Artificial Intelligence Algorithms and Mathematics

CSCN 8000

Classification

- Solutions for Overfitting
- PCA
- FDA/LDA

Ways to reduce overfitting

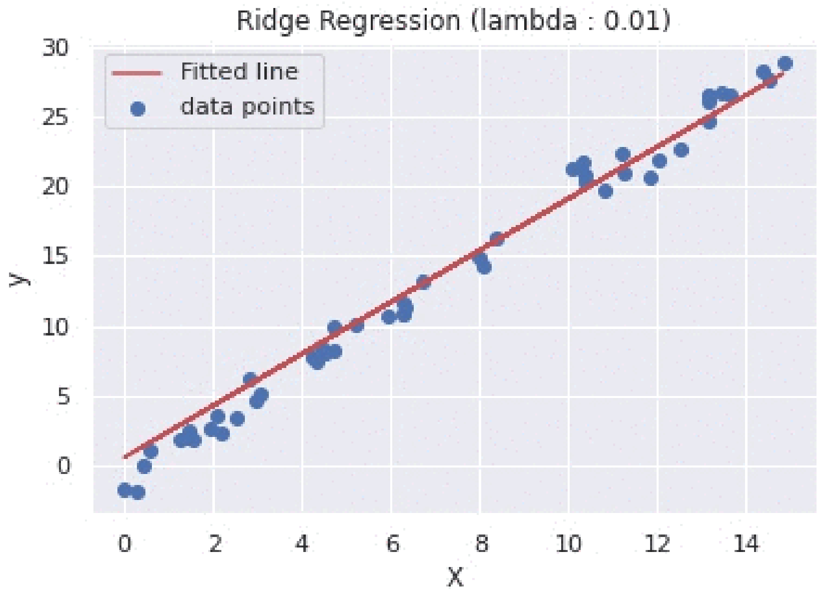
- Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise and random fluctuations
- To reduce the overfitting, a general rule of thumb is to try to decrease the model complexity, but not too much to avoid underfitting.
- Specifically, techniques that can be used include:
  - **Cross-Validation:** Assess the model's performance on multiple subsets of the data to ensure good generalization to different portions of the dataset.
  - **Increase Dataset size:** A larger dataset can provide more diverse examples and help the model generalize better.
  - **Early Stopping:** Stop training once the performance on the validation set starts to degrade, preventing the model from overfitting the training data.
  - **Dropout:** random neurons are "dropped out" (ignored) during each training iteration. This helps prevent co-adaptation of neurons and reduces overfitting.
  - **Data Augmentation:** Increase the size of the training dataset by applying various transformations to the existing data
  - ***Regularization?***

Regularization

- The intuition is that one way to decrease the overall model complexity is to reduce the magnitude of the learned weights $\vec{\boldsymbol{w}}$. This way we won't be assigning very high weights to certain features over the others → Simpler model
- This could be achieved by adding a *regularization term* to the loss function of any machine learning algorithm, such that,

$$L_{reg}(\vec{\boldsymbol{w}}, \boldsymbol{b}) = L(\vec{\boldsymbol{w}}, \boldsymbol{b}) + [\boldsymbol{\beta} \, * Regularization\ Term]$$

- This will allow the model to not only prioritize minimizing the cost term, but also minimizing the magnitude of the weights (controlled by the hyperparameter $\boldsymbol{\beta}$)
- There are different types of regularization terms that could be used.

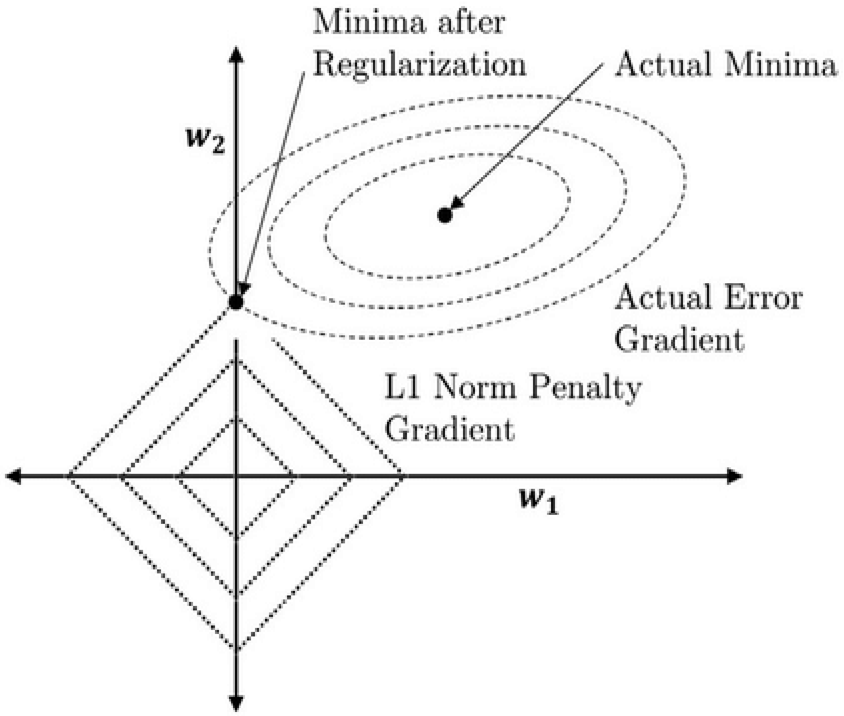


L1 Regularization (Lasso)

- L1 regularization introduces a penalty term proportional to the absolute values of the model parameters.
- Mathematically,

$$L_{reg}(\vec{\boldsymbol{w}}, \boldsymbol{b}) = L(\vec{\boldsymbol{w}}, \boldsymbol{b}) + \left[\boldsymbol{\beta} * \sum_{p} \left|\overrightarrow{\boldsymbol{w_p}}\right|\right]$$

- This encourages the model to prefer a sparse set of features, effectively driving some of them to exactly zero.
- Not only does it help prevent overfitting but also performs feature selection by excluding less relevant features.

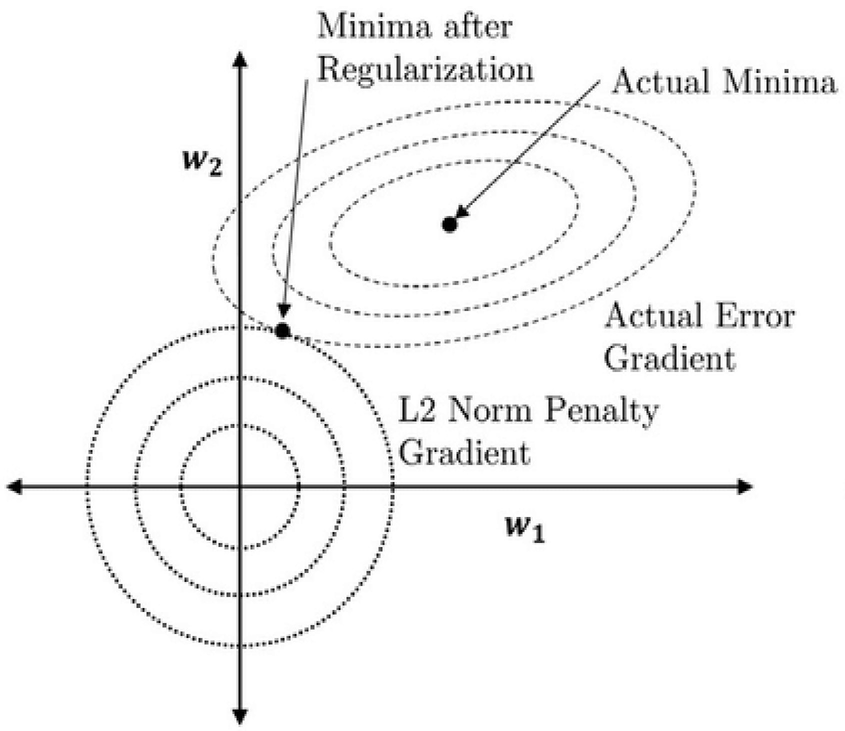


L2 Regularization (Ridge)

- L2 regularization introduces a penalty term proportional to the square of the model parameters.
- Mathematically,

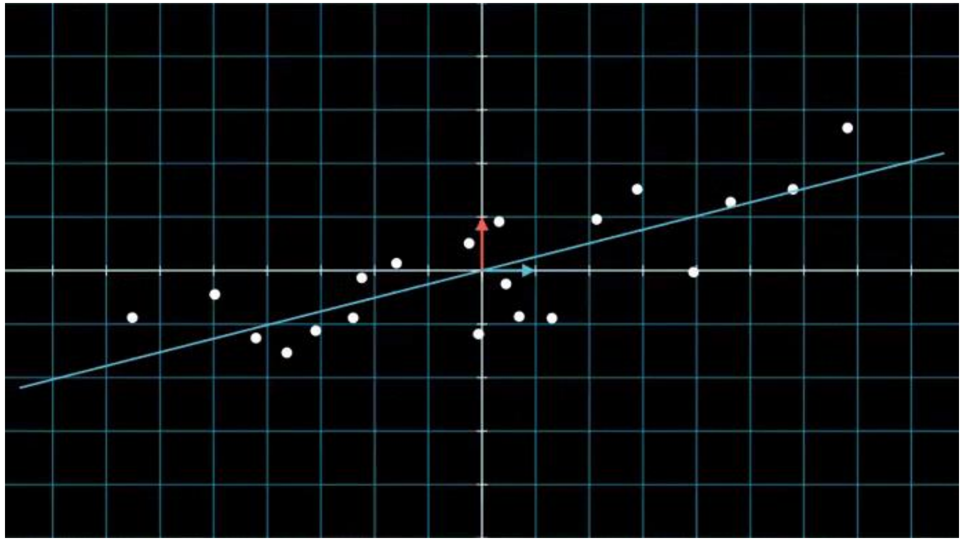
$$L_{reg}(\overrightarrow{\boldsymbol{w}}, \boldsymbol{b}) = L(\overrightarrow{\boldsymbol{w}}, \boldsymbol{b}) + \left[\boldsymbol{\beta} * \sum_{p} \left(\overrightarrow{\boldsymbol{w_p}}\right)^2\right]$$

- This discourages the weights from becoming too large, preventing individual features from dominating the model.
- It distributes the importance of features more evenly, leading to a smoother model.



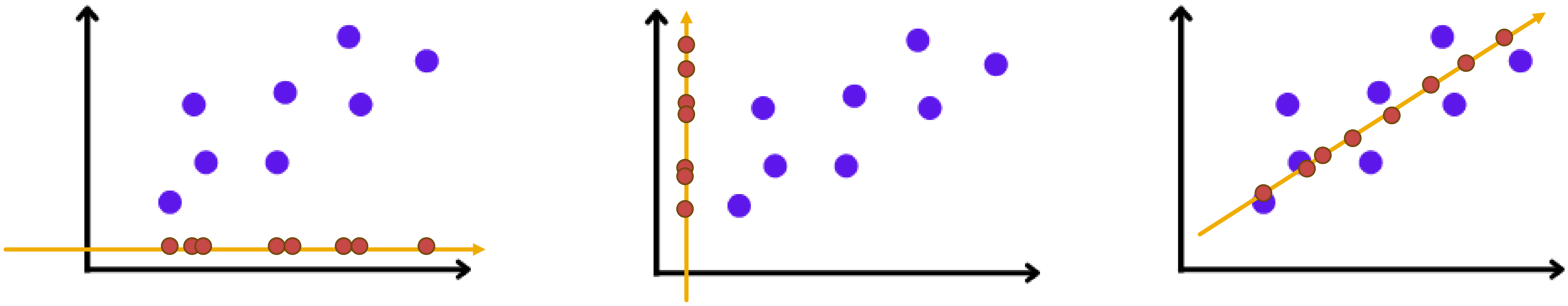
Dimensionality Reduction

- The efficiency of ML methods depends crucially on the choice of features that are used to characterize data points.
- Target → have a small number of highly relevant features to characterize data points.
- Dimensionality Reduction techniques reduce the number of input variables or features in a dataset *while retaining its essential characteristics*
- Benefits of dimensionality reduction:
  - Reduce excessive resource requirements
  - Reduce the probability of overfitting
  - Make data visualizations easier



Principal Component Analysis (PCA)

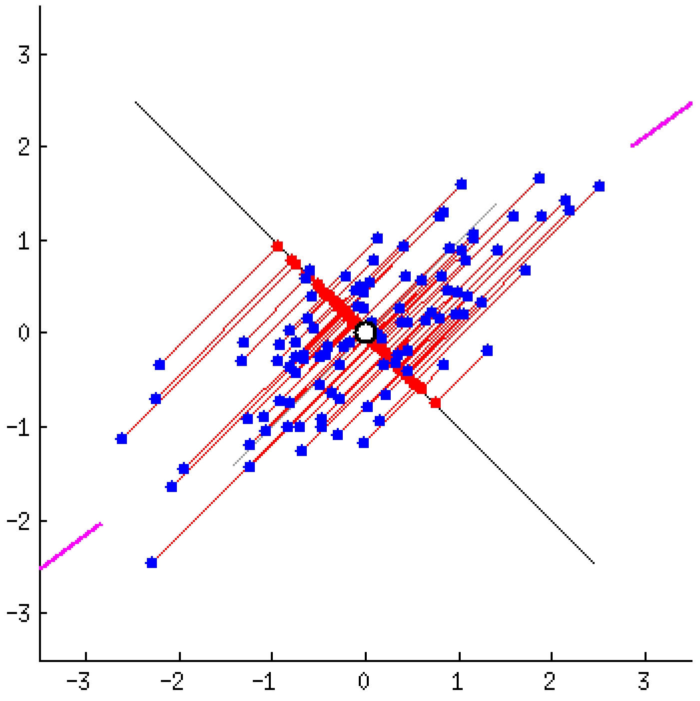
- Intuitively, if we want to project the following 2D points to only one dimension, which one of the following best preserves the shape of the dataset?



- Generally, it looks like in the best dimension, the projected data points have the *largest possible variance.*

Principal Component Analysis (PCA)

- The ***PCA*** is a dimensionality reduction technique that transforms high-dimensional data into a new coordinate system (principal components/axes).
- Target → **Maximize** the variance of projected data.
- Doesn't need any class labels (unsupervised)
- The projected data must obey certain properties:
  - Each new feature (principal component/axis) is a ***linear*** combination of the original features (axes)
  - Each principal component (axis) must be perpendicular to all other principal components (axes).
  - Each principal component (axis) must be a unit vector (magnitude = 1)



Principal Component Analysis (PCA)

- ***<u>Covariance Matrix:</u>*** provides important information about the relationships between pairs of variables in a dataset.
- Given a dataset $X \in R^{D*N}$ with D features and N rows, the covariance matrix will have size $D * D$ can be computed as follows:

$$C(X) = \frac{1}{N}(X - \bar{X})(X - \bar{X})^T$$

- $\bar{X}$ represents the mean of each feature in X.
- Cell C_{ij} represents the co-variance between feature i and feature j.
- If one feature increases/decreases and the other feature increases/decreases at the same time → $C_{ij} \neq 0$.
- If the features are not correlated → $C_{ij} = 0$.

Eigen Vectors and Eigen Values

- Given a vector v, apply linear transformation with square matrix A.

$$A.v = ?$$

$$A.v = \lambda v \rightarrow \text{[Special Case]}$$

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -1 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\lambda_1 = 3, \; v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = -1, \; v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Principal Component Analysis (PCA)

- Given the initial dataset with D features, we want to get the first axis of the new coordinate space.
- We will apply a linear transformation $u \in R^{D*1}$ on each original data point $x \in R^{D*1}$, such that the value z of the projected point at the new axis is formulated as,

$$z = u^T x$$

- Applying the same equation on the full dataset $X \in R^{D*N}$, such that,

$$Z = u^T X$$

- Target → **Maximize** the variance of projected data (1D) → **Maximize** the covariance matrix of projected data (Multi-Dimensional)

Principal Component Analysis (PCA)

- Target → **Maximize** the covariance matrix of projected data
- $Cov(Z) = \frac{1}{N}(Z - \bar{Z})(Z - \bar{Z})^T$
- $Cov(Z) = \frac{1}{N}(u^T X - u^T \bar{X})(u^T X - u^T \bar{X})^T$
- $Cov(Z) = \frac{1}{N}\left(u^T (X - \bar{X})\right)\left(u^T (X - \bar{X})\right)^T$ →Expand 2nd Transpose
- $Cov(Z) = \frac{1}{N}u^T (X - \bar{X})(X - \bar{X})^T u = u^T [\frac{1}{N}(X - \bar{X})(X - \bar{X})^T]u$
- $Cov(Z) = u^T S u, \;\; where\ S = Cov(X)$

Principal Component Analysis (PCA)

$$maximize\ u^T S u,$$
$$such\ that\ u^T u = 1$$

- The constraint guarantees that the axis u is a unit vector.
- By borrowing the concept of Lagrangian Multipliers, we will translate the equation to a loss function to be ***minimized***, such that,

$$L(u, \lambda) = -(u^T S u - \lambda(u^T u - 1)$$

- To minimize the loss with respect to u → Solve $\frac{dL}{du} = 0$

Matrix/Vector Rules and Derivatives

Rule	Comments
$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$	order is reversed, everything is transposed
$(\mathbf{a}^T\mathbf{Bc})^T = \mathbf{c}^T\mathbf{B}^T\mathbf{a}$	as above
$\mathbf{a}^T\mathbf{b} = \mathbf{b}^T\mathbf{a}$	(the result is a scalar, and the transpose of a scalar is itself)
$(\mathbf{A}+\mathbf{B})\mathbf{C} = \mathbf{AC}+\mathbf{BC}$	multiplication is distributive
$(\mathbf{a}+\mathbf{b})^T\mathbf{C} = \mathbf{a}^T\mathbf{C}+\mathbf{b}^T\mathbf{C}$	as above, with vectors
$\mathbf{AB} \neq \mathbf{BA}$	multiplication is **not** commutative

Scalar derivative			Vector derivative		
$f(x)$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}x}$	$f(\mathbf{x})$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{B}$	$\rightarrow$	$\mathbf{B}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{b}$	$\rightarrow$	$\mathbf{b}$
x^2	$\rightarrow$	$2x$	$\mathbf{x}^T\mathbf{x}$	$\rightarrow$	$2\mathbf{x}$
bx^2	$\rightarrow$	$2bx$	$\mathbf{x}^T\mathbf{B}\mathbf{x}$	$\rightarrow$	$2\mathbf{B}\mathbf{x}$

Principal Component Analysis (PCA)

$$L(u,\lambda) = -(u^T Su - \lambda(u^T u - 1)$$

- To minimize the loss with respect to u → Solve $\frac{dL}{du} = 0$
- $\frac{dL}{du} = -(2Su - 2\lambda u) = 0$

$$\boldsymbol{Su} = \boldsymbol{\lambda u} \rightarrow Result\ of\ \frac{dl}{du} = 0$$

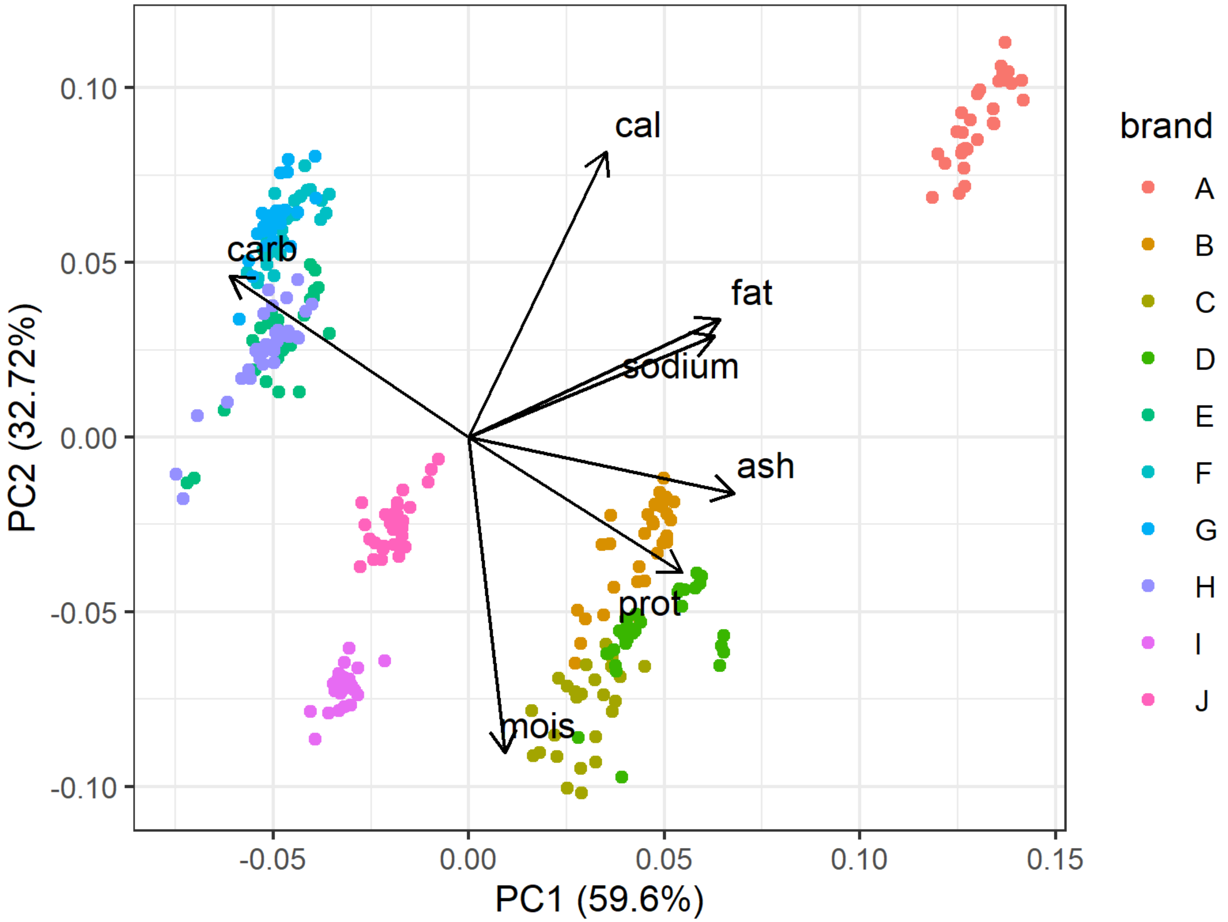
- We reach a formulation exactly similar to the one of *eigenvalues and eigenvectors*.
- In other words, u is considered an eigenvector of the covariance matrix S of the original data and λ is the associated eigenvalue.

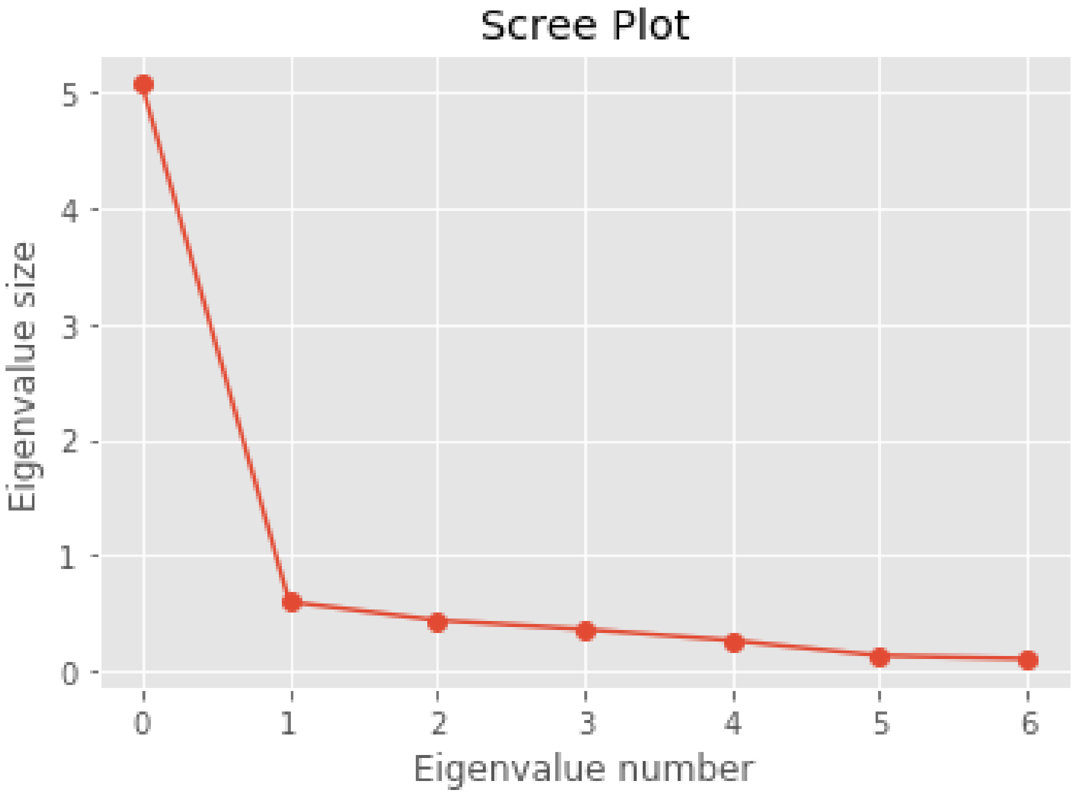
Principal Component Analysis (PCA)

$$\boldsymbol{Su = \lambda u}$$

- To project original data $X \in R^{D*N}$ to P features, where $P \le D$:
  - Calculate the Covariance Matrix $S = \frac{1}{N}(X - \bar{X})(X - \bar{X})^T$
  - Get all the possible eigenvalues and eigenvectors of S.
  - Sort the eigenvalues in descending order:
    - The Largest eigenvalue corresponds to the (eigenvector) axis with highest variance of data projected on that axis.
    - Lower eigenvalues correspond to axes that are worse in preserving the characteristics of the data.
  - Get the highest P eigenvalues and their eigenvectors.
  - Construct full matrix $U \in R^{P*D}$ by stacking all chosen eigenvectors vertically (row-wise)
  - Get the final full projected dataset $Z \in R^{P*N} \rightarrow Z = UX^T$

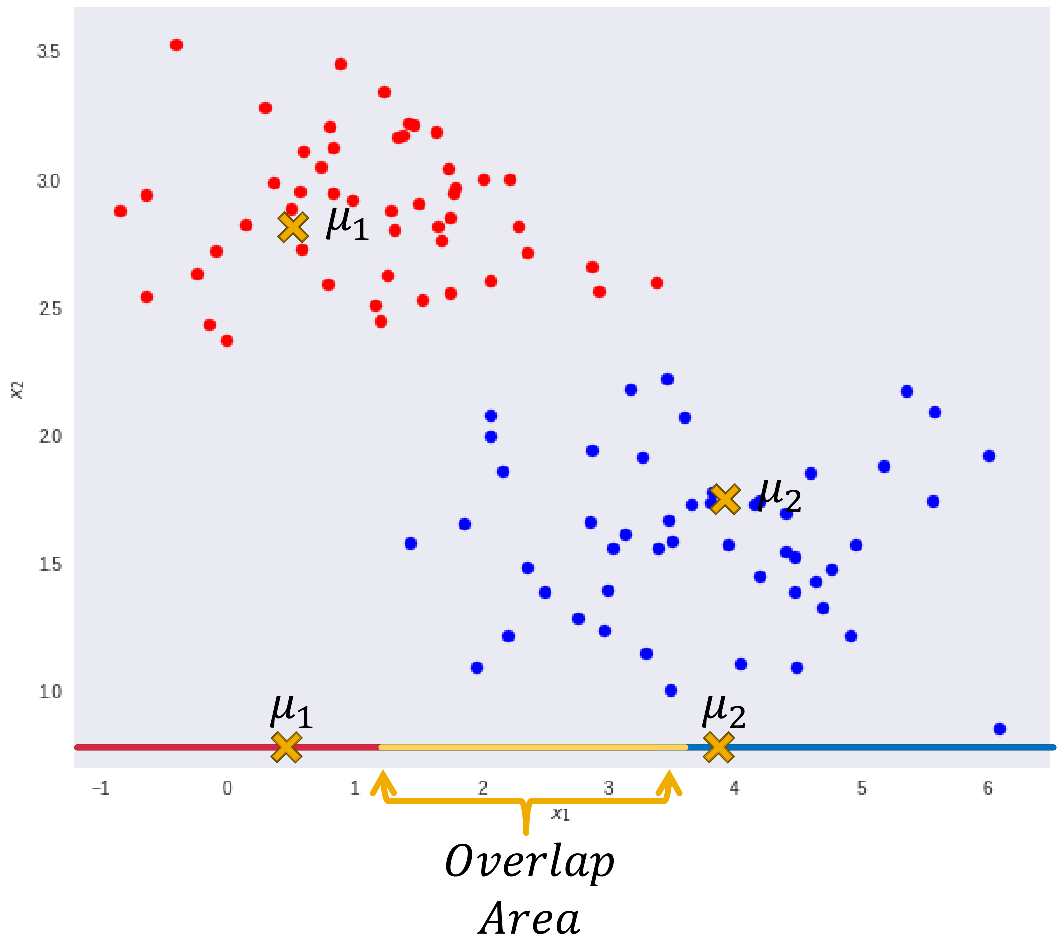
Principal Component Analysis (PCA)





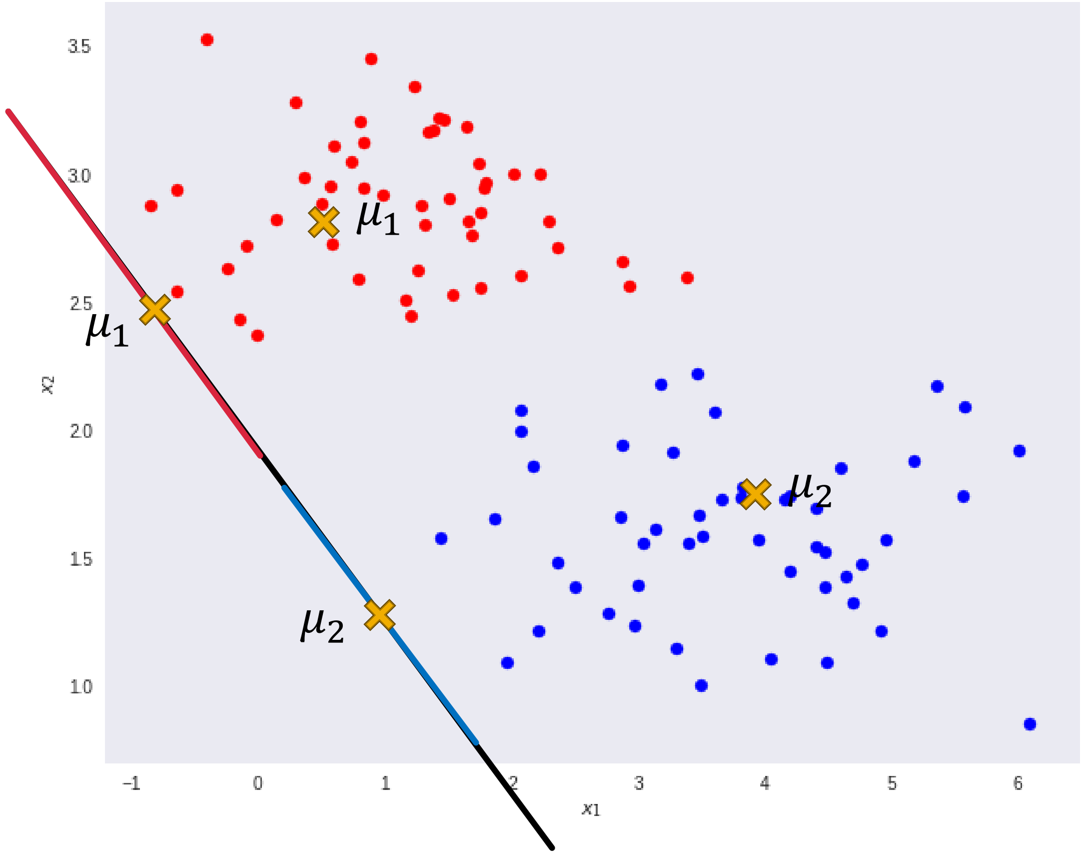
Fisher Discriminant Analysis (FDA)

- Assume we want to project our features to fewer dimensions *<u>while maintaining the separability of our classes.</u>*
- A possible approach would be to ***maximize*** the distance between the centers (means) of the projected classes.
- In the following example, does the proposed axis maintain the best separability between the classes?

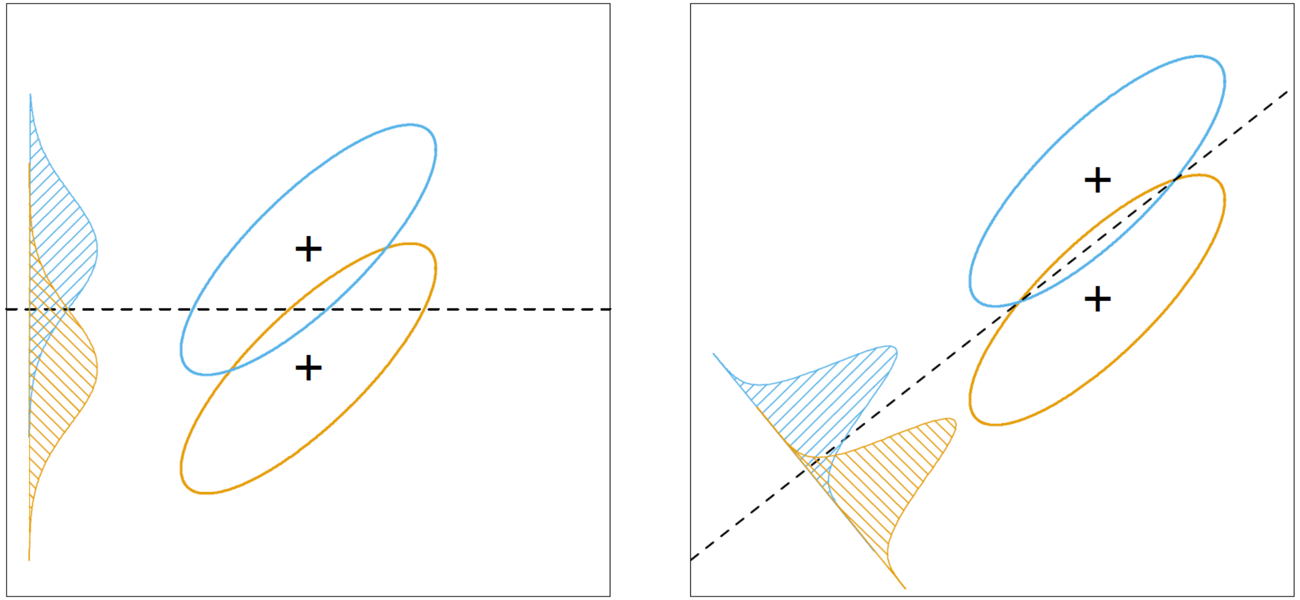


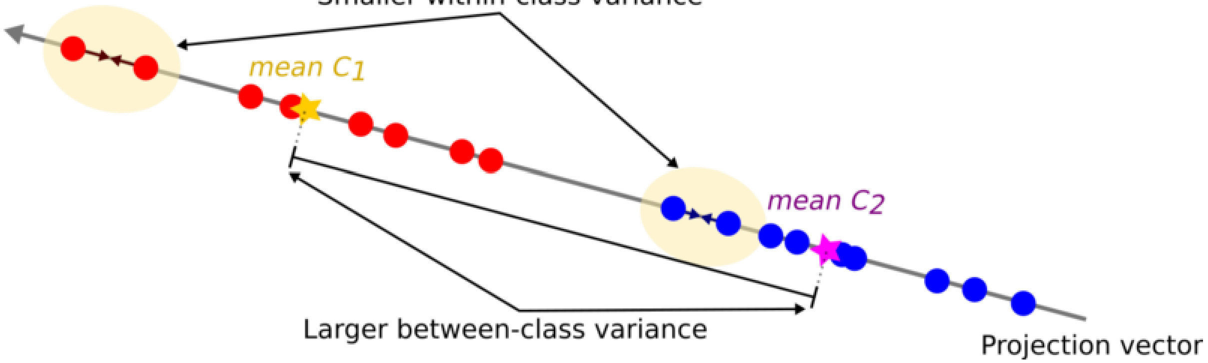
Fisher Discriminant Analysis (FDA)

- It looks like maximizing the distance between means of projected classes is not enough if the classes are wide-spread with high variance.
- An additional constraint could be imposed by *minimizing the within-class variance of the projected data.*
- Combining the two constraints leads to a new axis (dimensionality) that maintains the separability of the original data with minimum or no overlap.



Fisher Discriminant Analysis (FDA)





Fisher Discriminant Analysis (FDA)

- Fisher's Linear Discriminant Analysis (FDA) is a linear dimensionality reduction technique that aims to project high-dimensional data into a lower-dimensional space while maximizing the separation between classes.
- This is achieved through two targets:
  - **T1: Maximizing Inter-Class Variance (Distance between class means)**
  - **T2: Minimizing the Intra-Class Variance (Within-class variance).**

Fisher Discriminant Analysis (FDA)

- Assume that we have two classes to be projected.
- We will apply a linear transformation $u \in R^{D*1}$ on each original data point $x_{\{0,1\}} \in R^{D*1}$ belonging to classes 0 and 1, such that the value z of the projected point at the new axis is formulated as,

$$z = u^T X$$

- The means of the points in each class are formulated as,
  - $\mu_0 = \frac{1}{N_0} \sum_{i=0}^{N_0} x_0^i,$ $\mu_1 = \frac{1}{N_1} \sum_{i=0}^{N_1} x_1^i$

Fisher Discriminant Analysis (FDA)

- $\mu_0 = \frac{1}{N_0}\sum_{i=0}^{N_0} x_0^i,$ $\mu_1 = \frac{1}{N_1}\sum_{i=0}^{N_1} x_1^i$
- **For Target 1:** Distance between the projected means is formulated as:
  - $\left(u^T\mu_0 - u^T\mu_1\right)^2 = \left(u^T\mu_0 - u^T\mu_1\right)^T \left(u^T\mu_0 - u^T\mu_1\right)$
  - $\left(u^T\mu_0 - u^T\mu_1\right)^2 = (\mu_0 - \mu_1)^T u u^T (\mu_0 - \mu_1)$
  - $\left(u^T\mu_0 - u^T\mu_1\right)^2 = u^T(\mu_0 - \mu_1)(\mu_0 - \mu_1)^T u$
  - $\left(u^T\mu_0 - u^T\mu_1\right)^2 = \boldsymbol{u^T S_B u}\,,$ $S_B = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T$
  - Where S_B represents the distance between class means before projection.

Fisher Discriminant Analysis (FDA)

- Recall from PCA → $Cov(Z) = u^T S u, \quad where\ S = Cov(X)$
- **For Target 2:** Within class-variance for the two classes can be formulated as:
  - $Cov(Z_0 + Z_1) = Cov(Z_0) + Cov(Z_1)$
  - $Cov(Z_0 + Z_1) = u^T S_0 u + u^T S_1 u, \quad where\ S_0 = Cov(X_0), S_1 = Cov(X_1)$
  - $Cov(Z_0 + Z_1) = u^T (S_0 + S_1) u = \boldsymbol{u^T S_W u}, \quad S_W = S_0 + S_1$
  - Where S_W represents the within-class variance of the two classes altogether.

Fisher Discriminant Analysis (FDA)

- To Achieve both **Target 1 and Target 2**, our target could be formulated as follows:

$$maximize \ \frac{u^T S_B u}{u^T S_W u}$$

- Recall that our new axis needs to be unit vector (from PCA), to enforce it we can formulate the target as follows:

$$maximize \ \ u^T S_B u, \qquad s.t. \ \ u^T S_W u = 1$$

- To formulate it as a loss function, we will borrow the concepts from Lagrangian Multipliers:

$$L(u,\lambda) = -\left(u^T S_B u - \lambda\left(u^T S_W u - 1\right)\right)$$

  - The negative sign is added to minimize rather than maximize

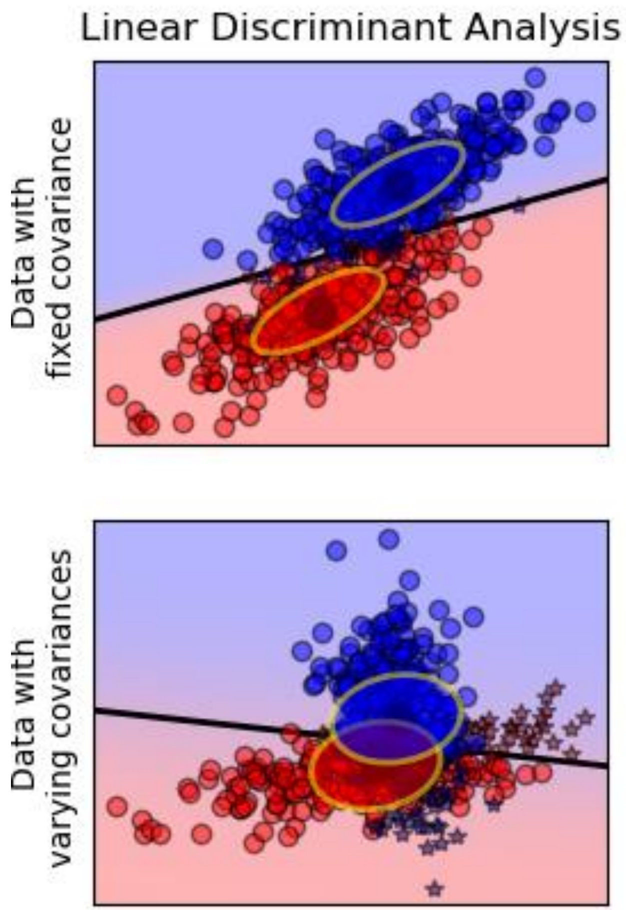
Fisher Discriminant Analysis (FDA)

$$L(u, \lambda) = -\left(u^T S_B u - \lambda\left(u^T S_W u - 1\right)\right)$$

- To minimize the loss with respect to u → Solve $\frac{dL}{du} = 0$
- $\frac{dL}{du} = 2S_B u - 2\lambda S_W u = 0$
- $S_B u = \lambda S_W u \rightarrow \left[\boldsymbol{S_W^{-1} S_B}\right]\boldsymbol{u} = \boldsymbol{\lambda u}$
- We reach a formulation exactly similar to the one of *eigenvalues and eigenvectors*.
- In other words, u is considered an eigenvector of the matrix $\boldsymbol{S_W^{-1} S_B}$calculated from the original data and λ is the associated eigenvalue.
- To transform the full dataset, follow the same steps as PCA, but with calculating $\boldsymbol{S_W^{-1} S_B}$ in the first step instead.

LDA vs FDA

- Both Linear Discriminant Analysis (LDA) and FDA refer to the same technique which aims to project the data to lower dimensions while maximizing the class separability.
- LDA is the direct extension of FDA to work with two **or more** classes.
- LDA is not only doing dimensionality reduction, but also computes the *linear decision boundary* between the classes in the projected space.
- LDA makes important assumptions about the shape of the data:
  - All classes follow a gaussian (normal) distribution
  - All classes have equal (identical) covariance matrices.
- If any of those assumptions doesn't hold, LDA won't perform well in classification or dimensionality reduction.



Extra Resources

- More information regarding the Matrix/Vector Derivatives:
  - https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

Thank you!

- Any questions?

Disclaimer

Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution

Thank You

Youssef Abdelkareem

yabdelkareem@conestogac.on.ca

Credits for Dr. Anasuya Bhima for assisting in the slides structure