https://ttpoll.com/p/664723

COSC 325: Introduction to Machine Learning

Dr. Hector Santos-Villalobos

THE UNIVERSITY OF TENNESSEE
KNOXVILLE

Lecture 17: Feature Extraction

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Class Announcements

Homework

Homework #5 due 11/06
Homework #6 due 11/13

Course Project:

Midterm Report due Tomorrow, Wednesday, 10/30.

Lectures:

11/25 Lecture: No attendance record. Thanksgiving week.

Quizzes:

Weekly quiz as usual.

Exams:

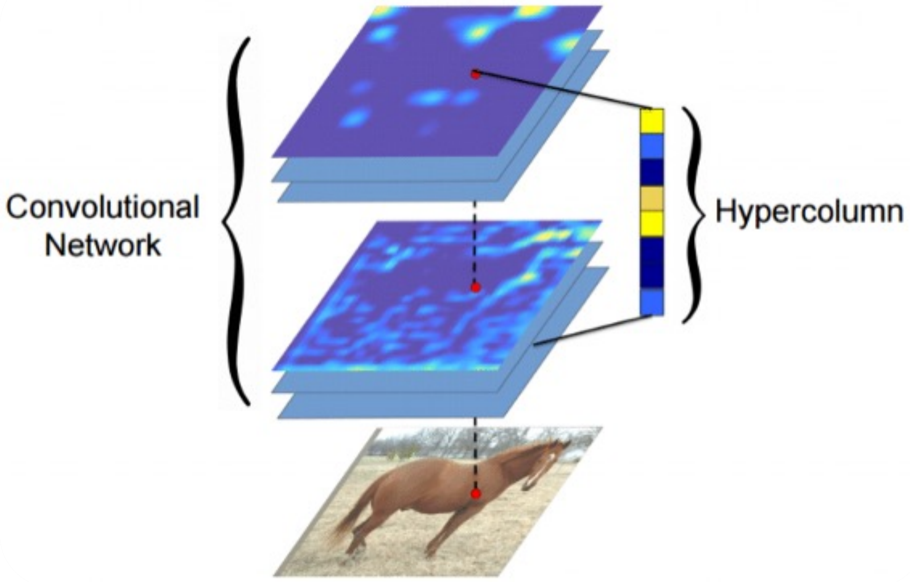
Next exam 11/21. Same format.

Review

- Feature Selection
  - Lasso Regularization
  - Sequential Backward Selection
  - Random Forest Feature Selection
  - Permutation Feature Importance
- Intro to Feature Extraction
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)

Today's Topics

Feature Extraction



Feature Selection

Terminology Check

- Feature selection:

- Feature extraction:

Terminology Check

- **Feature selection:** keeping the best or most important features for machine algorithm training and execution performance. Besides feature scaling, denoising, and encoding, the original feature intent and nature remain.

- **Feature extraction:** transforming the original features into a new feature space that retains the essential information about the original dataset.

Feature Extraction

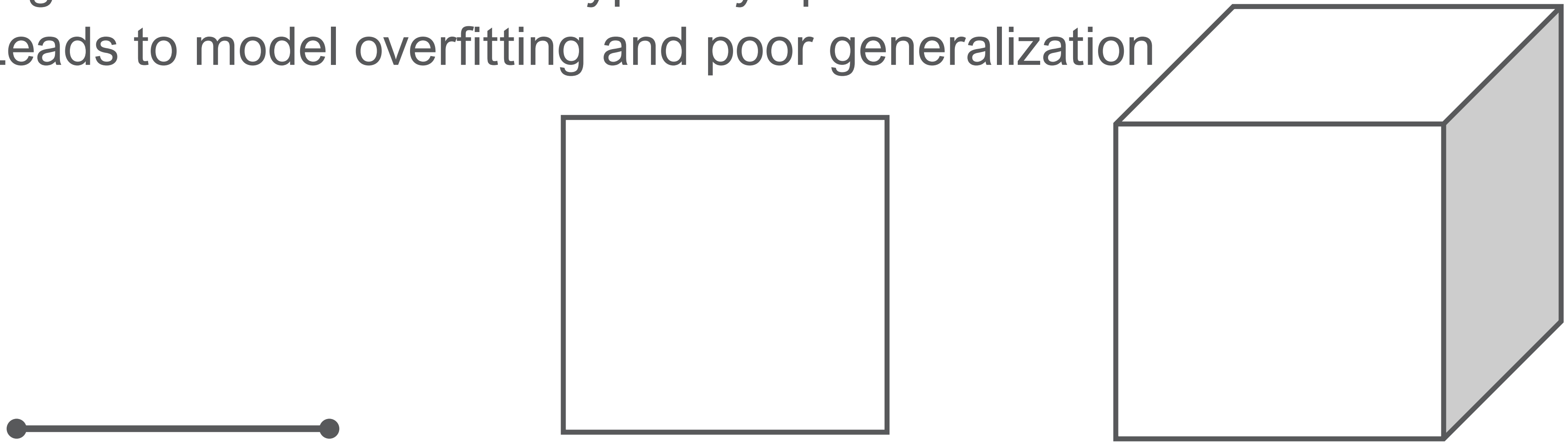
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Dimensionality Reduction

- Transforms original data onto a new feature subspace with lower dimensionality.
  - Other names: data summarization or compression.
- Benefits
  - Reduce storage space requirements
  - Increase computational efficiency of the ML algorithm
  - Improve predictive performance
    - Address the curse of dimensionality

Curse of dimensionality

- Coined by Richard Bellman, (Mathematician)
- As the number of features or dimensions grows, the amount of data (# of samples) we need to generalize grows exponentially.
  - High-dimensional data is typically sparse
  - Leads to model overfitting and poor generalization

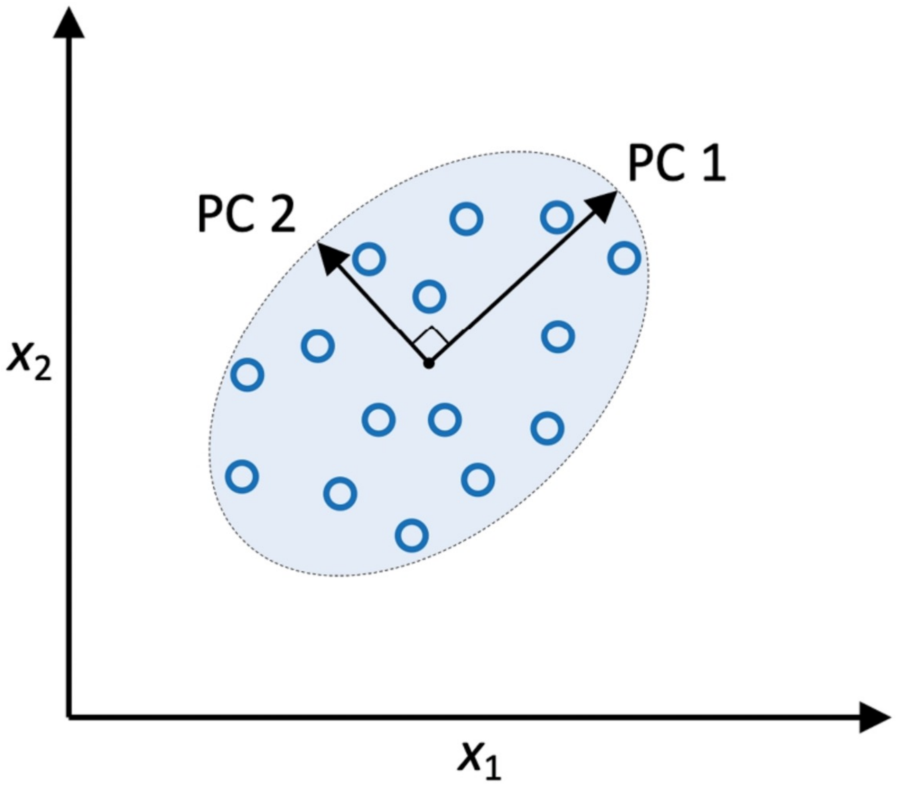


Feature Extraction

- **Principal Component Analysis (PCA)**
- Linear Discriminant Analysis (LDA)
- t-distributed stochastic neighbor embedding (t-SNE)

Principal component analysis (PCA)

- Unsupervised linear transformation technique that is widely used across different fields.
- Popular applications:
  - Dimensionality reduction
  - Feature extraction
  - Exploratory data analysis
  - Denoising of timeseries signals
  - Genome data and gene expression analysis



Source: Raschka, et. al., “Machine Learning with PyTorch and Scikit-Learn”

Objective of PCA

- We have a dataset sample $x^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)}\right], \; x^{(i)} \in \mathbb{R}^m$
- We want to find a projection matrix $W \in \mathbb{R}^{m \times k}$, where

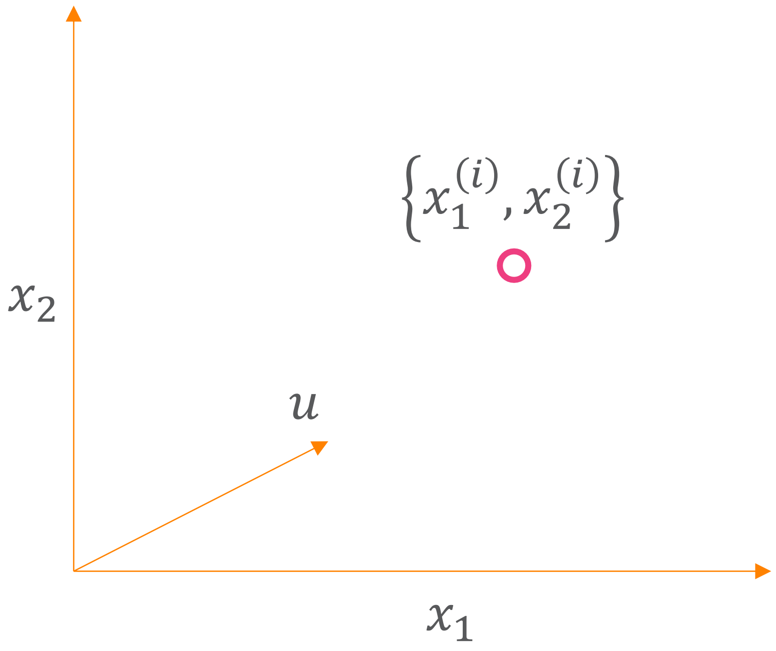
$$x^{(i)} W = z^{(i)}$$

- Resulting in the new vector

$$z^{(i)} = \left[z_1^{(i)}, z_2^{(i)}, \ldots, z_k^{(i)}\right], \qquad z^{(i)} \in \mathbb{R}^k$$

- The output vector is a lower dimensional feature space with $k \ll m$

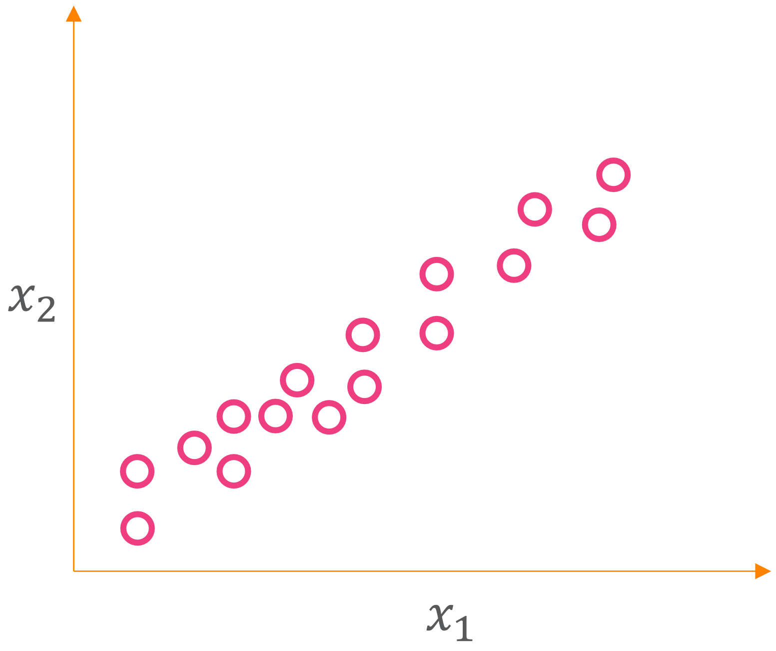
Projections



Unit vector $u \rightarrow u^T u = 1$

Projection $x^{(i)}u$

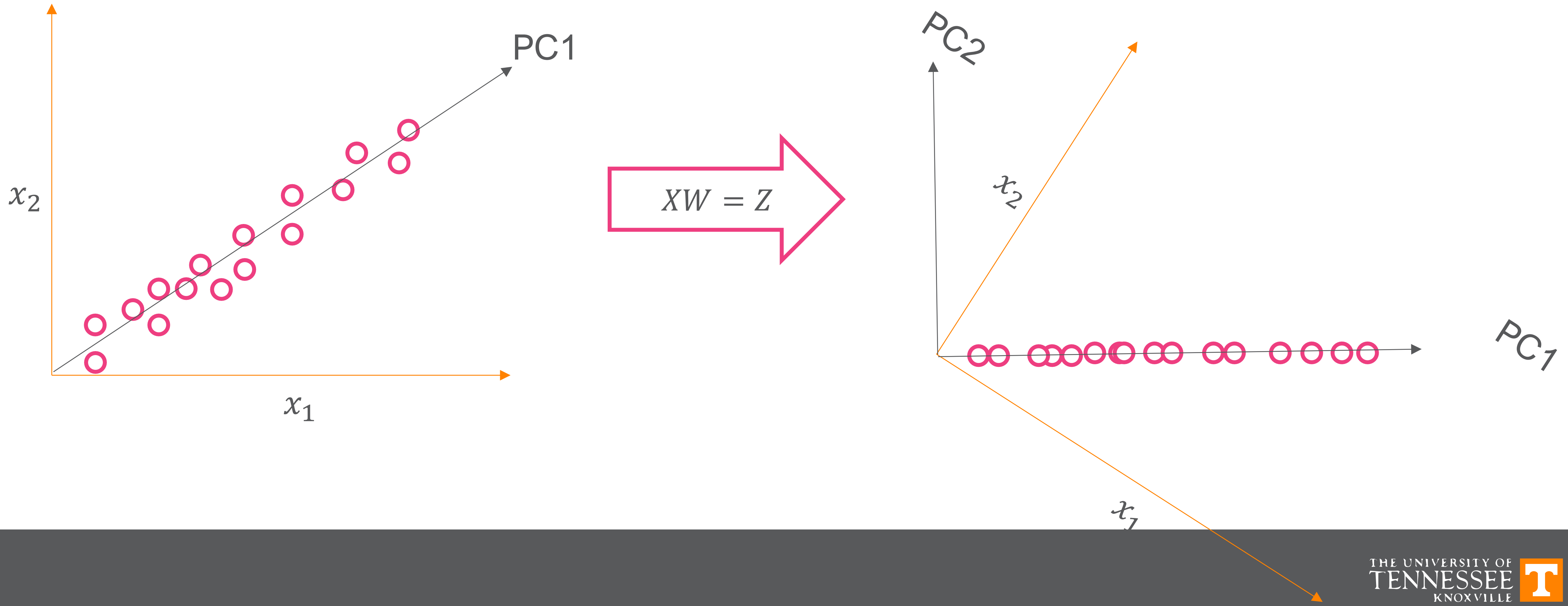
Variance and Principal Components Illustration



https://ttpoll.com/p/664723

Variance and Principal Components Illustration

Variance and Principal Components Illustration



https://www.youtube.com/watch?v=FD4DeN81ODY

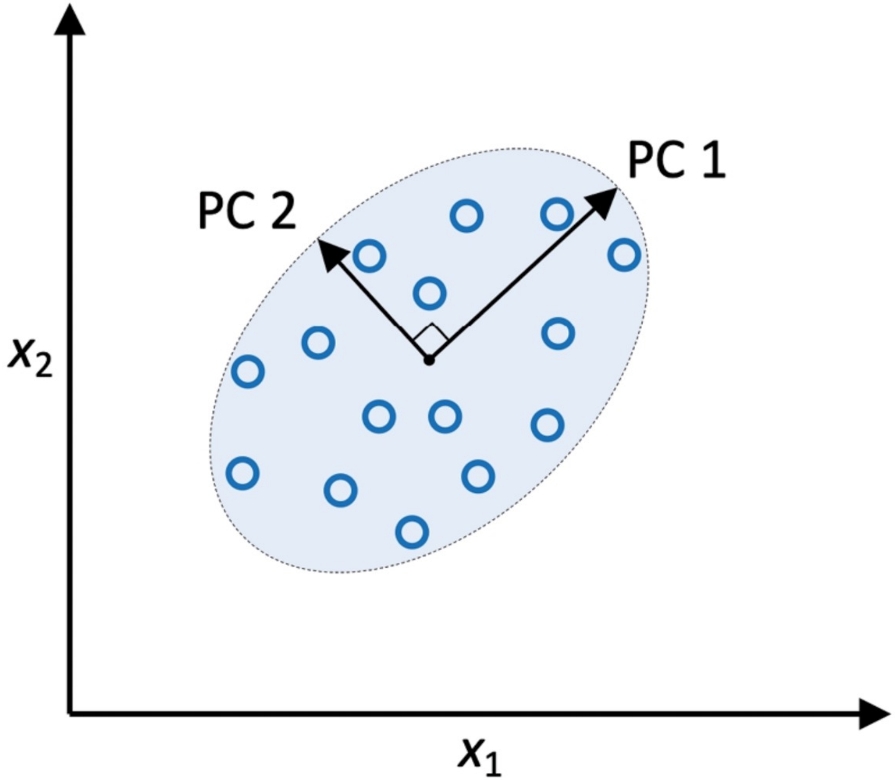
Check out this tutorial on PCA!

PCA Algorithm

1. Standardize original feature data x (Note: Assumes normality)
2. Construct features covariance matrix Σ
3. Decompose Σ in eigenvectors and eigenvalues
4. Compute Explained Variance
    1. Sort eigenvalues in decreasing order
    2. Rank eigenvector importance based on eigenvalues
5. Construct projection matrix W
6. Transform original feature data x with W to generate z

Eigendecomposition Review

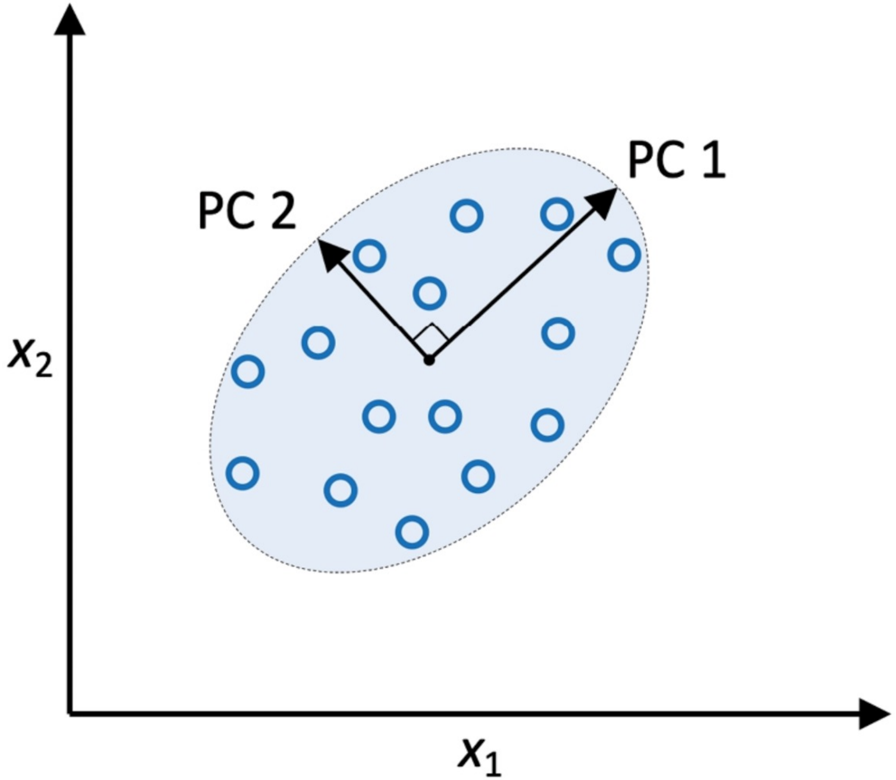
- Eigendecomposition is the factorization of a square matrix into eigenvalues and eigenvectors
  - Eigenvalues and eigenvectors come in pairs.
    - Eigenvalues: variance magnitude
    - Eigenvector: direction of variance (unchanged under linear transforms) Also known as Principal Components (PC)



Source: Raschka, et. al., “Machine Learning with PyTorch and Scikit-Learn”

Eigendecomposition Review

- Eigendecomposition is the factorization of a square matrix into eigenvalues and eigenvectors
  - Eigenvalues and eigenvectors come in pairs.
    - Eigenvalues: variance magnitude
    - Eigenvector: direction of variance (unchanged under linear transforms). Also known as Principal Components (PC)
- The covariance matrix is **square** and **symmetric**, $\Sigma = \Sigma^T$
  - Real eigenvalues
  - Orthogonal eigenvectors
- The eigenvector with the largest eigenvalue points in the direction of the largest variance in the data



Source: Raschka, et. al., "Machine Learning with PyTorch and Scikit-Learn"

Step 2: Covariance Construction

- The covariance between two feature vectors x_j and x_l is given by

$$\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} \left\{ \left(x_j^{(i)} - \mu_j \right) \left(x_l^{(i)} - \mu_l \right) \right\},$$

- μ_j and μ_l are the sample means for features j and l, respectively
  - Note that they are zero after standardization

- Example of a covariance matrix for dataset with three features

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$$

Step 3: Eigendecomposition

- We want to find the eigenvalues λ and eigenvectors v that satisfy the following equation

$$\Sigma v = \lambda v$$

- Computing these values and vectors is beyond this course. For those interested in the theory, please visit the following 3Blue1Brown video tutorial, "Eigenvectors and eigenvalues"
  - Link: https://www.youtube.com/watch?v=PFDu9oVAE-g

Step 3: Eigendecomposition

- We want to find the eigenvalues λ and eigenvectors v that satisfy the following equation

$$\Sigma v = \lambda v$$

- Computing these values and vectors is beyond this course. For those interested in the theory, please visit the following 3Blue1Brown video tutorial, "Eigenvectors and eigenvalues"
  - Link: https://www.youtube.com/watch?v=PFDu9oVAE-g

- We will use the Numpy library to compute eigenvalues and eigenvectors

$$\text{eigen\_vals, eigen\_vectors} = \text{np.linalg.eig(cov\_mat)}$$

PCA Algorithm

1. Standardize original feature data x (Note: Assumes normality)
2. Construct features covariance matrix Σ
3. Decompose Σ in eigenvectors and eigenvalues
4. **Compute Explained Variance**
   1. **Sort eigenvalues in decreasing order**
   2. **Use eigenvalues order to rank eigenvector importance**
5. Construct projection matrix W
6. Transform original feature data x with W to generate z

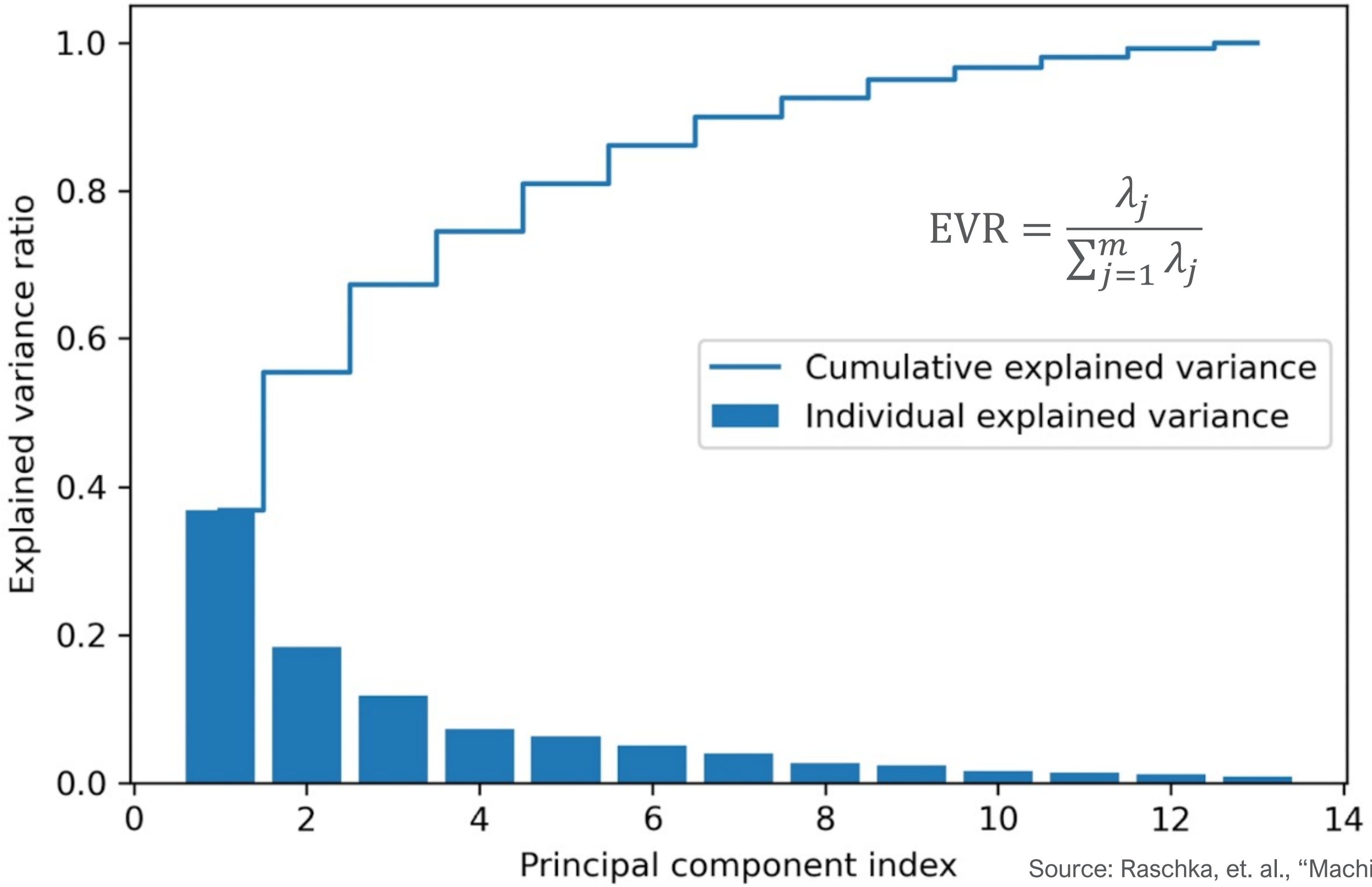
Step 4: Compute Explained Variance

- Select a subset of k eigenvectors that contains most of the information (variance) (i.e., the corresponding k-largest eigenvalues)
- How do we pick k?
  - Hyperparameter tuning
  - **Better way:** By using Total and Explained Variance
    - Compute the Explained Variance Ratio (EVR) $= \frac{\lambda_j}{\sum_{j=1}^{m} \lambda_j}$
    - Compute the cumulative sum of the EVR (Use Numpy cumsum function on EVR)
    - Select the number of eigenvectors that reach the desired cumulative value.

https://ttpoll.com/p/664723

Step 4: Select the k most important eigenvectors: EVR

SKLearn Wine dataset.



$$\text{EVR} = \frac{\lambda_j}{\sum_{j=1}^{m} \lambda_j}$$

Source: Raschka, et. al., “Machine Learning with PyTorch and Scikit-Learn”

Step 4b: Select the k most important eigenvectors: Sort eigenvalues/vector

```python
# Create tuples of eigenvalues and eigenvectors
eigen_data = [(np.abs(eigen_vals[i]), eigen_vecs[:,i])
              for i in range(len(eigen_vals))]

# Sort eigenvalues/vectors tuples from high to low
eigen_data.sort(key=lambda k: k[0], reverse=True)
```

Step 5: Construct projection matrix W

- You have k eigenvectors $v_1, v_2, \dots, v_k$ where $v_j \in \mathbb{R}^m$
- The matrix W should end up with shape $m \times k$
- Therefore, $W = [v_1 \quad v_2 \quad \dots \quad v_k]$
  - You can use np.hstack() function

```
# Construct W with first two eigenvectors
W = np.hstack((eigen_data[0][1][:,np.newaxis],
               eigen_data[1][1][:,np.newaxis]))
```

PCA Algorithm

1. Standardize original feature data x (Note: Assumes normality)
2. Construct features covariance matrix Σ
3. Decompose Σ in eigenvectors and eigenvalues
4. Compute Explained Variance
   1. Sort eigenvalues in decreasing order
   2. Use eigenvalues order to rank eigenvector importance
5. Construct projection matrix W
6. **Transform original feature data x with W to generate z**

Step 6: Transform original feature data x with W to generate z

- Sample transformation:

$$z^{(i)} = x^{(i)}W$$

- Whole dataset transformation:

$$X_{PCA} = XW$$

- Your ML algorithm ingests X_{PCA} instead of X

SkLearn Wine Raw Features

SkLearn Wine Dataset PC1 and PC2

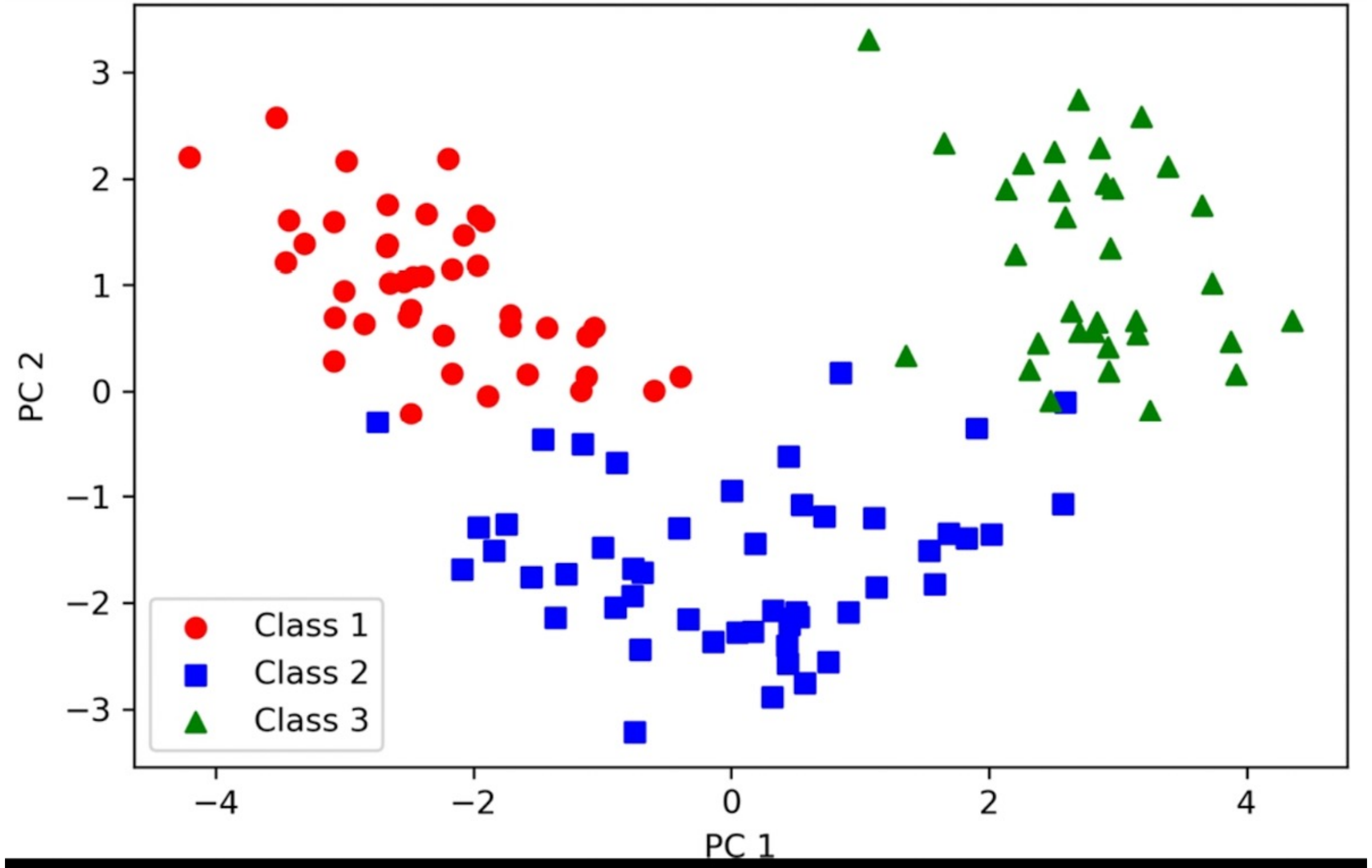


Figure 5.3: Data records from the Wine dataset projected onto a 2D feature space via PCA

```
from sklearn.decomposition import PCA

pca = PCA(n_components=None)
X_train_pca = pca.fit_transform(X_train_std)
pca.explained_variance_ratio_
```

Source: Raschka, et. al., “Machine Learning with PyTorch and Scikit-Learn”

Pop Quiz

If we split a dataset X in X_train, X_val, and X_test sets, what set do we use to compute the PCA projection matrix W?

A. The whole dataset X

B. The training set X_train

C. The validation set X_val

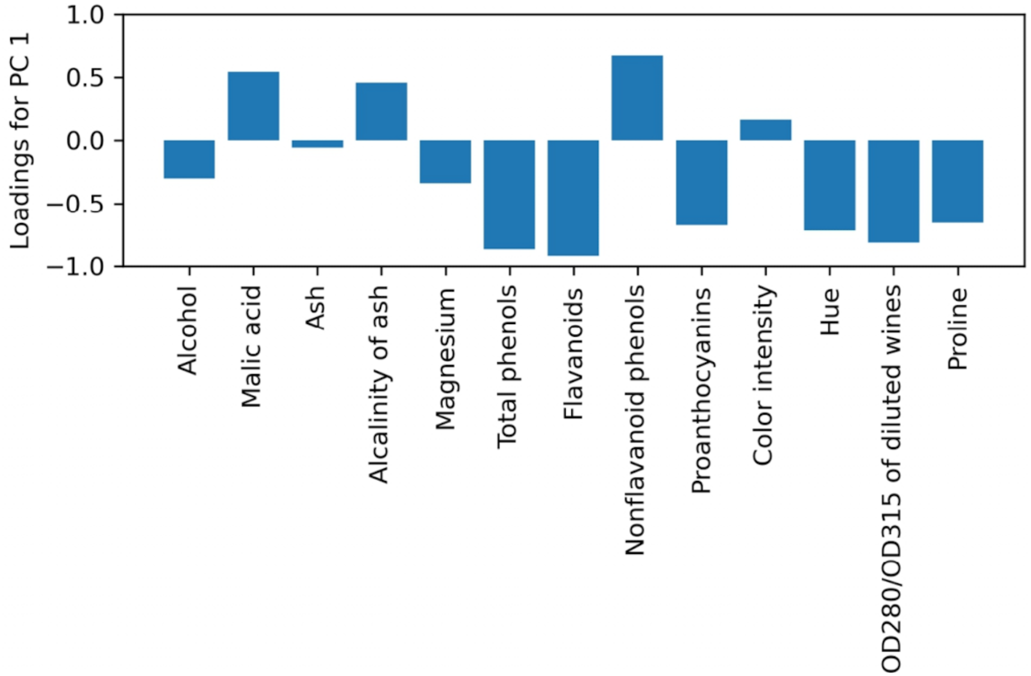
D. The test set X_test

E. The training and validation sets X_train and X_val

Can we learn anything about the raw features from the Principal Components?

Loadings

- Principal components are a combination of original features
- How can we assess the contributions of the original features?
  - Compute loadings
    $loadings = eigenvectors * np.sqrt(eigenvals)$
  - The loadings measures the correlation between the original features and the principal components.



Source: Raschka, et. al., “Machine Learning with PyTorch and Scikit-Learn”

Application Example: DNA2Face

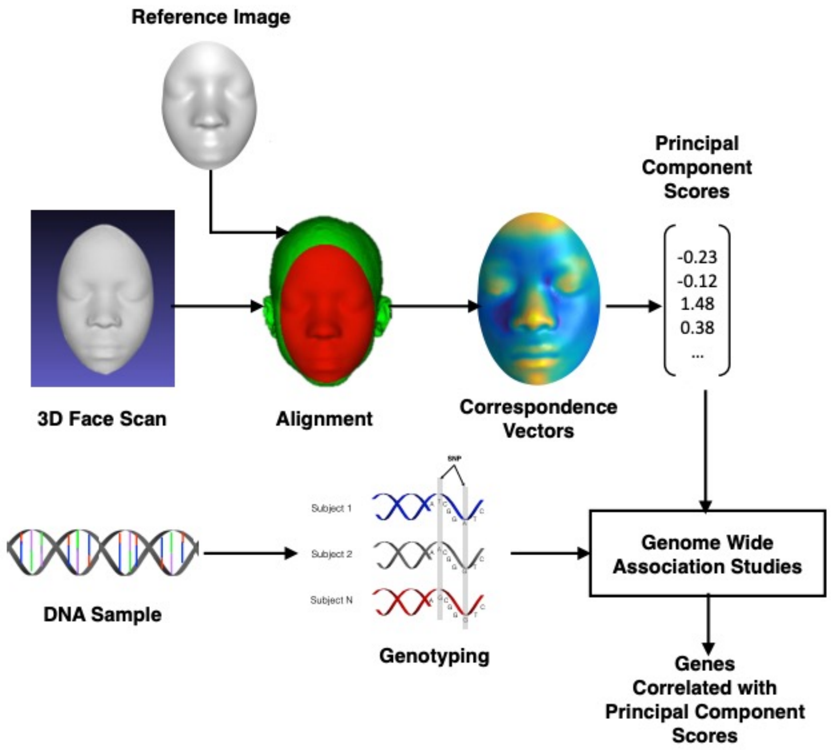


Figure 1: Proposed approach to finding correlations between DNA and facial structure.

Figure 3: Visualization of first 5 PCs from the dataset.

Srinivas, et. al., “DNA2FACE: An approach to correlating 3D facial structure and DNA,” IJCB, 2017

Feature Extraction

- Principal Component Analysis (PCA)
- **Linear Discriminant Analysis (LDA)**
- t-distributed stochastic neighbor embedding (t-SNE)

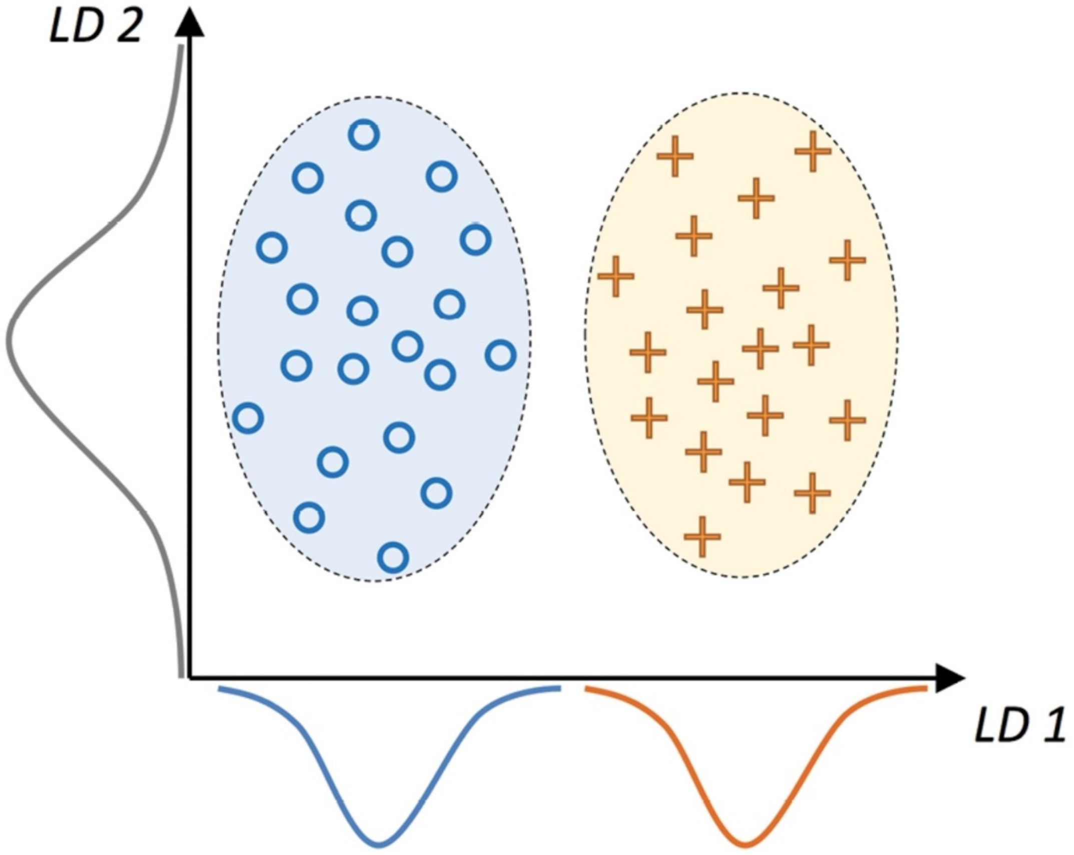
Linear Discriminant Analysis (LDA)

- Also known as Fisher's LDA (Ronald A. Fisher 1936)
- As PCA, the high-level goal is to reduce dimensionality
- Supervised technique (i.e., requires labels)
- Increase computational efficiency
- Reduce overfitting (Curse of dimensionality)
- **PCA Objective:** find the orthogonal component axes of maximum variance in a dataset
- **LDA Objective:** find the feature subspace that optimizes class separability.

Two-Class LDA Problem

- LD2: Good principal component
- LD1: Good LDA component

- Assumptions:
  - Data is normally distributed.
  - Classes have identical covariance matrices
  - Training examples are statistically independent of each other
- LDA typically works well even when one or more assumptions are violated.



Source: Raschka, et. al., "Machine Learning with PyTorch and Scikit-Learn"

LDA Algorithm

1. Standardize original feature data x (Note: Assumes normality) with m features.
2. For each class, compute the m-dimensional mean vector $\bar{v}$
3. Construct the between-class scatter matrix S_B and within class scatter matrix S_W
4. Compute the eigenvectors and corresponding eigenvalues of the matrix $S_W^{-1} S_B$
5. Sort eigenvalues in decreasing order. This also sorts eigenvectors.
6. Construct projection matrix W with the k largest eigenvalues (columns). W has shape $m \times k$.
7. Transform original feature data x with W to generate z

Step 2: m-dimensional mean vector

- For a dataset with C classes and m features, we will generate C m-dimensional mean vectors, each with m elements.
- General m-dimensional vector $\bar{v}_i$
- $\bar{v}_i = \begin{bmatrix} \mu_{i,x_1} \\ \mu_{i,x_2} \\ \vdots \\ \mu_{i,x_m} \end{bmatrix}$, where μ_{i,x_j} is the average feature j value for class i.

$$i \in \{1, 2, \ldots, C\}$$

m-dimensional Mean Vector Example

Sample ID	x_1	x_2	Label
1	1	3	0
2	2	3	0
3	3	2	0
4	4	3	1
5	4	2	1
6	5	3	1
7	2	5	2
8	3	4	2
9	4	5	2

$$\bar{v}_0 = \begin{bmatrix} \frac{(1+2+3)}{3} \\ \frac{(3+3+2)}{3} \end{bmatrix} = \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}$$

m-dimensional Mean Vector Example

Sample ID	x_1	x_2	Label
1	1	3	0
2	2	3	0
3	3	2	0
4	4	3	1
5	4	2	1
6	5	3	1
7	2	5	2
8	3	4	2
9	4	5	2

$$\bar{v}_0 = \begin{bmatrix} \frac{(1+2+3)}{3} \\ \frac{(3+3+2)}{3} \end{bmatrix} = \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}$$

$$\bar{v}_1 = \begin{bmatrix} \frac{(4+4+5)}{3} \\ \frac{(3+2+3)}{3} \end{bmatrix} = \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix}$$

$$\bar{v}_2 = \begin{bmatrix} \frac{(2+3+4)}{3} \\ \frac{(5+4+5)}{3} \end{bmatrix} = \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}$$

Step 3: Construct Scatter Matrices

- Compute within scatter matrix S_w of size $m \times m$

$$S_W = \sum_{i=1}^{C} S_i, \text{ where}$$

$$S_i = \sum_{x \in D_i} (x - \bar{v}_i)(x - \bar{v}_i)^T$$

- This is tracking the within class variance.

Step 3: Construct Scatter Matrices

Sample ID	x_1	x_2	Label
1	1	3	0
2	2	3	0
3	3	2	0
4	4	3	1
5	4	2	1
6	5	3	1
7	2	5	2
8	3	4	2
9	4	5	2

- Compute within scatter matrix S_w of size $m \times m$

$$S_W = \sum_{i=1}^{C} S_i, \text{ where}$$

$$S_i = \sum_{x \in D_i} (x - \bar{v}_i)(x - \bar{v}_i)^T$$

- Continuing with our example

$$\boxed{\bar{v}_0 = \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}} \quad \bar{v}_1 = \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \quad \bar{v}_2 = \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}$$

$$S_0 = \sum_{x \in D_0} (x - \bar{v}_0)(x - \bar{v}_0)^T$$

$$= \left(\begin{bmatrix} 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}\right)\left(\begin{bmatrix} 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}\right)^{\mathrm{T}} + \left(\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}\right)\left(\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}\right)^{\mathrm{T}}$$

$$+ \left(\begin{bmatrix} 3 \\ 2 \end{bmatrix} - \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}\right)\left(\begin{bmatrix} 3 \\ 2 \end{bmatrix} - \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix}\right)^{\mathrm{T}} = \begin{bmatrix} 2 & -1 \\ -1 & 0.6667 \end{bmatrix}$$

Step 3: Construct Scatter Matrices

Sample ID	x_1	x_2	Label
1	1	3	0
2	2	3	0
3	3	2	0
4	4	3	1
5	4	2	1
6	5	3	1
7	2	5	2
8	3	4	2
9	4	5	2

- Compute within scatter matrix S_w of size $m \times m$

$$S_W = \sum_{i=1}^{C} S_i, \text{ where}$$

$$S_i = \sum_{x \in D_i} (x - \bar{v}_i)(x - \bar{v}_i)^T$$

- Continuing with our example

$$\bar{v}_0 = \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix} \quad \boxed{\bar{v}_1 = \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix}} \quad \bar{v}_2 = \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}$$

$$S_1 = \sum_{x \in D_1} (x - \bar{v}_1)(x - \bar{v}_1)^T$$

$$= \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix} - \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \right) \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix} - \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \right)^T + \left(\begin{bmatrix} 4 \\ 2 \end{bmatrix} - \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \right) \left(\begin{bmatrix} 4 \\ 2 \end{bmatrix} - \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \right)^T$$

$$+ \left(\begin{bmatrix} 5 \\ 3 \end{bmatrix} - \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \right) \left(\begin{bmatrix} 5 \\ 3 \end{bmatrix} - \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \right)^T = \begin{bmatrix} 0.6667 & 0.3333 \\ 0.3333 & 0.6667 \end{bmatrix}$$

Step 3: Construct Scatter Matrices

Sample ID	x_1	x_2	Label
1	1	3	0
2	2	3	0
3	3	2	0
4	4	3	1
5	4	2	1
6	5	3	1
7	2	5	2
8	3	4	2
9	4	5	2

- Compute within scatter matrix S_w of size $m \times m$

$$S_W = \sum_{i=1}^{C} S_i, \text{ where}$$

$$S_i = \sum_{x \in D_i} (x - \bar{v}_i)(x - \bar{v}_i)^T$$

- Continuing with our example

$$\bar{v}_0 = \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix} \quad \bar{v}_1 = \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \quad \boxed{\bar{v}_2 = \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}}$$

$$S_2 = \sum_{x \in D_2} (x - \bar{v}_2)(x - \bar{v}_2)^T$$

$$= \left(\begin{bmatrix} 2 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}\right)\left(\begin{bmatrix} 2 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}\right)^{\mathrm{T}} + \left(\begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}\right)\left(\begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}\right)^{\mathrm{T}}$$

$$+ \left(\begin{bmatrix} 4 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}\right)\left(\begin{bmatrix} 4 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}\right)^{\mathrm{T}} = \begin{bmatrix} 2 & 0 \\ 0 & 0.6667 \end{bmatrix}$$

Step 3: Construct Scatter Matrices

- Compute within scatter matrix S_w of size $m \times m$

$$S_W = \sum_{i=1}^{C} S_i, \text{ where}$$

$$S_i = \sum_{x \in D_i} (x - \bar{v}_i)(x - \bar{v}_i)^T$$

- Continuing with our example

$$S_0 = \begin{bmatrix} 2 & -1 \\ -1 & 0.6667 \end{bmatrix}, \quad S_1 = \begin{bmatrix} 0.6667 & 0.3333 \\ 0.3333 & 0.6667 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 2 & 0 \\ 0 & 0.6667 \end{bmatrix}$$

$$S_W = S_0 + S_1 + S_2 = \begin{bmatrix} 4.667 & -0.667 \\ -0.667 & 2.0 \end{bmatrix}$$

Step 3: Construct Scatter Matrices

- Compute within scatter matrix S_w of size $m \times m$

$$S_W = \sum_{i=1}^{C} S_i, \text{ where}$$

$$S_i = \sum_{x \in D_i} (x - \bar{v}_i)(x - \bar{v}_i)^T$$

- Continuing with our example

$$S_0 = \begin{bmatrix} 2 & -1 \\ -1 & 0.6667 \end{bmatrix}, \quad S_1 = \begin{bmatrix} 0.6667 & 0.3333 \\ 0.3333 & 0.6667 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 2 & 0 \\ 0 & 0.6667 \end{bmatrix}$$

$$S_W = S_0 + S_1 + S_2 = \begin{bmatrix} 4.667 & -0.667 \\ -0.667 & 2.0 \end{bmatrix}$$

When classes are unbalanced, normalize each scatter matrix.

$$S_W = \sum_{i=1}^{C} \frac{S_i}{\mathrm{n_i}}$$

Step 3: Construct Scatter Matrices

- Compute ***between*** scatter matrix S_B of size $m \times m$

$$S_B = \sum_{i=1}^{C} n_i (\bar{v}_i - \bar{v})(\bar{v}_i - \bar{v})^T,$$

where $\bar{v}$ is the overall mean including examples from all C classes.

$$\bar{v}_0 = \begin{bmatrix} 2 \\ \frac{8}{3} \end{bmatrix} \quad \bar{v}_1 = \begin{bmatrix} \frac{13}{3} \\ \frac{8}{3} \end{bmatrix} \quad \bar{v}_2 = \begin{bmatrix} 3 \\ \frac{14}{3} \end{bmatrix}$$

Sample ID	x_1	x_2	Label
1	1	3	0
2	2	3	0
3	3	2	0
4	4	3	1
5	4	2	1
6	5	3	1
7	2	5	2
8	3	4	2
9	4	5	2

$$\bar{v} = \begin{bmatrix} 3.11 \\ 10 \end{bmatrix}$$

LDA Algorithm

1. Standardize original feature data x (Note: Assumes normality) with m features.
2. For each class, compute the m-dimensional mean vector $\bar{v}$
3. Construct the between-class scatter matrix S_B and within class scatter matrix S_W
4. Compute the eigenvectors and corresponding eigenvalues of the matrix $\boldsymbol{S_W^{-1} S_B}$
5. Sort eigenvalues in decreasing order. This also sorts eigenvectors.
6. Construct projection matrix W with the k largest eigenvalues (columns). W has shape $m \times k$.
7. Transform original feature data x with W to generate z

Similar to PCA

Step 4: Compute the eigenvectors and corresponding eigenvalues of the matrix $S_W^{-1} S_B$

```
eigen_vals, eigen_vecs =\
    np.linalg.eig(np.linalg.inv(S_W).dot(S_B))
```

Step 5: Sort eigenvalues and pick eigenvectors corresponding to k-largest eigenvalues.

Step 6: Construct projection matrix W

- Construct W as we did for PCA
- Expect at most $C - 1$ linear discriminants
- Discriminability Ratio is equivalent to PCA Explained Variance Ratio

SKLearn Wine dataset.

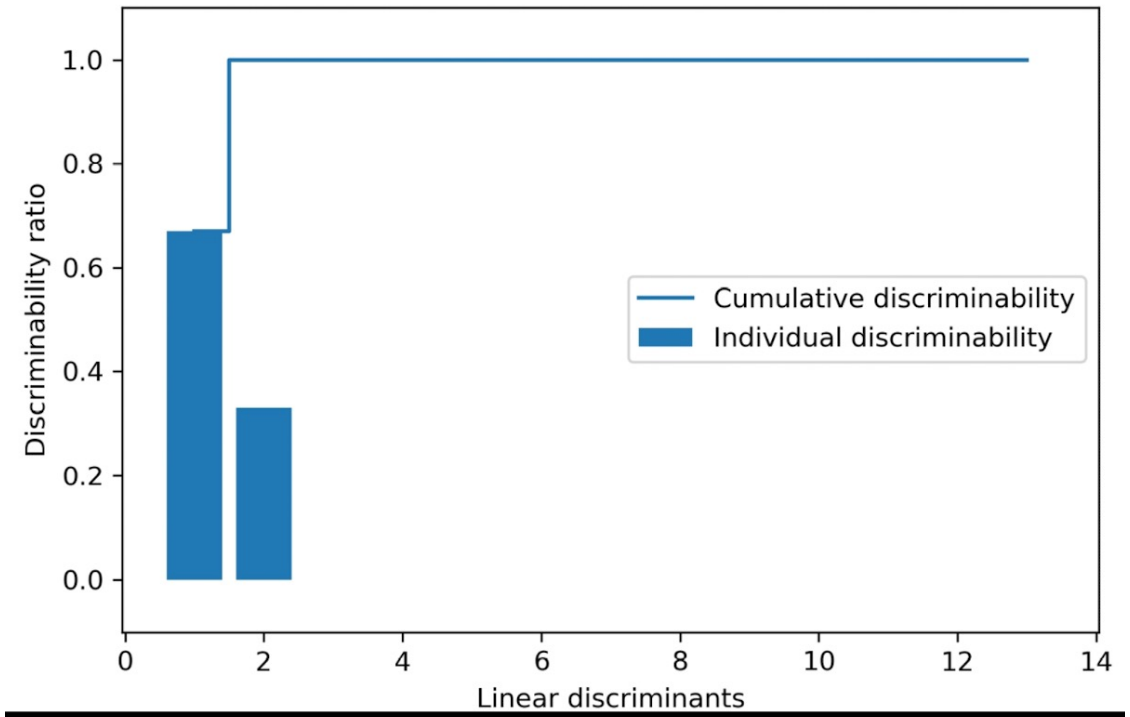


Figure 5.9: The top two discriminants capture 100 percent of the useful information

https://ttpoll.com/p/664723

Step 7: Transform original feature data x with W to generate z

- Same projection step as with PCA, $Z = XW$

PCA

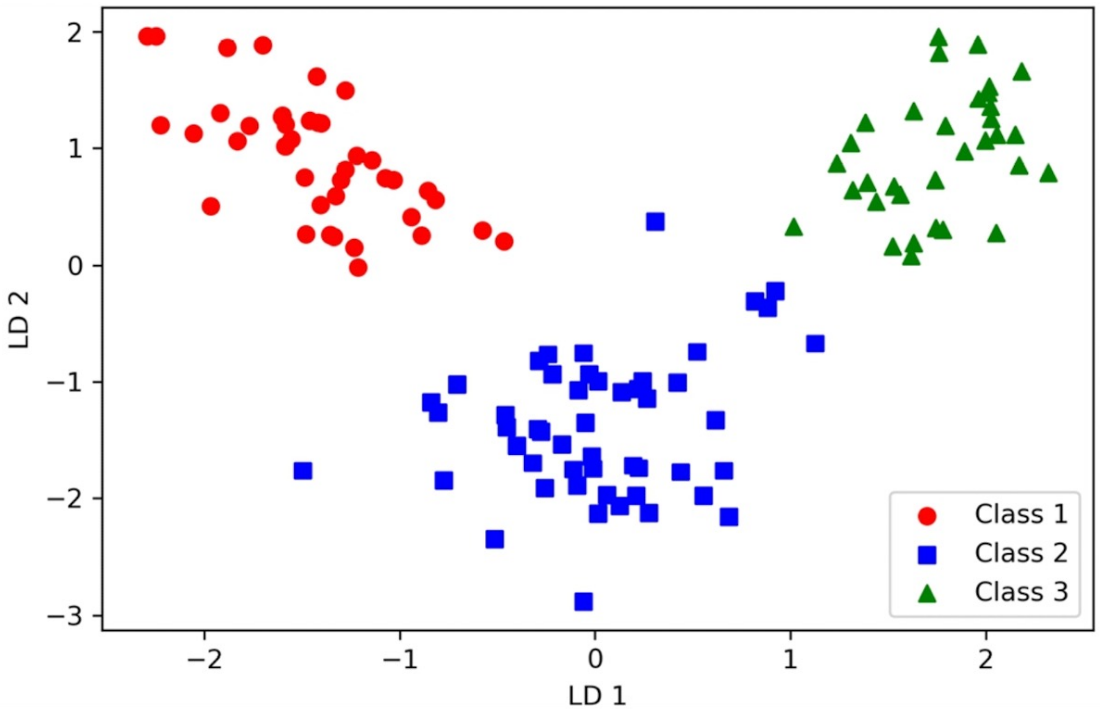


Figure 5.10: Wine classes perfectly separable after projecting the data onto the first two discriminants

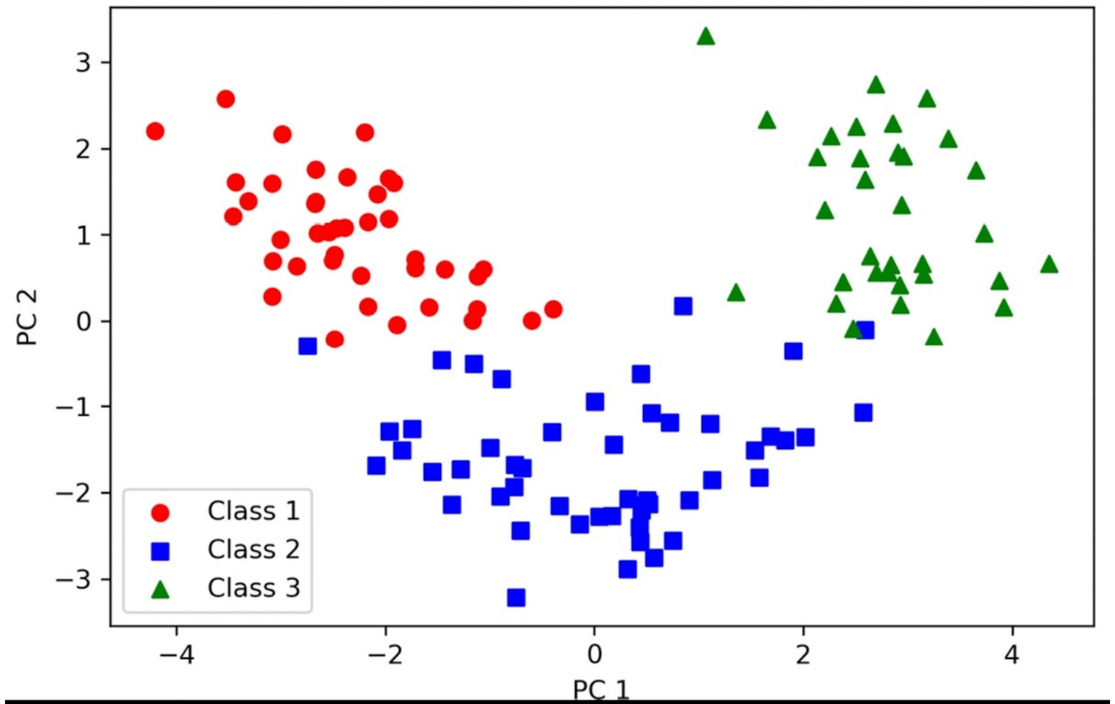


Figure 5.3: Data records from the Wine dataset projected onto a 2D feature space via PCA

https://ttpoll.com/p/664723

Step 7: Transform original feature data x with W to generate z

- Same projection step as with PCA, $Z = XW$

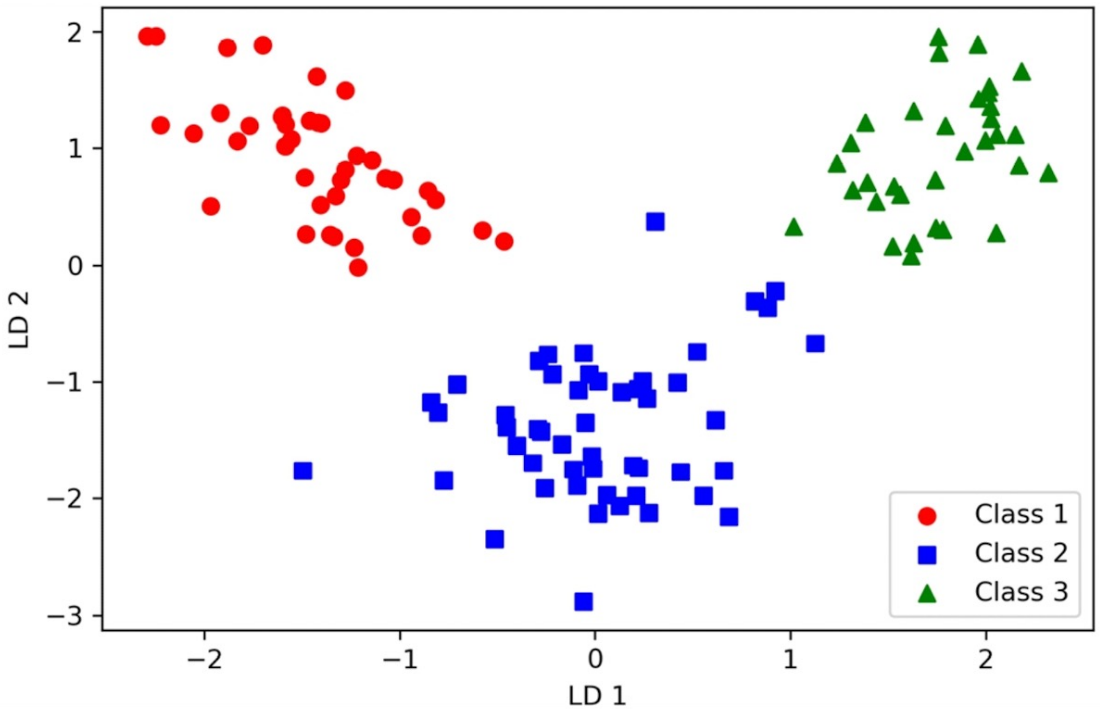


Figure 5.10: Wine classes perfectly separable after projecting the data onto the first two discriminants

```python
# LDA Library
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

# Constructor
lda = LDA(n_components=2)

# Fit and Transform
X_train_lda = lda.fit_transform(X_train_std, y_train)
```


THE UNIVERSITY OF TENNESSEE KNOXVILLE

Pop Quiz

X finds eigenvectors that best separate the classes, while Y finds the eigenvectors in the direction of the highest variance in the data independent of the class.

A. X is PCA and Y is LDA

B. X is LDA and Y is PCA

C. None of the above

Feature Extraction

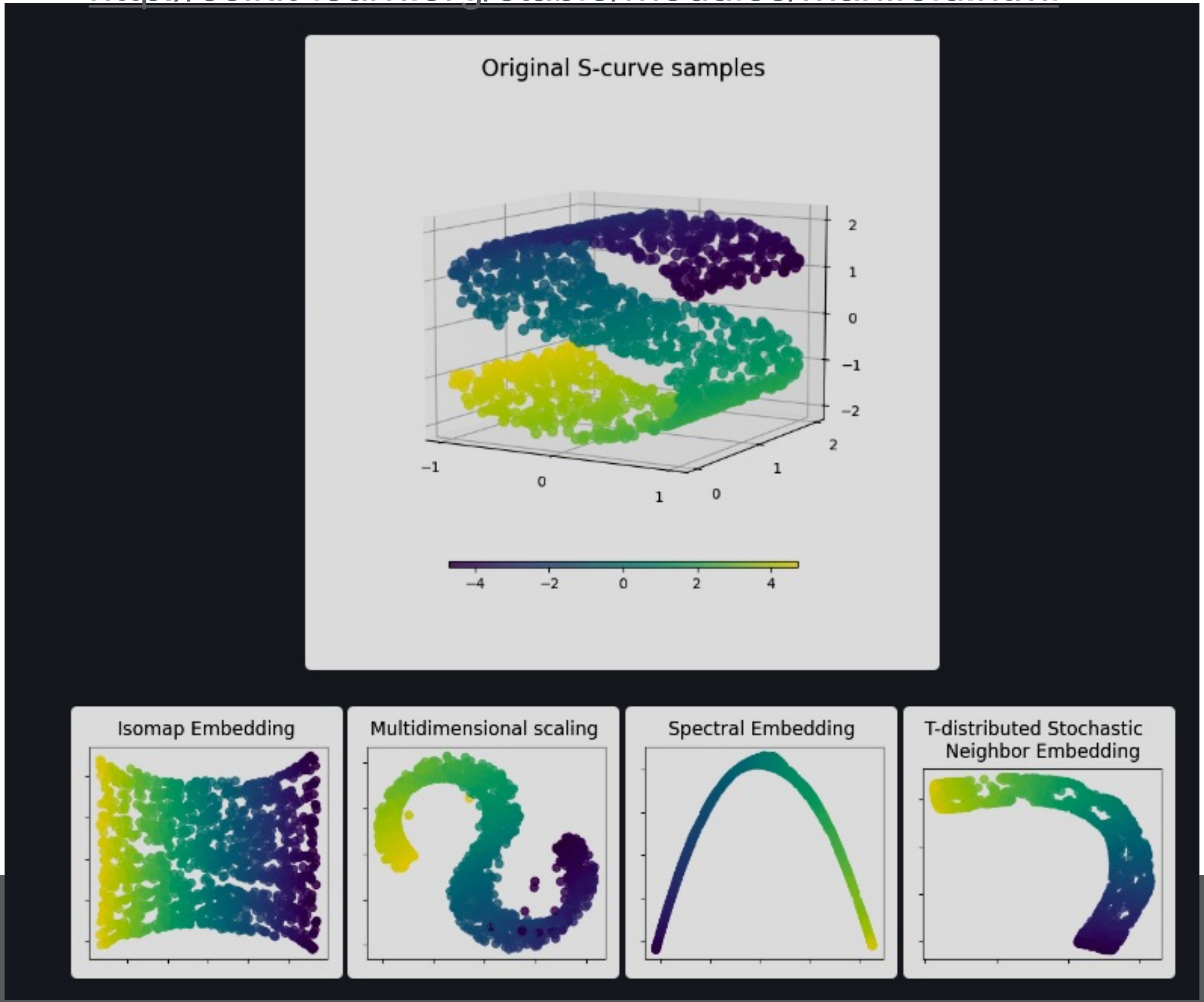
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- **t-distributed stochastic neighbor embedding (t-SNE)**

t-distributed stochastic neighbor embedding (t-SNE)

- Popular for data visualization
- Projects m-dimensional space to 2D/3D
- It is a non-linear dimensionality reduction technique also known as manifold learning
  - Manifold: lower-dimensional topological space embedded in a high-dimensional space
- Other sklearn non-linear techniques for dimensionality reduction can be found at http://scikit-learn.org/stable/modules/manifold.html

Manifold Learning

http://scikit-learn.org/stable/modules/manifold.html



t-SNE in SkLearn

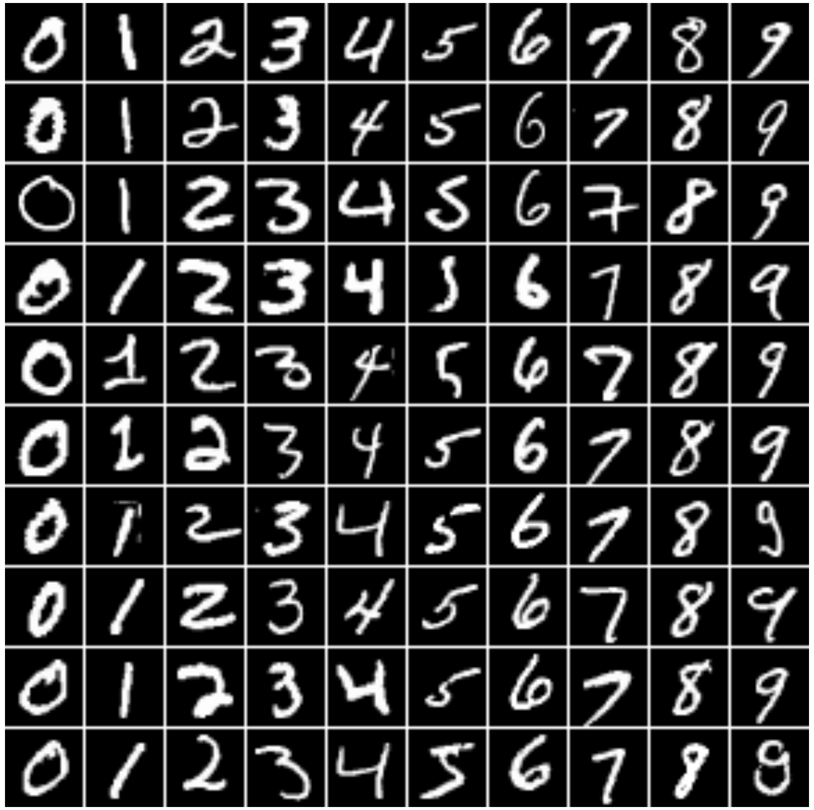
```python
from sklearn.manifold import TSNE
```

```python
# Initialize t-SNE with 2 components for 2D visualization
tsne = TSNE(n_components=2, random_state=42)

# Apply t-SNE to the data
X_tsne = tsne.fit_transform(X)
```

T-SNE

MNIST Digits Dataset



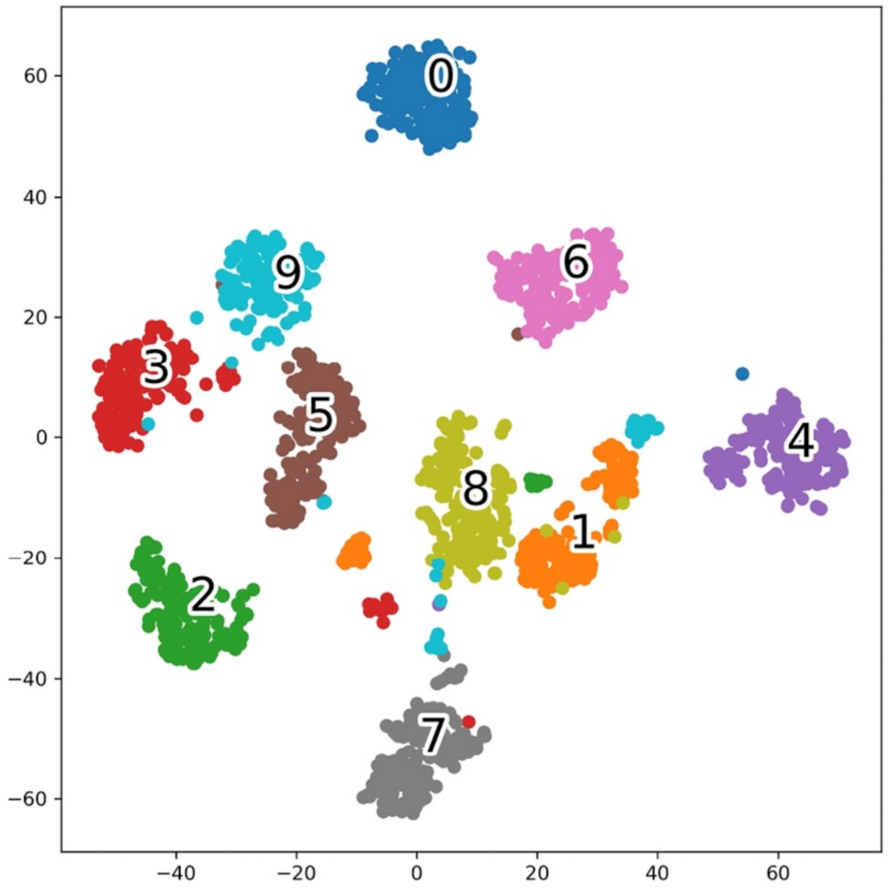
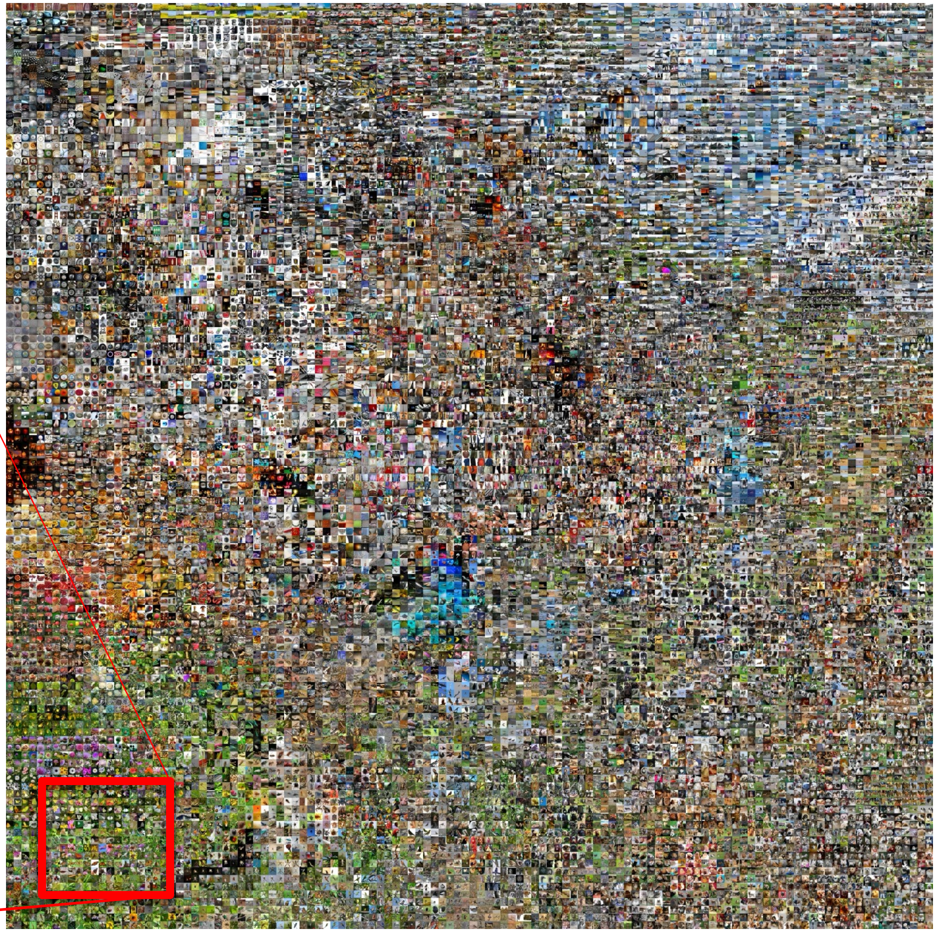


Figure 5.16: A visualization of how t-SNE embeds the handwritten digits in a 2D feature space

Dimensionality Reduction



http://cs.stanford.edu/people/karpathy/cnnembed/

Feature Extraction Review

- Principal Component Analysis (PCA)
  - For datasets with categorical features
    - Don't apply PCA to those features
    - Multiple Correspondence Analysis (MCA)
    - Factor Analysis of Mixed Data (FAMD)
- Linear Discriminant Analysis (LDA)
- t-distributed stochastic neighbor embedding (t-SNE)

Review

- Feature Extraction
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - t-distributed stochastic neighbor embedding (t-SNE)

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Next Lecture

- Shapley Values for XAI
- Unsupervised learning

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Helper Slides