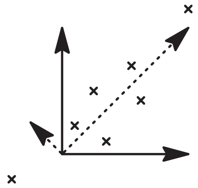
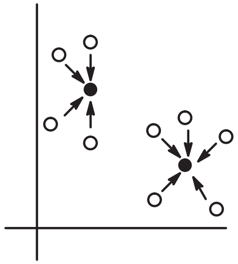
DATA COMPRESSION

There are two main ways of compressing numeric multidimensional data:



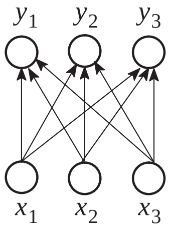


- * Dimensionality reduction:
  - + find directions that show maximal variation using *eigenvalue* techniques and neglect other directions.
- * Clustering or vector quantization: use *prototypes* to encode a group of points.

COORDINATE TRANSFORMATIONS

- * Linear transformation: $\boldsymbol{y} = W\boldsymbol{x}$.
- * Note that a single layer neural network without activation function also performs a linear transformation:

  $y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$, etc.



- * Suppose that a coordinate transformation is given by a matrix A:

$$\boldsymbol{x}^* = A\boldsymbol{x}$$
$$\boldsymbol{y}^* = A\boldsymbol{y}.$$

- * Because $\boldsymbol{x} = A^{-1}\boldsymbol{x}^*$ it follows: $\boldsymbol{y}^* = AWA^{-1}x^*$ or $\boldsymbol{y}^* = W^*x^*$, with $W^* = AWA^{-1}$. W^* and W are called *similar* matrices.

EIGENVALUES AND EIGENVECTORS

- * For some vectors: $W\boldsymbol{v} = \lambda\boldsymbol{v}$.
- * Such a vector $\boldsymbol{v}$ is called an *eigenvector* of W and λ is the corresponding *eigenvalue.*
- * Consider the matrix Y the columns of which are eigenvectors of W. Then: $WY = Y\Lambda$, with Λ a *diagonal matrix*.
- * Therefore: $Y^{-1}WY = Y^{-1}Y\Lambda = \Lambda$. This means that for each transformation W, there is an equivalent transformation by means of a diagonal matrix that uses the eigenvectors as a basis.

RANDOM VECTORS

- * A *random vector* is a vector whose components are random variables.
- * A random vector $\boldsymbol{X}$ has a probability density function $p(\boldsymbol{X})$.
- * The *mean vector* is defined as: $\boldsymbol{M} = E[\boldsymbol{X}] = \int \boldsymbol{X}p(\boldsymbol{X})d\boldsymbol{X}$ (integrate separately for each vector element).
- * And the *covariance matrix* is defined as: $\Sigma = E\left[(\boldsymbol{X} - \boldsymbol{M})(\boldsymbol{X} - \boldsymbol{M})^T\right]$.

SAMPLED RANDOM VECTORS

* In practice, the probability density function is unknown and one only has samples $\boldsymbol{X}^k$ $(k = 1, \ldots, N)$.

* The *sample mean vector* is defined as:

$$\boldsymbol{M} = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{X}^k.$$

* The *sample covariance matrix* is defined as:

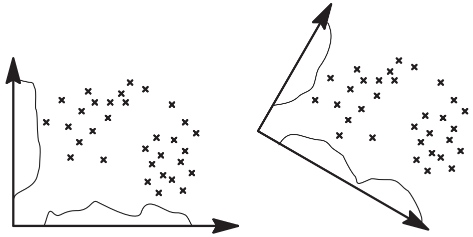
$$\Sigma = \frac{1}{N}\sum_{k=1}^{N}(\boldsymbol{X}^k - \boldsymbol{M})(\boldsymbol{X}^k - M)^T$$

PRINCIPAL COMPONENTS ANALYSIS (1)

* Abbreviated by PCA; also called *Karhunen-Loève transformation*.

* Question: given a random vector (a set of sampled vectors), find an orthogonal base that maximizes the variance along the subsequent dimensions of the base.

* Goal: achieve dimensionality reduction by leaving out those dimensions that show low variance.



PRINCIPAL COMPONENTS ANALYSIS (2)

* Consider the sampled vectors $\boldsymbol{X}^k$ $(k = 1, \ldots, N)$ with $\boldsymbol{M} = 0$ (if $\boldsymbol{M} \neq 0$, construct a new set of vectors $\boldsymbol{Z}^k = \boldsymbol{X}^k - \boldsymbol{M}$).

* We are looking for a unit vector $\boldsymbol{u}$ on which the $\boldsymbol{X}^k$ will be projected. The projection is:

$$p_k = \boldsymbol{X}^k \cdot \boldsymbol{u} = \boldsymbol{X}^{k^T}\boldsymbol{u} = \boldsymbol{u}^T\boldsymbol{X}^k$$

* The projection's mean is also zero:

$$\frac{1}{N}\sum_{k=1}^{N}p_k = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{u}^T\boldsymbol{X}^k = \boldsymbol{u}^T\frac{1}{N}\sum_{k=1}^{N}\boldsymbol{X}^k = 0.$$

* The projection's variance:

$$\sigma^2(\boldsymbol{u}) = \frac{1}{N}\sum_{k=1}^{N}p_k^2 = \frac{1}{N}\sum_{k=1}^{N}\left(\boldsymbol{u}^T\boldsymbol{X}^k\right)\left[\boldsymbol{X}^{k^T}\boldsymbol{u}\right] = \boldsymbol{u}^T\Sigma\boldsymbol{u}.$$

PRINCIPAL COMPONENTS ANALYSIS (3)

* At the maximal point of variance:

$$\sigma^2(\boldsymbol{u} + \Delta\boldsymbol{u}) = \sigma^2(\boldsymbol{u})$$

$$\sigma^2(\boldsymbol{u} + \Delta\boldsymbol{u}) = (\boldsymbol{u} + \Delta\boldsymbol{u})^T\Sigma(\boldsymbol{u} + \Delta\boldsymbol{u})$$

* Ignoring second-order terms:

$$\sigma^2(\boldsymbol{u} + \Delta\boldsymbol{u}) = \boldsymbol{u}^T\Sigma\boldsymbol{u} + \Delta\boldsymbol{u}^T\Sigma\boldsymbol{U} + \boldsymbol{u}^T\Sigma\Delta\boldsymbol{u}$$

* Because Σ is symmetric, $\boldsymbol{A}^T\Sigma\boldsymbol{B} = \boldsymbol{B}^T\Sigma\boldsymbol{A}$, and therefore:

$$\sigma^2(\boldsymbol{u} + \Delta\boldsymbol{u}) = \boldsymbol{u}^T\Sigma\boldsymbol{u} + 2\Delta\boldsymbol{u}^T\Sigma\boldsymbol{u} = \sigma^2(\boldsymbol{u}) + 2\Delta\boldsymbol{u}^T\Sigma\boldsymbol{u}$$

* It can be concluded that:

$$\Delta\boldsymbol{u}^T\Sigma\boldsymbol{u} = 0$$

* The unit-vector constraint means:

$$(\boldsymbol{u} + \Delta\boldsymbol{u})^T(\boldsymbol{u} + \Delta\boldsymbol{u}) = 1 \text{ or } \Delta\boldsymbol{u}^T\boldsymbol{u} = 0.$$

PRINCIPAL COMPONENTS ANALYSIS (4)

- Using the technique of *Lagrange multipliers* one gets:

$$\Delta \boldsymbol{u}^T \Sigma \boldsymbol{u} - \lambda \Delta \boldsymbol{u}^T \boldsymbol{u} = 0 \text{ or } \Sigma \boldsymbol{u} = \lambda \boldsymbol{u}.$$

- So, the vector $\boldsymbol{u}$ should be an eigenvector of the covariance matrix Σ. But:

$$\sigma^2(\boldsymbol{u}) = \boldsymbol{u}^T \Sigma \boldsymbol{u} = \boldsymbol{u}^T \lambda \boldsymbol{u} = \lambda.$$

- So the largest variance is achieved when the eigenvector corresponding to the largest eigenvalue is chosen.
- Note that the eigenvectors of a symmetrical matrix are orthogonal and can be chosen as a base.

PRINCIPAL COMPONENTS ANALYSIS (5)

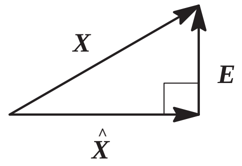
- Take the eigenvalues in decreasing order ($\lambda_1 > \lambda_2 > \ldots > \lambda_n$) and construct a matrix $\Phi = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n]$ where each column is an eigenvector.
- One can then write: $\Sigma\Phi = \Phi\Lambda = \Lambda\Phi$ where Λ is a diagonal matrix with the eigenvalues in decreasing order in the diagonal. This leads to: $\Phi^{-1}\Sigma\Phi = \Phi^T\Sigma\Phi = \Lambda$.
- So, after a transformation with Φ, the covariance matrix becomes a diagonal matrix.

DIMENSIONALITY REDUCTION (1)

- Expressing a vector $\boldsymbol{X}$ as a linear combination of the eigenvectors gives: $\boldsymbol{X} = \sum_{i=1}^{n} q_i \boldsymbol{u}_i$.
- One can approximate $\boldsymbol{X}$ by leaving out the last $n - m$ terms of the sum ($m < n$): $\hat{\boldsymbol{X}} = \sum_{i=1}^{m} q_i \boldsymbol{u}_i$.
- The approximation error is then:

$$\boldsymbol{E} = \boldsymbol{X} - \hat{\boldsymbol{X}} = \sum_{i=m+1}^{n} q_i \boldsymbol{u}_i.$$

- Note that the error is always orthogonal to the approximating vector. This is the *principle of orthogonality.*



DIMENSIONALITY REDUCTION (2)

- Consider the total variance of the approximating vector:

$$\sum_{i=1}^{m} \sigma_i^2 = \sum_{i=1}^{m} \lambda_i$$

where σ_i is the variance in the ith dimension after projection on the base of eigenvectors Φ.
- This means that it is indeed a good idea to order the eigenvalues from large to small and take as many vectors from Φ as desired.

DIMENSIONALITY REDUCTION (3)

- Denote: $\Phi^m = \left[\boldsymbol{u}_1, \dots, \boldsymbol{u}_m\right]$.
- $\hat{\boldsymbol{X}} = \sum_{i=1}^{m} q_i \boldsymbol{u}_i$, means that the n-dimensional $\boldsymbol{X}$ can now be represented by an m-dimensional vector $\boldsymbol{Q}$.
- So: $\hat{\boldsymbol{X}} = \Phi^m \begin{bmatrix} q_1 \\ \dots \\ q_m \end{bmatrix} = \Phi^m \boldsymbol{Q}$.
- $\boldsymbol{Q}$ is found by: $Q = (\Phi^m)^T \boldsymbol{X}$ from the original vector $\boldsymbol{X}$.

For more information on PCA, consult:

[1] Haykin, S., Neural Networks, A Comprehensive Foundation, Prentice Hall International, Upper Saddle River, New Jersey, Second Edition, (1999).

[2] Jang, J.S.R., C.T. Sun and E. Mizutani, Neuro–Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence, Prentice Hall, Upper Saddle River, NJ, (1997).

HIGH-DIMENSIONAL SPACES

- The *sample covariance matrix* when the mean is zero, is defined as:

$$\Sigma = \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{X}^k (\boldsymbol{X}^k)^T$$

- Create an $n \times N$ matrix A composed of the sampled vectors:

$$A = \left[\boldsymbol{X}^1, \boldsymbol{X}^2, \dots, \boldsymbol{X}^N\right]$$

- Then the $n \times n$ covariance matrix can also be written as:

$$\Sigma = \frac{1}{N} A A^T.$$

- Finding the eigenvectors of Σ for large n is difficult, if not intractable. We consider here cases where $N < n$.
- Consider an eigenvector $\boldsymbol{v}$ of $A^T A$, an $N \times N$ matrix:

$$A^T A \boldsymbol{v} = \mu \boldsymbol{v}.$$

- Premultiply by A:

$$A A^T A \boldsymbol{v} = A \mu \boldsymbol{v} = \mu A \boldsymbol{v}.$$

- So, when $\boldsymbol{v}$ is an eigenvector of $A^T A$, $A\boldsymbol{v}$ is an eigenvector of Σ!

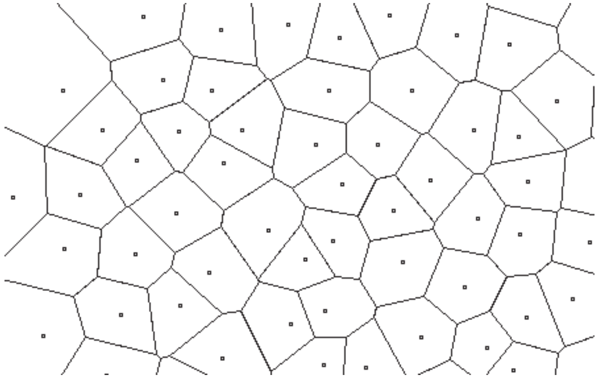
EXAMPLE: EIGENFACES

- It considers images of faces represented by $256 \times 256 = 65536$ pixels.
- A direct application of the PCA to find the eigenvalues and eigenvectors of the covariance matrix would lead to a calculation involving 65536×65536 matrix!
- However, using the technique for high-dimensional spaces requires finding the eigenvectors for an $N \times N$ matrix, where N is the number of samples, e.g. 16.
- The eigenvectors found form a base to represent faces, the so-called *face space*.
- The coordinates in face space form a feature vector that can be used for face recognition.

CLUSTERING: VECTOR QUANTIZATION

- *Clustering* is the process of representing a set of N data points $\boldsymbol{X}^i$ ($i = 1, \dots, N$) by a set of M data points $\boldsymbol{Y}^j$ ($j = 1, \dots, M$, $M < N$); the smaller set should be in some sense “representative” of the larger one.
- *Vector quantization* is an application of clustering in lossy data compression. The M data points form a *codebook* that is available at the receiver side. Given a point $\boldsymbol{X}^i$ to transmit, the sender finds the point $\boldsymbol{Y}^j$ closest to $\boldsymbol{X}^i$ from the codebook and simply sends the index j. The receiver reconstructs an approximation of the data by retrieving $\boldsymbol{Y}^j$ from the codebook.

VORONOI DIAGRAMS



- Subdivide the plane in *Voronoi* regions by perpendicular bisectors between neighboring pairs of points.

k-MEANS CLUSTERING ALGORITHM (1)

- It finds k mean vectors that represent a set of data points by means of k classes.
- The algorithm is as follows:

1) Start with cluster points $\boldsymbol{Y}^j$ ($j = 1, \ldots, M$) and determine the corresponding Voronoi region $\mathcal{V}^j$.

2) Determine the *centroids* of the data points $\boldsymbol{X}^i$ ($i = 1, \ldots, N$) contained in each Voronoi region:

$$\frac{1}{|\mathcal{V}^j|}\sum_{\boldsymbol{X}^i \in V^j} \boldsymbol{X}^i$$

3) Assign the centroids to the cluster points $\boldsymbol{Y}^j$ and repeat from Step 2 until convergence.

k-MEANS CLUSTERING ALGORITHM (2)

Remarks:

- The algorithm will always converge, but not necessarily to the global minimum.
- Quality of final solution strongly depends on initial assignment of cluster points (e.g. random choice, or a choice based on principle components).

Source:

[3] Moon, T.K. and W.C. Stirling, ”Mathematical Methods and Algorithms for Signal Processing”, Prentice Hall, Upper Saddle River, (2000).