CS 217: Artificial Intelligence and Machine Learning **Jan-Apr 2024**

Lecture 18: Supervised Dimensionality Reduction

Lecturer: Swaprava Nath *Scribe(s): SG35 and SG36*

18.1 Introduction

We continue with the problem of dimensionality reduction. We are given some input data, with dimension d. The problem we want to solve is whether we can reduce the dimension of the data to some $m < d$, while preserving a lot of the information.
In the last lecture, we saw one method of doing this called Principal Component Analysis (PCA). This method tries to maximise the variance of the projections. The problem with this approach is that PCA does not use class data and is a blind approach. For example consider the following data (red and green are two different classes)

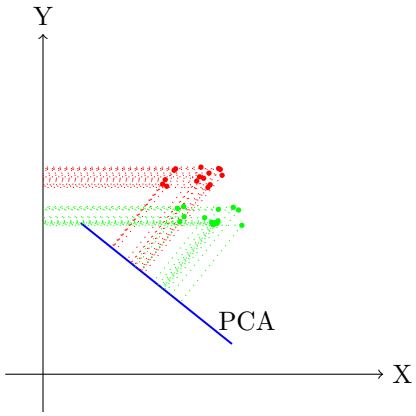


Figure 18.1: Projection of classes of data points onto various lines

Here, we see that the variance of the projections of the data points on the PCA line is more than the variance of the projection of the data points on the Y axis. Yet the problem with it is the fact that there is some overlap between projections of data points of the red and green classes. We want the following objectives:

1. The data of the two classes should be well separated.

2. Within a class, the data should NOT be well separated.

Now, we shall demonstrate another method for dimensionality reduction that utilises the class information called Linear Discriminant Analysis (LDA).

18.2 Setup for Optimisation Problem

The dataset D is given by $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, where $\mathbf{x_i} \in \mathbb{R}^d$ and we know that each $\mathbf{x_i}$ belongs to a class c_j. For now, let us assume that there are only two classes c_1 and c_2. Here, d is usually quite large. Our goal is to construct a linear map from the data D, onto a dimension $m < d$ such that the objectives are satisfied.

1. The means of the two classes (let us call them c_1 and c_2 for brevity) should be well separated. We know that the magnitude of the projection of a vector $\mathbf{x}$ onto a direction vector $\mathbf{u}$, with magnitude 1, is given by $\mathbf{x}^T\mathbf{u}$. Therefore, we define the distance between the two classes as $J(\mathbf{u}) =\mid \boldsymbol{\mu_1}^T\mathbf{u} - \boldsymbol{\mu_2}^T\mathbf{u} \mid$ where $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ represent the means of c_1 and c_2 respectively.
2. The variance within a class (also called "scatter") should be minimized

18.3 LDA for 2 Classes

Let C_i be the set of points $\mathbf{x_i}$ that belong to class i. For now, since there are 2 classes, $i \in \{1, 2\}$.
We define S_i as follows (note that $\mid C_i \mid$ is the cardinality of the set C_i)

$$S_i = \frac{1}{\mid C_i \mid} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \boldsymbol{\mu_i})(\mathbf{x} - \boldsymbol{\mu_i})^T$$

Note that, as we have seen in the previous lecture, S_i is the covariance matrix for the class i. Also, as we saw in the previous lecture, the variance of the projected data points of the class i on the vector $\mathbf{u}$ is given by $\mathbf{u}^T S_i \mathbf{u}$. The sum of the "within-class" variances $= \mathbf{u}^T S_1 \mathbf{u} + \mathbf{u}^T S_2 \mathbf{u} = \mathbf{u}^T (S_1 + S_2)\mathbf{u}$.
We define $S_W = S_1 + S_2$ and thus our job is to minimise $\mathbf{u}^T S_W \mathbf{u}$
Another objective that we have is to maximise the variance between the means of the two classes-
Variance $= (\mathbf{u}^T\boldsymbol{\mu_1} - \mathbf{u}^T\boldsymbol{\mu_2})^2 = \mathbf{u}^T(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^T\mathbf{u} = \mathbf{u}^T S_B \mathbf{u}$

18.4 LDA as an Optimization Problem

Our optimization problem is (we want to maximise $J(\mathbf{u})$)

$$J(\mathbf{u}) = \frac{\mathbf{u}^T S_B \mathbf{u}}{\mathbf{u}^T S_W \mathbf{u}}$$

We see that $J(\mathbf{u})$ is scale invariant, i.e. $J(\boldsymbol{\lambda}\mathbf{u}) = J(\mathbf{u})$
Our goal is to maximise $J(\mathbf{u})$. Now, written in this form, $J(\mathbf{u})$ is not very useful to analyze, so we try and convert it into an equivalent optimisation problem.
Our new optimisation problem is

$$\max_{\mathbf{u}} \mathbf{u}^T S_B \mathbf{u} \quad \text{s.t} \;\; \mathbf{u}^T S_W \mathbf{u} = 1$$

18.4.1 Solving the Optimisation Problem

We use the method of Lagrange multipliers and eigenvector decomposition to solve this. We assume S_W is invertible. Let the Lagrange multiplier be λ

$$L(\lambda, \mathbf{u}) = -\mathbf{u}^T S_B \mathbf{u} + \lambda(\mathbf{u}^T S_W \mathbf{u} - 1)$$

The necessary condition for duality is that the partial derivative of L with respect to the primal variable is 0

$$\begin{aligned} \frac{\delta L}{\delta \mathbf{u}} &= 0 \\ \implies -2S_B\mathbf{u} + 2\lambda S_W\mathbf{u} &= 0 \\ \implies S_B\mathbf{u} &= \lambda S_W\mathbf{u} \\ \implies S_W^{-1}S_B\mathbf{u} &= \lambda\mathbf{u} \end{aligned}$$

Now the optimisation problem becomes

$$\max_{\mathbf{u}} \mathbf{u}^T S_B\mathbf{u} = \max \mathbf{u}^T\lambda S_W\mathbf{u} = \max \lambda(\mathbf{u}^T S_W\mathbf{u}) = \max \lambda$$

Also, due to the above equation, we can conclude that λ is an eigenvalue of $S_W^{-1}S_B$
Hence, we should project the data points on a direction which is an eigenvector corresponding to the maximum eigenvalue of $S_W^{-1}S_B$
Note that we can represent the eigenvector decomposition of $S_W^{-1}S_B$ as follows

$$S_W^{-1}S_B = V\Sigma V^T$$

Here the columns of V are the eigenvectors of $S_W^{-1}S_B$ and Σ is a diagonal matrix with values corresponding to the eigenvalues of $S_W^{-1}S_B$. Assuming the eigenvalues are in descending order, i.e. the first eigenvalue is the maximum, we want to project the data on the first eigenvector of V

18.5 MultiClass LDA

So far, our analysis has been restricted to only 2 classes. Now we analyze what would happen to the problem if we had $c > 2$ classes.
So, in our problem now we have c classes and sets C_i, which is the set of all data points that belong to the class i, $i \in \{1, 2 \cdots c\}$
Firstly, we note that S_W generalises pretty easily to c classes.

$$S_W = S_1 + S_2 \cdots S_c$$

where S_i are defined as the variance of the class i as before.
Now, we analyse how S_B generalises to c classes. The new formula for S_B is as follows -

$$S_B = \sum_{i=1}^{c} n_i(\boldsymbol{\mu_i} - \boldsymbol{\mu})(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T$$

where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{c} n_i\boldsymbol{\mu_i}$ and $n_i = \mid C_i \mid$
The rest of the problem remains pretty similar to the one class problem.

$$J(\mathbf{u}) = \frac{\mathbf{u}^T S_B\mathbf{u}}{\mathbf{u}^T S_W\mathbf{u}}$$

The optimal solution again reduces to the following-

$$S_W^{-1}S_B\mathbf{u} = \lambda\mathbf{u}$$

18.5.1 Projecting Onto Higher Dimensions

Let's suppose we wanted to project our data points on a k dimensional space. What we would do in PCA is select the eigenvectors corresponding to the top k eigenvalues of $S_W^{-1}S_B$.
But, we shall show that it is not possible to project our data points on a k dimensional space where $k \geq c$. For this, we need to analyse S_B. In particular, we want to analyse the rank of S_B. The rank of a matrix is the maximum number of linearly independent columns of the matrix.

$$S_B = \begin{bmatrix}\sqrt{n_1}(\boldsymbol{\mu_1} - \boldsymbol{\mu}) & \sqrt{n_2}(\boldsymbol{\mu_2} - \boldsymbol{\mu}) & \cdots & \sqrt{n_c}(\boldsymbol{\mu_c} - \boldsymbol{\mu})\end{bmatrix} \begin{bmatrix}\sqrt{n_1}(\boldsymbol{\mu_1} - \boldsymbol{\mu})^T \\ \sqrt{n_2}(\boldsymbol{\mu_2} - \boldsymbol{\mu})^T \\ \vdots \\ \sqrt{n_c}(\boldsymbol{\mu_c} - \boldsymbol{\mu})^T\end{bmatrix} = AA^T$$

where A is given as follows -

$$A = \begin{bmatrix}\sqrt{n_1}(\boldsymbol{\mu_1} - \boldsymbol{\mu}) & \sqrt{n_2}(\boldsymbol{\mu_2} - \boldsymbol{\mu}) & \cdots & \sqrt{n_c}(\boldsymbol{\mu_c} - \boldsymbol{\mu})\end{bmatrix}$$

Now, we state a property that we shall use in our analysis - rank(XY) = min(rank(X),rank(Y))
First, we aim to find the rank of A. We shall show that rank(A) $\leq c - 1$
To see this, we will simply show that all the columns of A are linearly dependent.
The set of vectors $\mathbf{x}_1, \mathbf{x}_2 \cdots \mathbf{x}_c$ are linearly dependent if there exist $\gamma_1, \gamma_2, \cdots \gamma_c$ such that they are not all 0 and

$$\sum_{i=1}^{c} \gamma_i \mathbf{x_i} = 0$$

Here, if we choose γ_i to be $\sqrt{n_i}$ we can see that

$$\sum_{i=1}^{c} \gamma_i \mathbf{x_i} = \sum_{i=1}^{c} \sqrt{n_i}(\sqrt{n_i}(\boldsymbol{\mu_i} - \boldsymbol{\mu})) = \sum_{i=1}^{c} n_i(\boldsymbol{\mu_i} - \boldsymbol{\mu}) = \mathbf{0}$$

Hence rank(A) $\leq c - 1$. Now, since $S_B = AA^T$, rank(S_B) $\leq c - 1$ as well.
Hence rank($S_W^{-1}S_B$) $\leq c - 1$ and hence we can find at most $c - 1$ discriminatory directions.

18.6 LDA Algorithm

Algorithm 1 Linear Discriminant Analysis (LDA)

1: **Input:** Training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ is the feature vector and y_i is the class label.
2: Compute the mean vectors μ_j for each class c_j.
3: Compute the within-class scatter matrix S_W and the between-class scatter matrix S_B.
4: Compute the eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d\}$ and corresponding eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$ of $S_W^{-1}S_B$.
5: Sort the eigenvectors according to their eigenvalues in descending order.
6: Choose the k eigenvectors corresponding to the k largest eigenvalues to form the transformation matrix $\mathbf{U} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$.
7: Project the input data (X) onto the new subspace: $\mathbf{U^T} \cdot \mathbf{X}$.

18.7 Artificial Intelligence: A Dual Approach

Artificial Intelligence (AI) is a vast field that encompasses a wide range of approaches. These can be broadly categorized into those inspired by human cognition (Human-Inspired Approach) and those based on rational decision-making principles (Rational Approach).

Aspect	Human-Inspired Approach	Rational Approach
Thinking	• Natural Language Processing (NLP) • Computer Vision • Automated Reasoning	• Logician's Approach • Assumes Complete Information and Struggles with Uncertainties
Acting	• Cognitive Science – Studies Brain's Functions	• Agent-Based Approach – Single Agent Systems – Multi-Agent Systems

Table 18.1: Comparison of Human-Inspired and Rational Approaches in AI

Multi-agent AI, a subset of the rational approach, represents systems where multiple AI entities interact. This adds a new level of complexity and potential to the field.

This leads us to an important question: What does it mean to be rational? In the context of AI, rationality primarily refers to the ability to make decisions based on reason.

The decisions depends on several factors:

- **Performance measure:** The criteria used to evaluate the success of an AI agent's actions.
- **Agent's prior knowledge:** The information an AI agent has about its environment and other agents.
- **Available actions:** The set of possible actions an AI agent can take. The more actions available, the better the performance measure we can select.
- **History:** The record of past states or actions, which can influence current decisions.

In different fields of AI, rationality takes on different forms:

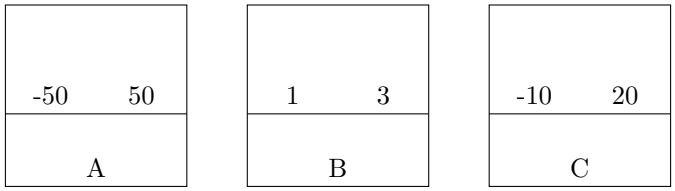
Machine Learning: The goal is to minimize a loss function, which serves as the performance measure. The model learns from historical data (prior knowledge) to make predictions or decisions (actions).

Robotics: Reinforcement learning is often used, where an agent learns to perform actions based on a reward function. The agent's aim is to maximize the total reward over time.

Multi-Agent Systems: In systems with two or more agents, each agent seeks to maximize its own utility function. The utility function quantifies the agent's preference for each possible outcome, guiding its decision-making process.

18.7.1 Two-Player Game

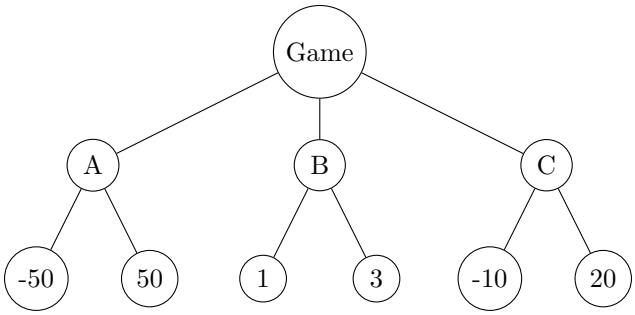
Consider a two player game in which there are 3 containers as shown below and in each container there are 2 numbers.



The Rules of the game are as follows,

- You choose one of the bins
- Then your opponent chooses one number from that bin
- Your Preference/ Utility is the number picked

Consider these 2 strategies: Opponent is adversarial choice or opponent may be random
Let's first construct a game tree



If the opponent chooses the second strategy then we should choose the container "C" as it has the highest expectation value among all the other containers.
And if the opponent chooses the first strategy then we should choose the container "B" as it is the box with the lowest minimum value.