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

Lecture 17: Dimensionality Reduction

Lecturer: Swaprava Nath *Scribe(s): SG33, SG34*

Disclaimer: *These notes aggregate content from several texts and have not been subjected to the usual scrutiny deserved by formal publications. If you find errors, please bring to the notice of the Instructor.*

17.1 Dimensionality reduction

Dimensionality reduction involves condensing dataset information while preserving essential characteristics. The primary objective is to reduce algorithmic complexity and enhance model performance by removing redundant or irrelevant features without losing much information. Suppose 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, without losing, or compromising a lot of the information.
We have two methods to achieve this:

- Unsupervised: D = $\{x_i\}$ $\implies$ Principal Component Analysis (PCA)
- Supervised: D = $\{x_i, y_i\}$ $\implies$ Linear Discriminant Analysis

17.2 Principle Component Analysis

Principal Component Analysis (PCA) is a fundamental technique that helps extract the most informative features and transform high-dimensional data into a more manageable low-dimensional form. It identifies a set of orthogonal axes called principal components (PCs) that capture the maximum variance upon orthogonal projection of the data onto these axes.
Say we have some data as shown in the figure. Let the vectors be u_1 and u_2 onto which projections are done.

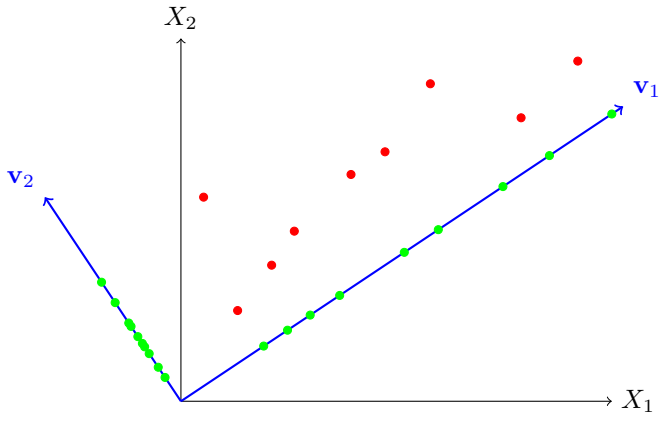

Figure 17.1: PCA illustration

We can say that v1 is a better basis vector than v2 since the variations among the data points in the original dataset are best depicted by the projections on v1 rather than v2.

Consider a dataset $D = \{x_1, x_2, ..., x_n\}, x_i \in \mathbb{R}^d$
Our aim is to map dataset D onto m dimensions where $m < n$.
The question becomes which vectors to choose as our new basis vectors. To be able to capture the most important trends or patterns among the given data points, they have to be projected onto a vector that **maximises the variance of projected data**

17.2.1 Optimisation

Here, we're interested in getting a unit vector $\boldsymbol{u_1} \in \mathbb{R}^d$, such that the variance of the projected data is maximum. And this vector will be our first PC. So here, for the sake of notation, we define the projection of $\boldsymbol{x_1}$ *on* $\boldsymbol{u_1}$ as :

$$\mathbf{Proj_{u_1}}\left(\mathbf{x_1}\right) = \left(\mathbf{x_1^T}.\mathbf{u_1}\right)\mathbf{u_1} = \left(\mathbf{u_1^T}.\mathbf{x_1}\right)\mathbf{u_1}$$

where $\boldsymbol{u_1}$ is a unit vector
Now, to calculate the variance, we get the mean of the projected data as:

$$\begin{aligned}\text{Mean of the projected data on } \mathbf{u_1} &= \frac{1}{n}\sum_{i=1}^{n}\text{Proj}_{\mathbf{u_1}}(x_i)\\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{u_1}^T.\bar{x_i}\right)\mathbf{u_1}\\ &= \left(\mathbf{u_1}^T.\bar{x}\right)\mathbf{u_1}\end{aligned}$$

Now, we find the variance of the projected data. We know that each of the projected data points is $\left(\mathbf{u_1^T}.\mathbf{x_i}\right)\mathbf{u_1}$. Since all projected points are in the direction of $\boldsymbol{u_1}$, and $\boldsymbol{u_1}$ is a unit vector, we can therefore write variance in terms of distance of $\left(\mathbf{u_1^T}.\mathbf{x_i}\right)\mathbf{u_1}$ from the $\left(\mathbf{u_1^T}.\bar{\mathbf{x}}\right)\mathbf{u_1}$ as :

$$\begin{aligned}\text{Variance of the projections} &= \frac{1}{n}\sum_{i=1}^{n}\left(u_1^T\cdot x_i - u_1^T\bar{x}\right)^2\\ &= \frac{1}{n}\sum_{i=1}^{n}u_1^T\left(x_i-\bar{x}\right)\left(x_i-\bar{x}\right)^T u_1\\ &= u_1^T S u_1 \qquad\qquad \text{, where } S = \frac{1}{n}\sum_{i=1}^{n}\left(x_i-\bar{x}\right)\left(x_i-\bar{x}\right)^T\end{aligned}$$

Here, we take a small digression to take a note of $\mathbf{S}$, we can write $\mathbf{X} = [\mathbf{x_1} - \bar{\mathbf{x}}\ \mathbf{x_2} - \bar{\mathbf{x}}\ ...\ \mathbf{x_n} - \bar{\mathbf{x}}]$, where $\boldsymbol{x_i}$*'s and* $\bar{\boldsymbol{x}} \in \mathbb{R}^d$. We have $\boldsymbol{S'} = \mathbf{XX^T}$, and we get that

$$\mathbf{S'(i,j)} = \sum_{\mathbf{k=1}}^{\mathbf{n}}\left(x_{ki}-\bar{x_i}\right)\left(x_{kj}-\bar{x_j}\right)$$

Now, also we have,

$$\mathbf{S(i,j)} = \frac{\mathbf{1}}{\mathbf{n}}\sum_{\mathbf{k=1}}^{\mathbf{n}}\left(x_{ki}-\bar{x_i}\right)\left(x_{kj}-\bar{x_j}\right)$$

Therefore, we have $\mathbf{S} = \frac{\mathbf{1}}{\mathbf{n}}\mathbf{X}\mathbf{X}^{\mathbf{T}}$, hence $\mathbf{S}$ is covariance matrix, of the vectors $\boldsymbol{x_1, x_2, ..., x_n}$

Now, we are back to the original discussion, that is, getting a unit vector $\boldsymbol{u_1}$ such that the variance of the projected data is maximum. So far, we have reduced the optimisation to the following problem:

$$\max_{\mathbf{u_1}} \mathbf{u_1^T S u_1}$$

$$st\ \boldsymbol{u_1^T u_1 = 1}$$

Now the question comes: is it a convex optimization problem? Here the optimisation function $f(\boldsymbol{u_1}) = \boldsymbol{u_1^T S u_1}$, is convex because $\nabla^2 f(\boldsymbol{u_1}) = \boldsymbol{2S = \frac{2}{n} X X^T}$ is positive semi definite. The problem is not convex because the feasible region itself is not convex, and instead of minimizing it, we are maximizing it. So, it looks like we will find another way out.

17.2.1.1 Orthonormal Eigenvectors

Before we state the main theorem that we are going to use, we first introduce a few terminologies:

- For a matrix $\boldsymbol{A}$, we say a vector $\boldsymbol{v}$ to be eigenvector with eigenvalue λ, if following holds:

$$\boldsymbol{Av} = \lambda \boldsymbol{v}$$

- If two vectors $\boldsymbol{v}, \boldsymbol{u}$, satisfies :

$$\boldsymbol{v^T.v = 1}\ ,\ \boldsymbol{u^T.u = 1} \text{ and } \boldsymbol{v^T.u = u^T.v = 0}$$

  then we say, the vectors $\boldsymbol{v}, \boldsymbol{u}$ as orthonormal vectors.

Theorem (*Spectral theorem for real matrices*)

A real symmteric martix $\underset{d\times d}{\boldsymbol{A}}$ has a **orthonormal** basis of $\mathbb{R}^d$, formed by eigenvectors of $\boldsymbol{A}$.
Although we ain't going to use the actual version of the theorem, instead, we use the equivalent version, which states that *every real symmetry matrix* $\underset{d\times d}{\boldsymbol{A}}$ *has d orthonormal eigenvectors.*

17.2.1.2 Optimising the reduced problem

Here, we can observe that S is a real symmetry matrix,

$$\boldsymbol{S^T} = \left(\frac{\mathbf{1}}{\boldsymbol{n}}\boldsymbol{X X^T}\right)^{\boldsymbol{T}} = \frac{\mathbf{1}}{\boldsymbol{n}}\left(\boldsymbol{X X^T}\right)^{\boldsymbol{T}} = \frac{\mathbf{1}}{\boldsymbol{n}}\left(\boldsymbol{X^T}\right)^{\boldsymbol{T}}\boldsymbol{X^T} = \frac{\mathbf{1}}{\boldsymbol{n}}\boldsymbol{X X^T} = \boldsymbol{S}$$

Now, using the theorm stated above, we can say that $\exists\ \boldsymbol{v_i}, \lambda_i$ such that ,

$$\boldsymbol{S v_i} = \lambda_i \boldsymbol{v_i} \qquad \text{where } \begin{cases} \boldsymbol{v_i^T.v_j} = 0 & \text{if } i \neq j,\ i,j \in [1,2,...,d] \\ \boldsymbol{v_i^T.v_i} = 1 & \text{if } i \in [1,2,...,d] \end{cases}$$

$$\text{for } i = 1,2,...,d$$

Which can be equivalently rewritten as:

$$\mathbf{S}\begin{bmatrix}\mathbf{v_1} & \mathbf{v_2} & \cdots & \mathbf{v_d}\end{bmatrix} = \begin{bmatrix}\lambda_1\mathbf{v_1} & \lambda_2\mathbf{v_2} & \cdots & \lambda_d\mathbf{v_d}\end{bmatrix}$$
$$= \begin{bmatrix}\mathbf{v_1} & \mathbf{v_2} & \cdots & \mathbf{v_d}\end{bmatrix}\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_d\end{bmatrix}$$

Denoting $\begin{bmatrix}\boldsymbol{v_1} & \boldsymbol{v_2} & \cdots & \boldsymbol{v_d}\end{bmatrix}$ by $\boldsymbol{V}$, and by the condition that $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_d}$ are orthonormal vectors we have that

$$\boldsymbol{V^T V = I}$$

we have,

$$\boldsymbol{SV = V\Sigma}$$

where,

$$\boldsymbol{\Sigma} = \begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & ... & \lambda_d\end{bmatrix}$$

Therefore,

$$\boldsymbol{S = V\Sigma V^T}$$

Now, putting this back into the reduced optimization problem, we have the following optimisation problem:

$$\max_{u_1} \boldsymbol{u_1^T V\Sigma V^T u_1}$$

$$\text{st } u_1^T u_1 = 1$$

Since the vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_d}$ form a basis of $\mathbb{R}^d$ $\exists\ \alpha_1, \alpha_2, ..., \alpha_d$ such that $\boldsymbol{u_1} = \alpha_1\boldsymbol{v_1} + \alpha_2\boldsymbol{v_2} + ... + \alpha_d\boldsymbol{v_d}$ which is equivalently,

$$\boldsymbol{u_1 = V\alpha^T} \text{ where } \boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_d\end{bmatrix}$$

$\therefore$ Our optimisation problem reduces as:

$$\max_{\alpha} \boldsymbol{\alpha}\left(\boldsymbol{V^T V}\right)\boldsymbol{\Sigma}\left(\boldsymbol{V^T V}\right)\boldsymbol{\alpha^T} \qquad \text{st } \boldsymbol{\alpha}\left(\boldsymbol{V^T V}\right)\boldsymbol{\alpha^T}$$
$$\max_{\alpha} \boldsymbol{\alpha\Sigma\alpha^T} \qquad \text{st } \boldsymbol{\alpha\alpha^T}$$
$$\max_{\alpha} \sum_{i=1}^{d} \alpha_i^2\lambda_i \qquad \text{st } \sum_{i=1}^{d}\alpha_i^2 = 1$$

Suppose, $\lambda_j = \max(\lambda_1, \lambda_2, ..., \lambda_d)$, then clearly the solution to our optimisation is $\boldsymbol{\alpha}$ such that $\alpha_i = \mathbb{1}(i = j)\ \forall\ i\ \in [1, 2, ..., d]$, we summarise the result for first **PC** $\boldsymbol{u_1}$ as follow:

$$\max_{\boldsymbol{u_1}}\ \boldsymbol{u_1^T S u_1} \text{ st } \boldsymbol{u_1^T . u_1} = 1 \text{ and } \boldsymbol{S = S^T}$$
$$\implies \boldsymbol{u_1 = v_j} \text{ where } \boldsymbol{\lambda_j} = \max\boldsymbol{(\lambda_1, \lambda_2, ..., \lambda_d)} \text{ and } \boldsymbol{Sv_i = \lambda_i v_i}\ \forall\ i\ \in [1, 2, ..., d]$$

17.2.2 Generalisation for reduction to m dimensions

After we have the first PC $\boldsymbol{u_1}$, we can find the second PC $\boldsymbol{u_2}$, by solving the following optimisation problem:

$$\max_{\boldsymbol{u_1}} \ \boldsymbol{u_2^T S u_2} \text{ st } \boldsymbol{u_2^T . u_2} = 1 \text{ and } \boldsymbol{u_1^T . u_2 = 0}$$

$$\implies \boldsymbol{u_2} = \boldsymbol{v_k} \text{ st } \lambda_k = \max\{\{\lambda_1, \lambda_2, ..., \lambda_d\} \setminus \{\lambda_j\}\}$$

$$\text{where } \lambda_j = \max\{\lambda_1, \lambda_2, ..., \lambda_d\}, \boldsymbol{S v_i = \lambda_i v_i} \ \forall \ i \ \in [1, 2, ..., d]$$

In other words, if we order the eigenvectors according to their eigenvalues, then $\boldsymbol{u_1}$ is the largest eigenvector, and $\boldsymbol{u_2}$ is the second largest eigenvector of $\boldsymbol{S}$.
To generalize in the end, to reduce to m dimension, we just take the projection of the data points to the first m eigenvectors of $\boldsymbol{S}$, where eigenvectors are being ordered for the eigenvalues. In a nutshell, take the first m largest eigenvectors, say $\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_m}$, and then to reduce the dimension to m, take the projection of each of the data points $x_i \in \mathbb{R}^d$, in $Span\{\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_m}\}$.

17.2.3 PCA Algorithm

1. Compute mean of data points, $\bar{x}$
2. Mean centre the data i.e., compute $x_i - \bar{x}, \forall i$
3. Compute the covariance matrix, $S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$
4. Do eigenvalue decomposition of $S = V\Sigma V^T$
5. Pick m top eigenvectors (corresponding to largest eigenvalues): $u_1, ...u_m \rightarrow$ Principal Components
6. Create a projection matrix, $U = [u_1, ...u_m]$
7. For each $x_i \in D$, $U^T x_i$ will give the projection on m-dimensional space

Note that for $U^T x$ is a m-dimensional vector.

$$U^T x = \begin{bmatrix} u_1^T x \\ u_2^T x \\ \vdots \\ u_m^T x \end{bmatrix}$$

17.3 Supervised Learning

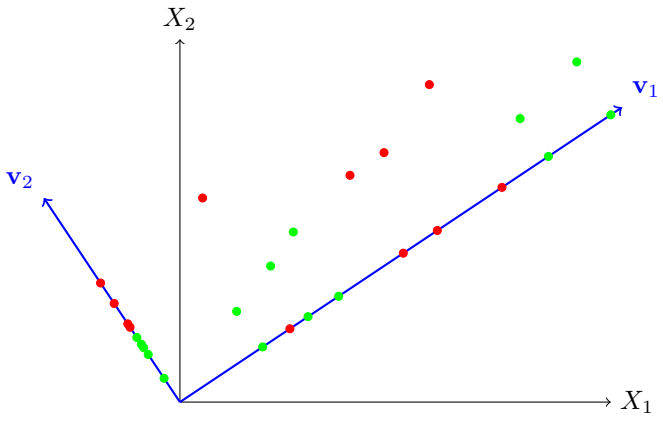

Figure 17.2: PCA's Blindness towards Class Discrimination

Utilizing the vector v_1 from PCA for data projection results in the loss of class distinctions. Thus, employing vector v_2 would be more advantageous in this context.
As we can see, PCA fails to preserve the intrinsic class characteristics. We need a projection that provides us with more resolution between classes.
We need a good measure to evaluate these projections.

$$J(u) = |\mu_1^T u - \mu_2^T u| \text{ where } \mu_i \text{ is the mean of class } i$$

$J(u)$ is one possible measure, but it alone is not good enough since it only considers the means of various classes.

Objective:

- Different classes are well separated.
- Data within the same class is not well separated.