaling of the variables. A scale-invariant form of PCA has been developed.

The applicability of PCA is limited by certain assumptionsmade in its derivation. [1]

**Applications of PCA :**

### Interest Rate Derivatives Portfolios:

Principal component analysis can be directly applied to the risk management of interest rate derivatives portfolios. Trading multiple swap instruments which are usually a function of 30-500 other market quotable swap instruments is sought to be reduced to usually 3 or 4 principal components, representing the path of interest rates on a macro basis. Converting risks to be represented as those to factor loadings (or multipliers) provides assessments and understanding beyond that available to simply collectively viewing risks to individual 30-500 buckets. [1]

### Neuroscience:

A variant of principal components analysis is used in [neuroscience](https://en.wikipedia.org/wiki/Neuroscience) to identify the specific properties of a stimulus that increase a [neuron](https://en.wikipedia.org/wiki/Neuron)'s probability of generating an [action potential](https://en.wikipedia.org/wiki/Action_potential). This technique is known as [spike-triggered covariance analysis](https://en.wikipedia.org/wiki/Spike-triggered_covariance). In a typical application an experimenter presents a [white noise](https://en.wikipedia.org/wiki/White_noise) process as a stimulus (usually either as a sensory input to a test subject, or as a [current](https://en.wikipedia.org/wiki/Electric_current) injected directly into the neuron) and records a train of action potentials, or spikes, produced by the neuron as a result. Presumably, certain features of the stimulus make the neuron more likely to spike. In order to extract these features, the experimenter calculates the [covariance matrix](https://en.wikipedia.org/wiki/Covariance_matrix) of the *spike-triggered ensemble*, the set of all stimuli (defined and discretized over a finite time window, typically on the order of 100 ms) that immediately preceded a spike. The [eigenvectors](https://en.wikipedia.org/wiki/Eigenvectors_and_eigenvalues) of the difference between the spike-triggered covariance matrix and the covariance matrix of the *prior stimulus ensemble* (the set of all stimuli, defined over the same length time window) then indicate the directions in the [space](https://en.wikipedia.org/wiki/Vector_space) of stimuli along which the variance of the spike-triggered ensemble differed the most from that of the prior stimulus ensemble. Specifically, the eigenvectors with the largest positive eigenvalues correspond to the directions along which the variance of the spike-triggered ensemble showed the largest positive change compared to the variance of the prior. Since these were the directions in which varying the stimulus led to a spike, they are often good approximations of the sought after relevant stimulus features. In neuroscience, PCA is also used to discern the identity of a neuron from the shape of its action potential. [Spike sorting](https://en.wikipedia.org/wiki/Spike_sorting) is an important procedure because [extracellular](https://en.wikipedia.org/wiki/Electrophysiology#Extracellular_recording) recording techniques often pick up signals from more than one neuron. In spike sorting, one first uses PCA to reduce the dimensionality of the space of action potential waveforms, and then performs [clustering analysis](https://en.wikipedia.org/wiki/Cluster_analysis) to associate specific action potentials with individual neurons.

PCA as a dimension reduction technique is particularly suited to detect coordinated activities of large neuronal ensembles. It has been used in determining collective variables, i.e. [order parameters](https://en.wikipedia.org/w/index.php?title=Order_parameters&action=edit&redlink=1), during [phase transitions](https://en.wikipedia.org/wiki/Phase_transitions) in the brain. [1]

* **Linear Discriminant Analysis**

**Introduction**

The famous dimensional reduction technique used is “Principal Component Analysis”, is used for dimensionality reduction by focusing on the data with the most variations. To solve dimensionality reduction for data with higher attributes, LDA, also known as Fisher’s linear discriminant analysis (FDA), is used in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset of n-dimensional samples, onto a lower-dimensional space with good class-separability. The general LDA approach is very similar to a PCA, we not only try to find the component axes that maximize the variance of the data (PCA), we are also interested in the linear transformation that maximize the separation between multiple classes (LDA).[4]

Here we demonstrate the Linear Discriminant Analysis algorithm which is similar to the Principal Component Analysis, which is useful plotting data with a lot dimensions onto a simple X/Y plot. Using LDA, the goal is to project a feature space onto a smaller subspace between multiple classes. [4]

**Linear Discriminant Analysis Steps**

Listed below are the 5 general steps for performing a linear discriminant analysis which is just another preprocessing step for a classification task.

* 1. Compute the d-dimensional mean vectors for the different classes from the dataset, where **m**i is the mean vectors.
	2. Compute the scatter matrices:

LDA computes a transformation that the vector maximizes the between-class scatter while minimizing the within-class scatter. We have to calculate the vector v that maximizes the ratio Sb / Sw , where , W is the "within-class scatter matrix” and B is the "between-class scatter" matrix.

* + 1. Computing the Within-class scatter matrix Sw





* + 1. Computing the Between-Class scatter matrix Sb



Where **m** is the overall mean, and *mi* and *Ni* are the sample mean and sizes of the respective classes. [4]

* 1. Computing the eigenvectors (e1, e2, ..., ed) and corresponding eigenvalues

(λ1, λ2, ..., λd) for the scatter matrices. The eigenvectors are basically the direction of a distortion in the linear transformation, and the eigenvalues are the scaling factor for the eigenvectors that describing the magnitude of the distortion. While performing LDA, the eigen vectors that form the new axes of out feature subspace gives us how informative they are for the dimensionality reduction. [4]

* 1. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a *d × k* dimensional matrix W (where every column represents an eigenvector):

To decide which eigenvector(s) we want to drop for our lower-dimensional subspace, we must look at the corresponding eigenvalues of the *k*-eigenvectors. The eigenvectors with the lowest eigenvalues bearing no or least information about the distribution of the data, and those are the ones we want to eliminate. [4]

* 1. Use this *d × k* eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication:

Y = X × W (where X is a n × *d*-dimensional matrix representing the n samples, and y are the transformed n × k-dimensional samples in the new subspace). This would generate linear discriminants that separates the classes quite nicely. [4]

**Applications**

 Linear Discriminant Analysis (LDA) is a well-known technique used for feature extraction and dimension reduction. It is widely used in many applications involving high-dimensional data, such as face recognition and image retrieval. LDA has also been proposed for generic object recognition [7], but results using a large database of objects have not been reported yet.

 **PCA vs LDA**

In appearance-based recognition methods, PCA and LDA have been demonstrated to be useful for many applications such as face recognition. Although one might think that LDA should always outperform PCA (since it deals directly with class discrimination), based on the size and dimensions of data, the result would vary. PCA might outperform LDA when the number of samples per class is small or when the training data nonuniformly sample the underlying distribution. [5]

PCA can be described as an “unsupervised” algorithm, since it “ignores” class labels and its goal is to find the directions that maximize the variance in a dataset, which can be used to group similar objects. In contrast to PCA, LDA is “supervised” and computes the directions that will represent the axes that maximize the separation between multiple classes, which can be used in making predictions. [4]

The dataset we would be using to demonstrate the difference between PCA and LDA is Tic Tac Toe, which is publicly available dataset [6]. Using the PCA & LDA algorithm, we will try to determine which algorithm outperforms the other.

**References:**

[1] [https://en.wikipedia.org/wiki/Principal\_component\_analysis#](https://en.wikipedia.org/wiki/Principal_component_analysis)

[2] [http://sebastianraschka.com/Articles/415\_pca\_in\_3\_steps.html#a-summary-of-the-pca-approach](http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html#a-summary-of-the-pca-approach)

[3] <http://cs.fit.edu/~dmitra/ArtInt/ProjectPapers/PcaTutorial.pdf>

[4] Sebastian Raschka, Linear Discriminant Analysis Bit by Bit, <http://sebastianraschka.com/Articles/414_python_lda.html> , 414.

[5] Zhihua Qiao, Lan Zhou and Jianhua Z. Huang, Effective Linear Discriminant Analysis for High Dimensional, Low Sample Size Data

[6] Tic Tac Toe Dataset - [https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame](https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe%2BEndgame)

[7] D.L. Swets and J.J. Weng, “Using Discriminant Eigenfeatures for Image Retrieval”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, no. 8, pp. 831-836, Aug. 1996.