Linear Discriminant Analysis

MATH 156 Final Project

Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

University of California, Los Angeles

Overview

- LDA can be used a multiclass classification model or a supervised dimensionality reduction routine
- LDA is a dimensionality reduction technique similar to PCA. Unlike PCA, it is supervised and it focuses on class separability.
- Assumes normally distributed data from each class and equal covariance matrices for each class.
- Requires large sample size
- Easy data visualization
- Linear decision boundaries

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

Mathematical Formulation

- **Assumptions**: Data within each class is sampled from distribution $\mathcal{N}(\mathbf{m}_k, \Sigma)$ where covariance Σ is same for all classes
- **Goal**: Project data down to a lower dimension in such a way that it maximizes separation between classes and minimizes separation within classes (between-class variance vs. within-class variance).
- **Projected within class variance** (for class k):

$$\begin{aligned}
&\frac{1}{|C_k|}\sum_{i\in C_k}(\mathbf{v}^\top \mathbf{x}_i - \mathbf{v}^\top \mathbf{m}_k)^2 \\
&\quad = \frac{1}{|C_k|}\sum_{i\in C_k}[\mathbf{v}^\top(\mathbf{x}_i - \mathbf{m}_k)][(\mathbf{x}_i - \mathbf{m}_k)^\top \mathbf{v}] \\
&\quad = \mathbf{v}^\top \Sigma \mathbf{v}
\end{aligned}$$

- **Between class covariance**:

$$\mathbf{B} = \frac{1}{C}\sum_{k=1}^{C}(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^\top$$

- **Objective**: Find a direction vector $\mathbf{v}$ such that the projected variance of class means is maximized and the projected variance within each class is minimized. We will assume we are projecting to $\mathbb{R}^1$.

$$\underset{\mathbf{v}}{\text{argmax}}\ \frac{\mathbf{v}^\top \mathbf{B}\mathbf{v}}{\mathbf{v}^\top \Sigma \mathbf{v}}$$

- **Solution**: This objective function is maximized by the eigenvectors of the M largest eigenvalues of $\Sigma^{-1}\mathbf{B}$ (more details on the next slide). Due to homogeneity of covariances, class prediction is done by minimizing Euclidean distance between projection of a new point and projection of class means.

We impose an equality constraint on the denominator of $\frac{\mathbf{v}^\top \mathbf{B}\mathbf{v}}{\mathbf{v}^\top \Sigma \mathbf{v}}$ and solve the optimization problem

$$\begin{aligned} \max \quad & \mathbf{v}^\top \mathbf{B}\mathbf{v} \\ \text{s.t.} \quad & \mathbf{v}^\top \Sigma \mathbf{v} = 1 \end{aligned}$$

Using the fact that Σ is symmetric and positive definite, our Lagrange condition is given by

$$\begin{aligned} 2\mathbf{B}\mathbf{v} - 2\lambda\Sigma\mathbf{v} &= 0 \\ \mathbf{B}\mathbf{v} &= \lambda\Sigma\mathbf{v} \\ \Sigma^{-1}\mathbf{B}\mathbf{v} &= \lambda\mathbf{v} \end{aligned}$$

Thus, we have that the vectors satisfying the first-order condition must be eigenvectors of $\Sigma^{-1}\mathbf{B}$. Due to our norm constraint, we have that $\mathbf{v}^\top \mathbf{B}\mathbf{v} = \lambda \mathbf{v}^\top \Sigma \mathbf{v} = \lambda$, so our maximizing vector $\mathbf{v}$ must be the eigenvector associated with the largest eigenvalue of $\Sigma^{-1}\mathbf{B}$.

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

Subsection 1

Multiclass Classifier: Star Types

Star Classification: Background

We aim to classify star types given a set of stars of the following kind: **brown dwarf, red dwarf, white dwarf, main sequence, supergiants, and hypergiants.**

Our data includes the following attributes of the stars: temperature, luminosity, radius, absolute magnitude, star color, and spectral type, which depends on all the previous attributes.

Below is a table of how stars are grouped by spectral type.

Class	Effective temperature[1][2]	Vega-relative chromaticity[3][4][a]	Chromaticity (D65)[5][6][3][b]	Main-sequence mass[1][7] (solar masses)	Main-sequence radius[1][7] (solar radii)	Main-sequence luminosity[1][7] (bolometric)
O	≥ 30,000 K	blue	blue	≥ 16 $M_\odot$	≥ 6.6 $R_\odot$	≥ 30,000 $L_\odot$
B	10,000–30,000 K	blue white	deep blue white	2.1–16 $M_\odot$	1.8–6.6 $R_\odot$	25–30,000 $L_\odot$
A	7,500–10,000 K	white	blue white	1.4–2.1 $M_\odot$	1.4–1.8 $R_\odot$	5–25 $L_\odot$
F	6,000–7,500 K	yellow white	white	1.04–1.4 $M_\odot$	1.15–1.4 $R_\odot$	1.5–5 $L_\odot$
G	5,200–6,000 K	yellow	yellowish white	0.8–1.04 $M_\odot$	0.96–1.15 $R_\odot$	0.6–1.5 $L_\odot$
K	3,700–5,200 K	light orange	pale yellow orange	0.45–0.8 $M_\odot$	0.7–0.96 $R_\odot$	0.08–0.6 $L_\odot$
M	2,400–3,700 K	orange red	light orange red	0.08–0.45 $M_\odot$	≤ 0.7 $R_\odot$	≤ 0.08 $L_\odot$

Figure: Harvard Spectral Classification
Source: `https://en.wikipedia.org/wiki/Stellar_classification`

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

Star Classification: Is spectral type a problem?

Figure 1 shows spectral type's dependence on temperature, radius, and luminosity. So we need to ask ourselves if spectral type will be a problem in our model.

Kind of. Consider the Hertzsprung-Russell (HR) diagram:

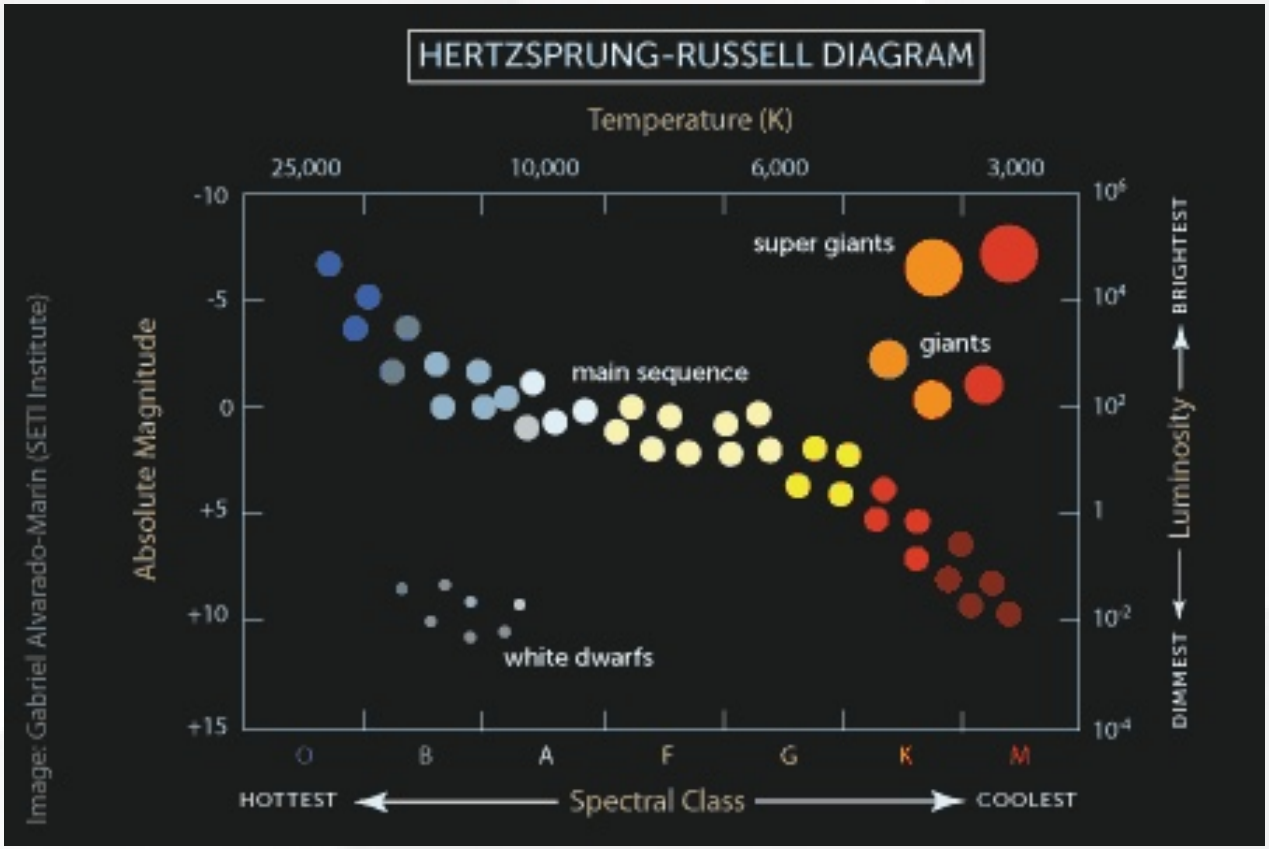


Figure: Hertzsprung-Russell Diagram

Implementation

```
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis \
    as LDA
from sklearn.metrics import accuracy_score

X, y = data.drop(columns=['Spectral class','Star type']),
                    data['Star type']

# Split data into 70% train, 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y,
    train_size=0.7, random_state=0)

model = LDA()
model.fit(X_train, y_train)
predicted = model.predict(X_test)

print("Model accuracy:", accuracy_score(y_test, predicted))

    >>> Model accuracy: 0.9861111111111112
```

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

More Robust Accuracy Analysis

```
def LDA_error(X, y, train_size, random_state):
  # Define train and test sets
  X_train, X_test, y_train, y_test = train_test_split(X, y,
           train_size=train_size, random_state=random_state)
  model = LDA().fit(X_train, y_train)
  return accuracy_score(y_test, model.predict(X_test))

n_iter = 50
errors = [LDA_error(X, y, train_size=0.7, random_state=seed)
          for seed in range(n_iter)]

print(f"Model accuracy over {n_iter} splits")
print("Mean:   ", np.mean(errors))
print("Std dev:", np.std(errors))
```

```
>>> Model accuracy over 50 splits
>>> Mean:    0.9886111111111111
>>> Std dev: 0.011348165675453048
```

Just because we were curious...

What happens if we project the data to 2D? Here's what we found:

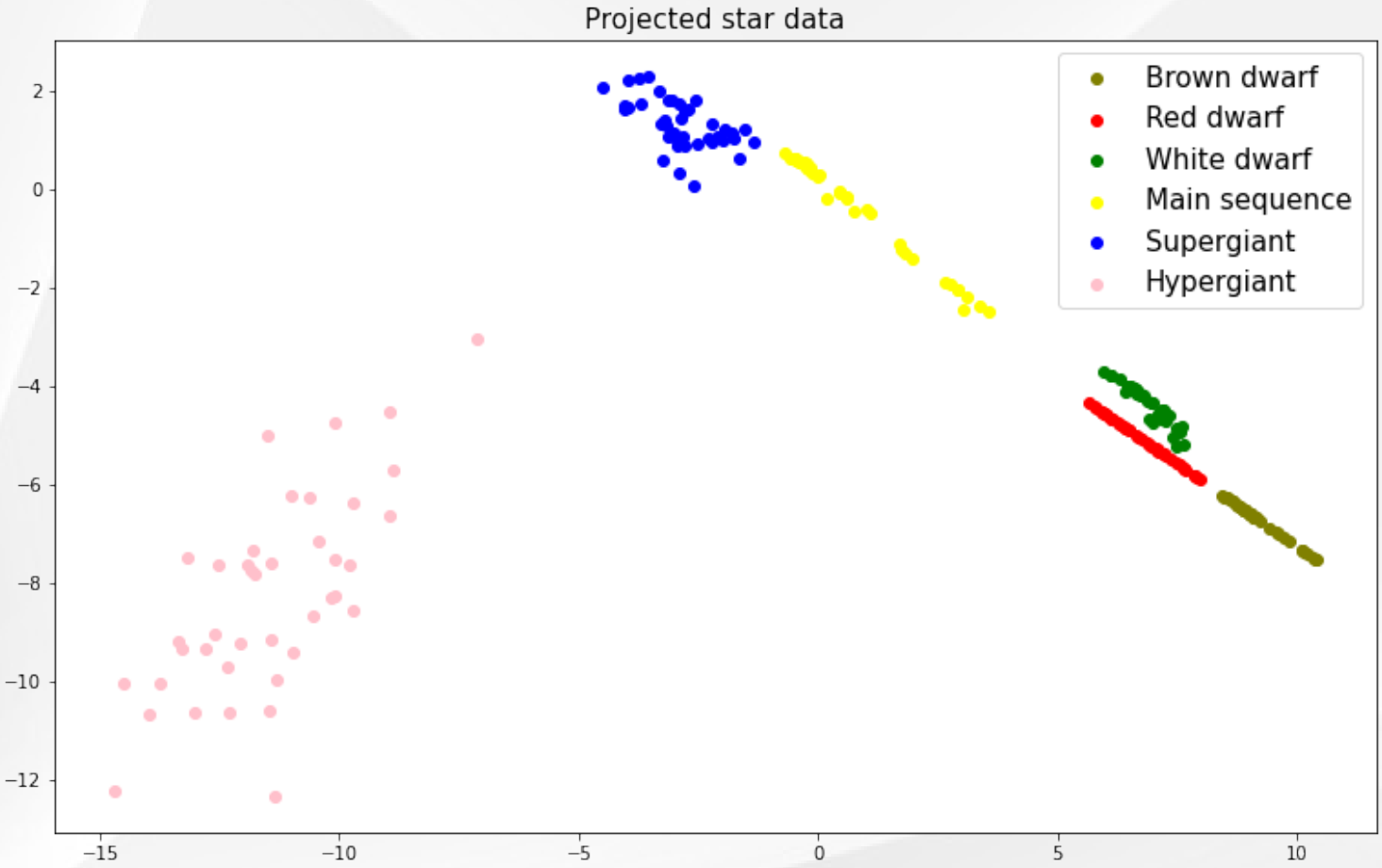


Figure: LDA Projection of star data. It looks a bit like we've reconstructed a HR diagram!

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

Section 4

LDA for Dimensionality Reduction

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

Subsection 1

Comparison with PCA

Why Supervise Dimensionality Reduction?

Given our knowledge of other dimensionality reduction algorithms like PCA, how is LDA different and what benefits does it provide?

Rather than explain the theoretical differences between the two algorithms, we will instead demonstrate how they perform on a toy dataset. We will be using the familiar Red Wine Quality dataset, which is built in to scikit-learn.

[Note: The difference between the built-in dataset and the one we used in Project 1 is that the target variable, wine quality, is split into three classes (low, medium, and high) based on their scores.]

The target variable, wine quality, depends on 13 attributes, including alcohol content, color and hue, and the concentrations of various chemical compounds. We will reduce the 13-dimensional feature vectors down to 2 dimensions and plot the results.

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

The Code

```
from sklearn.datasets import load_wine
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis \
    as LDA
from sklearn.decomposition import PCA

# Load and separate data
wine = load_wine()
X, y = wine.data, wine.target

models = [LDA(n_components=2), PCA(n_components=2)]
proj = [m.fit_transform(X, y) for m in models]

# Plot the projected data
fig, ax = plt.subplots(1, 2, figsize=(19, 10))
model_names = ['LDA', 'PCA']
colors = ['red', 'blue', 'darkorange']
for i in range(2):
    for label in range(3):
        ax[i].scatter(proj[i][y==label][:,0], proj[i][y==label][:,1],
                      color=colors[label])
    ax[i].set_title(model_names[i] + " Projection", fontsize=18)
fig.show()
```

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

The Results

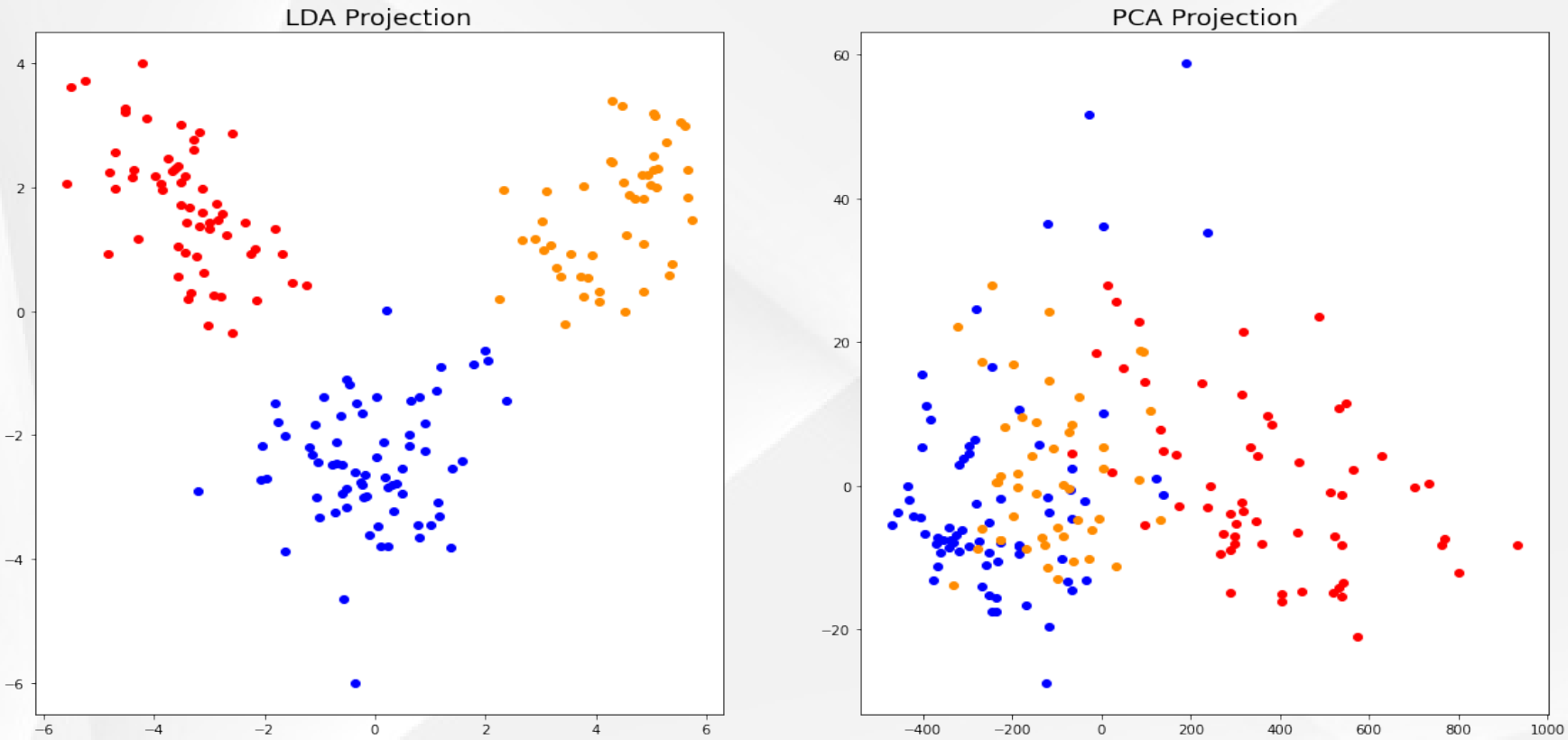


Figure: LDA and PCA projection of the Red Wine dataset

You can see the presence of the class labels accomplished something entirely different to PCA.

Why so different?

While PCA finds a projection subspace that aims to preserve what sets each data point apart from all the other data points, LDA does the exact opposite. With LDA, we *want* the data points to be grouped together according to their class labels, so variance is necessarily destroyed in the projection process. The following figure illustrates the difference in chosen subspaces:

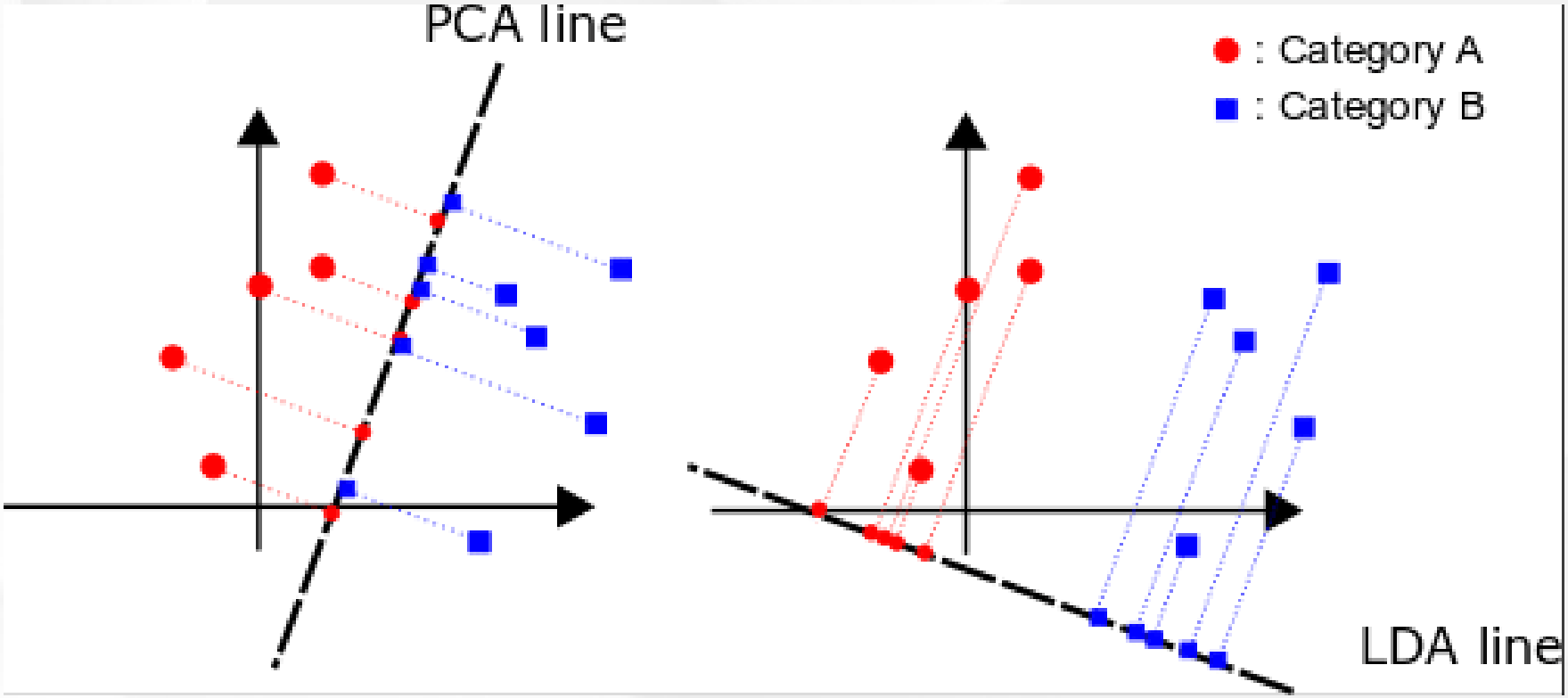


Figure: The projection subspaces in this example are nearly orthogonal
Credit: Thériault et al.

Subsection 2

Pushing the Limits of LDA: Chest X-Rays

Problem

- We use LDA-based dimensionality reduction to diagnose patients with pneumonia based on chest X-ray images. Our training set of 5000 images labels X-rays with **normal**, **bacterial pneumonia**, and **viral pneumonia**.
- Two-Stage Training Model:
    1. Retain initial classifications and run LDA to project data to two dimensions.
    2. Group bacterial and viral pneumonia under the same class, and train three different binary classifiers (logistic regression, LDA, and SVM) on the transformed training set.

Viewing the Data

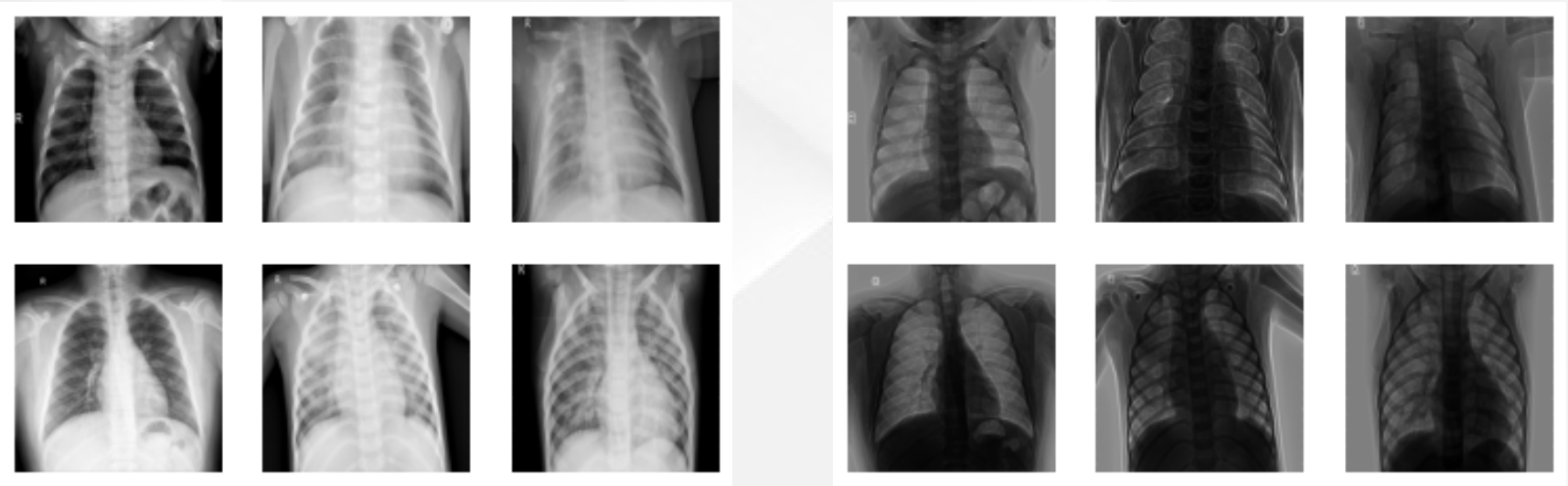


Figure: Originals and negatives of X-ray samples

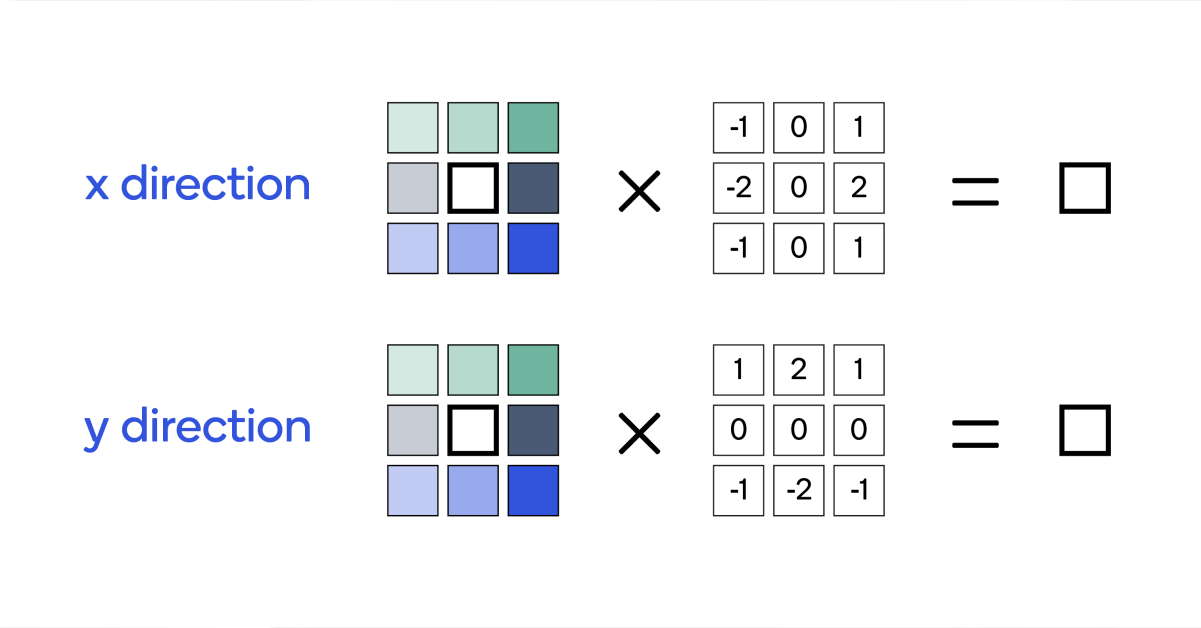
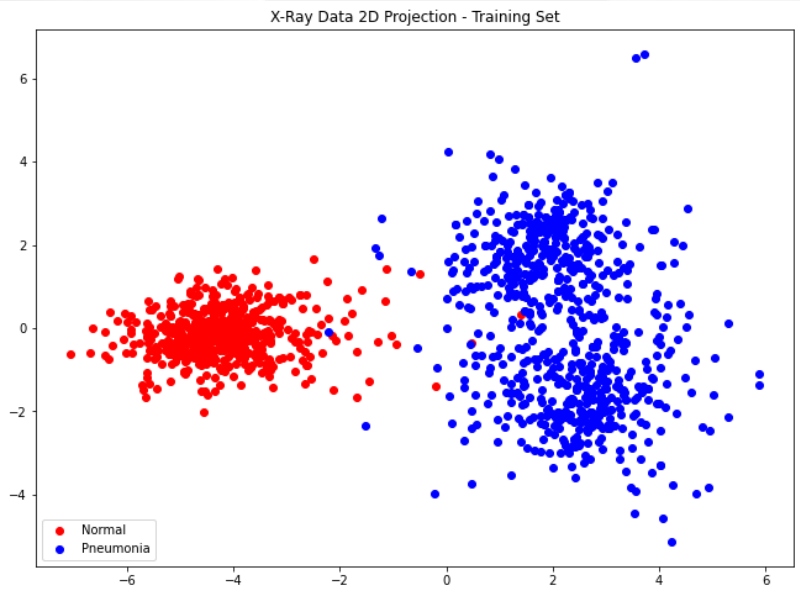
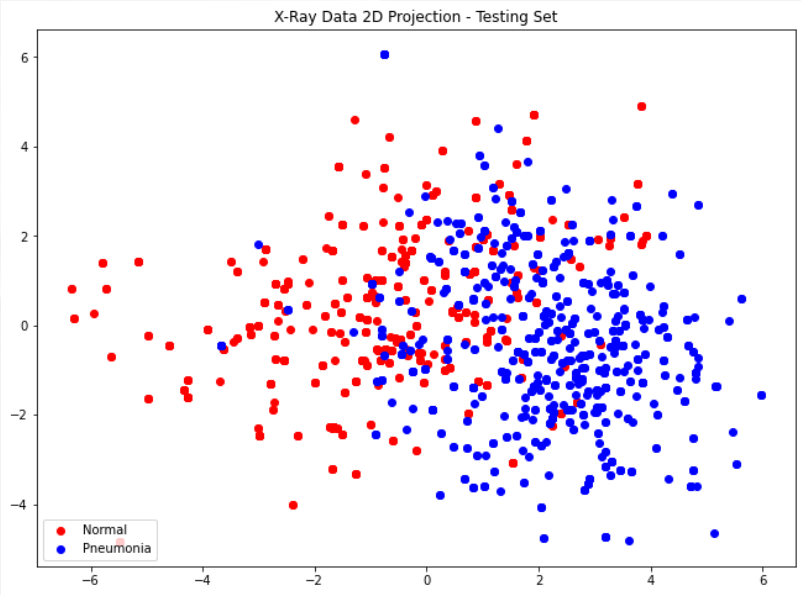


Figure: Applying the Sobel Filter
Source: `https://developer.qualcomm.com/sites/default/files/attachments/`

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

Projection Results





For all three second-stage binary classifiers, the accuracy of our two-stage model was approximately 95% for the training set and 70% for the testing set. By preprocessing the images with a Sobel filter, our test accuracy improved to 75%.

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo

Thanks for listening!

Feel free to ask questions now or on Campuswire!

Our full code and presentation material is available on a Github repository here:
`https://github.com/warewaware/LDAFinalProj`

© Aatmun Baxi, Abhijith Vemulapati, Andy Chen, Johnny Mo