# ECON484 Machine Learning

## 5. Dimension Reduction (Part 2, Feature Extraction)

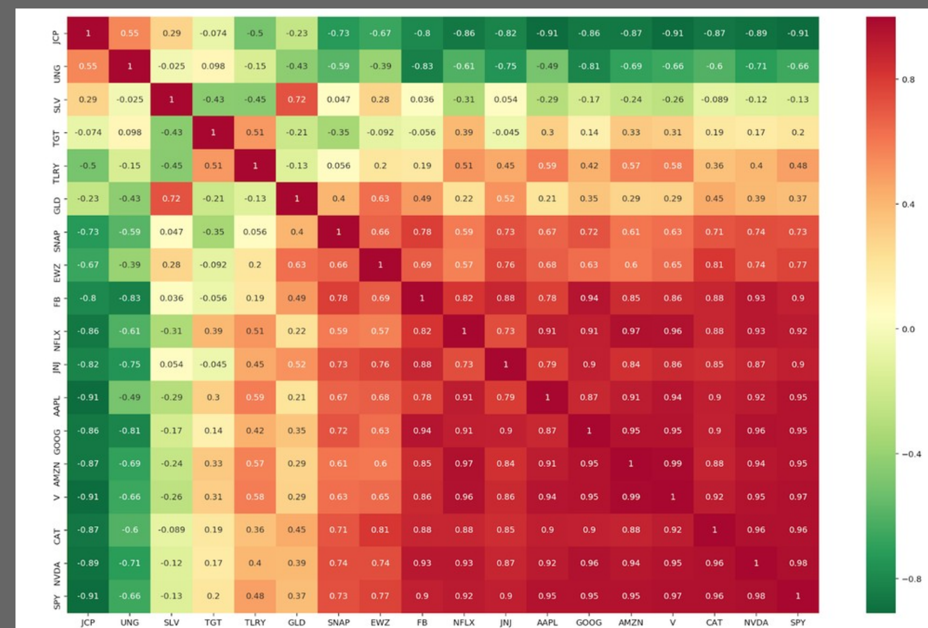
Lecturer:
Bora GÜNGÖREN
bora.gungoren@atilim.edu.tr

# Independent Component Analysis (ICA)

- Let’s assume that our training data set is produced by putting together a number of independent sources.
  - This is realistic in some scenarios where a client just gathers **all data they have** and ask you to make something out of it.
  - This is also realistic in **carefully designed experiments in science and engineering**. Factories and their quality control systems, production management systems, logistics involve a lot of experiments conducted by engineers.
  - This is also an important step in understanding what would be different if the data set did not include independent variables.
- Our concern in such a data set is noise.
  - In noise terminology our data is called the signal. This is due to its roots from electrical engineering.
  - **Independent Component Analysis (ICA)** gives us a technique to separate noise from the actual signal for analysis. The largest signals can then be modeled, ignoring smaller signals as noise. If there is a great amount of noise superimposed on a small signal, noise can be separated such that the small signal isn’t lost.
  - The only assumption is that source signals must be non-Gaussian.

# Independent Component Analysis (ICA)

- If we develop a heat map by coloring a covariance matrix, we will see a checkered pattern.
  - This matrix basically shows us which variables vary together (ie. correlated)
  - Code sample developing the image shown – https://bit.ly/3O4ZmP5
- Each checker represents a group of variables very strongly related.
  - They could be dependent on each other.
  - They could be representing a single quality altogether.
  - Preferring the second explanation gives us many opportunities in data manipulation, for example working with missing data.



# Independent Component Analysis (ICA)

- In Python, Sci-kit library has an implementation of ICA called FastICA. It is usually used for exploratory data analysis.
  - FastICA Documentation and Example – https://bit.ly/3jwbRVJ
  - Detailed Examples with References to Research Papers – https://bit.ly/3jACHfA
- In R we have the FastICA as a separate package
  - R example - https://bit.ly/3vbqwLC
- In better use, we should first **check our input signals for being non-Gaussian**. This is done with typical normality tests, in particular Shapiro-Wilk test and Anderson-Darling test.
  - Shapiro-Wilk test **quantifies how likely it is that your signal was drawn from a Gaussian distributed source**.
  - Anderson-Darling test **gives a number of statistics instead of a single value**.
  - Python Example (with many techniques) - https://bit.ly/3JDEL0T
  - R Examples - https://bit.ly/3xp1evV  and https://bit.ly/37FRVNw

# Principle Component Analysis (PCA)

- ICA is popular among data scientists because it does not require much additional memory and can be implemented rather fast using the FastICA algorithm.
  - However, its assumption that the data sources are independent it also often violated.
- When we have to assume some signals are dependent, we usually end up with trying to reduce them.
  - Selection techniques may not work well because the information content of the discarded signals would reduce the explanatory capacity of our model.
  - How about **creating a new set of variables** that include most of the information?
- PCA is useful because it projects the data onto the principal components of the data set.
  - The principal components are the eigenvectors of the covariance matrix.
  - The first principle component is the eigenvector corresponding to the largest eigenvalue, the second component corresponds to the second largest and so on.

# Principle Component Analysis (PCA)

- PCA is useful because it projects the data onto the principal components of the data set. (cnt’d)
  - The eigenvalues are (by definition) uncorrelated.
    - This results in the projected data **having no covariance**, avoiding the collinearity problem altogether.
  - The principal components (eigenvectors) are **ordered by their contribution to the variance** in the original data set.
    - So **when you are constrained for an exact number of variables** (ie. memory) it is trivial how to choose the components.
  - PCA is therefore very popular
    - Python example - https://bit.ly/37H7zrW
    - R example - https://bit.ly/3xoUuhN

# Principle Component Analysis (PCA)

- PCA is a variation of FA.
- That means there are some **assumptions**:
  - If we constrain $\Psi = \sigma^2 I$ such that
    - Let W (factor loading matrix) be **orthonormal**, **and**
    - Let $\sigma^2 \rightarrow 0$, then we have PCA.
    - If we let $\sigma^2$ be nonzero, then we have probabilistic PCA (PPCA).
  - Recall from linear algebra:
    - If two vectors in an inner product space are orthonormal if they are orthogonal (or perpendicular along a line) unit vectors.
    - A matrix being orthonormal means, the column vectors form an orthonormal set.
  - In factor analysis, W is a matrix used to transform transform $z_i$ to $x_i$, ie. $x_i = Wz_i$, so that we assumed $x_i$ is a linear combination of $z_i$.
- However these assumptions make these two techniques different in their applications.
  - PCA is efficient in finding the components that maximize variance. This is useful **if we are interested in reducing the number of variables while keeping a maximum of variance**.
  - However, PCA does not estimate specific effects.

# Principle Component Analysis (PCA)

- How complex is PCA?
  - For N data points, each with C features to reduce in our training data set, PCA is O( min($N^3$,$C^3$) ) which is not very desirable. However, assuming there are **roughly as many samples as features** it gets reduced to O($N^2$).
  - We also need enough memory to factorize the matrix. Which is difficult to have with larger data sets.
  - So with larger data sets and/or large number of features PCA may become a problem computation-wise.

# Principle Component Analysis (PCA)

- An example. We have multiple test scores of several high school students
  - Maths
  - Physics
  - Chemistry
  - Biology
  - Languages (Turkish, English, any additional)
  - History
  - Geography
  - Philosophy
- How do we suggest who could be successful as
  - A computer engineer
  - A psychologist
  - An economist
  - A lawyer

- How would Factor Analysis approach the problem
- How would Principle Component Analysis approach the problem?

# Non-Negative Matrix Factorization (NMF)

- NMF is a technique for obtaining low rank representation of matrices with non-negative or positive elements.
  - Given a data matrix A of N rows and C columns with each and every element $a_{ij} \geq 0$,
  - NMF seeks matrices W and H of size N rows and k columns, and k rows and C columns, respectively, such that $A \approx WH$, and every element of matrices W and H is either zero or positive.
  - The quantity k is set by the modeler and is required to be $k \leq \min(N,C)$.

# Non-Negative Matrix Factorization (NMF)

- The matrix W  is generally called the dictionary or basis matrix, and H is known as expansion or coefficient matrix.
    - The core idea is that a given data matrix A can be expressed in terms of **summation of k basis vectors** (columns of W) multiplied by the corresponding coefficients (columns of H).
    - The matrices W and H are determined by **minimizing the Frobenius norm**: $||A–WH||^2$.
        - In the minimization problem's solution, optimization is carried out by an iterative search process and the solution may be a local minimum.
        - Therefore based on your initial point, NMF may end up with different solutions.
    - NMF **works best with sparse matrices**.

# Non-Negative Matrix Factorization (NMF)

- How do we assessing the stability of the clusters obtained for a given rank (k)?
  - We get to make multiple independent NMF runs, and obtain different clusters (ie. solutions).
  - We need to achieve **a consensus** between these runs.
- We first define a **connectivity matrix C** of a given partition of a set of samples.
  - C contains only 0 or 1 entries such that: $C_{ij} = 1$ if sample i belongs to the same cluster as sample j, 0 otherwise.
- The **consensus matrix** is the average connectivity matrix of multiple NMF runs.
  - From a **frequentist perspective**, the entries of consensus matrix may be considered as **a "probability" that columns, i and j will belong to the same cluster after an NMF run**.
  - Then we can use this consensus matrix with **any clustering method based on distances.**
  - A very typical choice of distance is **$1.0 - C_{ij}$** although it is not a mathematically nice thing to treat a probability as distance.

# Non-Negative Matrix Factorization (NMF)

- NMF is similar to PCA in the sense that it does not need to explain anything.
- So what wo we achieve using NMF (ie. finding W)?
  - NMF is used for **unsupervised clustering with a selected number of clusters (k)**.
  - Each column of the generated W matrix represents a cluster.
    - The row values in these columns denote the features.
- In practice
  - NMF has been used to perform document clustering, making recommendations, visual pattern recognition such as face recognition, gene expression analysis, feature extraction, source separation etc.
  - Basically, it can be used in any application where data matrix A has no negative elements.
  - NMF **also works when your data points are themselves vectors or matrices**. Therefore it is suitable for text, image and video processing.

# Non-Negative Matrix Factorization (NMF)

- Selected Examples
  - Python (Topic modeling with a text data set) - https://bit.ly/3O9PxPZ
  - Python (Image classification, comparison with other techniques) - https://bit.ly/3xkOo21
  - R (image compression) - https://bit.ly/3O7cenP
  - R (detailed example using heatmaps with NMF) - https://bit.ly/37NpWvv

# Non-Negative Matrix Factorization (NMF)

- Homework Assignment #4 (Due Saturday, May 7th)
  - Check data set from course Github repo.
  - You will also submit through Github as usual. In this assignment you will submit **a report** in addition to your code.
  - You will be given **anonymized exam and homework scores** from a large number of students.
    - The exams and homework assignments were actually given weights in grading but you will not be given those weights, nor the letter grades.
  - Part 1: You are **required to cluster these students based on their exam and homework scores (as their feature values) using NMF with multiple runs**.
    - You should use a larger value for k, and then move towards a smaller number.
    - For each k you will try to explain these clusters (ie. why do you think they were together) in a report format.
    - When you can explain all clusters, stop trying for a different k.
    - Then try to assign letter grades to each cluster.
    - If you cannot assign a single letter grade to a cluster but need to assign multiple letter grades, explain why.

# Non-Negative Matrix Factorization (NMF)

- Homework Assignment #4 (Due Saturday, May 7th)
  - Part 2: Analyze the data using ICA.
    - First make sure that the exam scores are independent. Use Shapiro-Wilk test.
      - If they are not independent, try to explain why.
    - Then plot the heatmap for the exam scores.
  - Part 3: Analyze the data using PCA.
    - Try to explain why you get the number of principle components.
  - Part 4: Using this data set, could we in theory devise a better measurement system for this particular course (ie. number of homeworks, and exams)? Please elaborate.

# Questions?

CONTACT:
bora.gungoren@atilim.edu.tr