# Unsupervised Learning II

Dr. Alex Williams

October 21, 2020

THE UNIVERSITY OF TENNESSEE KNOXVILLE

COSC 425: Introduction to Machine Learning
Fall 2020 (CRN: 44874)

# Today's Agenda

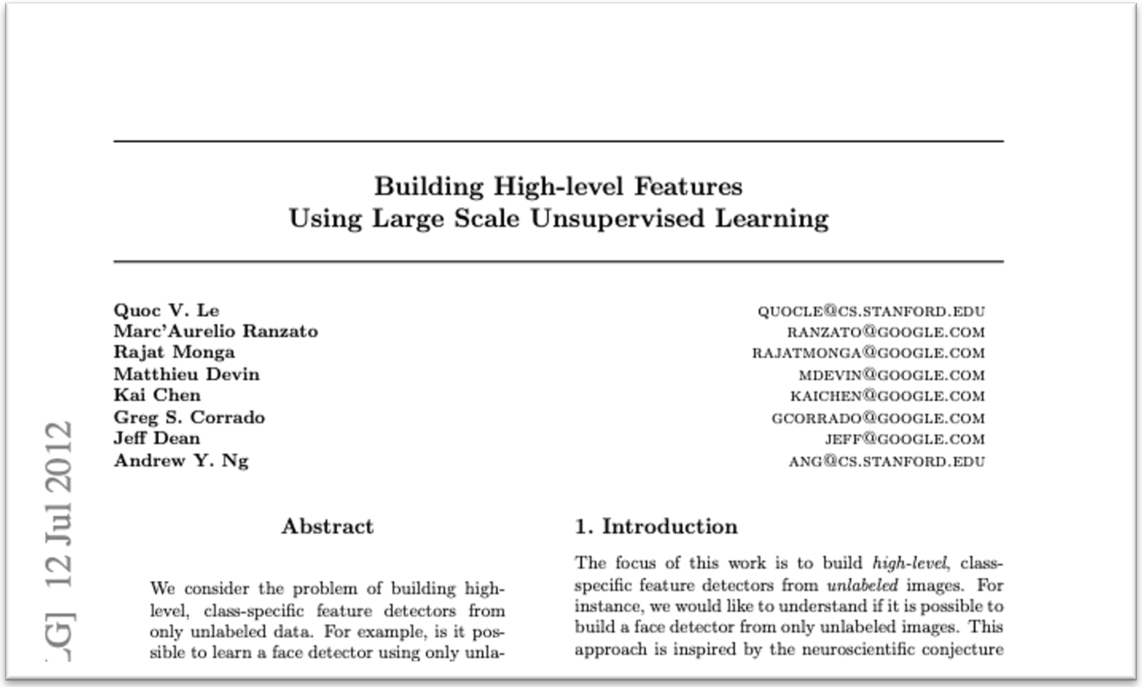
**We will address:**

1. Dimensionality Reduction with PCA

# Unsupervised Learning: Feature Selection

**Long-Term Goal.**
- Figure out which inputs matter.

**Feasible, but Challenging.**
- Data, data, and more data.



G] 12 Jul 2012

Building High-level Features
Using Large Scale Unsupervised Learning

Quoc V. Le QUOCLE@CS.STANFORD.EDU
Marc'Aurelio Ranzato RANZATO@GOOGLE.COM
Rajat Monga RAJATMONGA@GOOGLE.COM
Matthieu Devin MDEVIN@GOOGLE.COM
Kai Chen KAICHEN@GOOGLE.COM
Greg S. Corrado GCORRADO@GOOGLE.COM
Jeff Dean JEFF@GOOGLE.COM
Andrew Y. Ng ANG@CS.STANFORD.EDU

Abstract

We consider the problem of building high-level, class-specific feature detectors from only unlabeled data. For example, is it possible to learn a face detector using only unla-

1. Introduction

The focus of this work is to build *high-level*, class-specific feature detectors from *unlabeled* images. For instance, we would like to understand if it is possible to build a face detector from only unlabeled images. This approach is inspired by the neuroscientific conjecture

[+2000 Citations!](https://arxiv.org/pdf/1112.6209.pdf)
https://arxiv.org/pdf/1112.6209.pdf

# Supervised Learning: Deterrents

**Challenges in Practice**

→ Finding a ”labeler” may be difficult, expensive or impossible.

→ Unsupervised learning is concerned with learning without a teacher.

| | isUTKEmail | HeaderKeyword | Word 1 | Word 2 | | sSp |
|---|---|---|---|---|---|---|
| x1 | Yes | CS425 | Hi | Prof | … | [illegible] |
| x2 | Yes | Orientation | Alex | You | … | [illegible] |
| x2 | No | urgent | Dear | Sir | … | [illegible] |
| x4 | No | cash | hello | I | … | [illegible] |
| x5 | No | help | are | you | … | [illegible] |
| x6 | Yes | Survey | Faculty | this | … | [illegible] |
| | | | … | | | |

# Unsupervised Learning

**In unsupervised learning, data consists only of examples and not the corresponding labels.**

→ Our job is to make sense of or find some pattern of regularity in the data even though no one has provided correct labels.

For example, we might want to do:

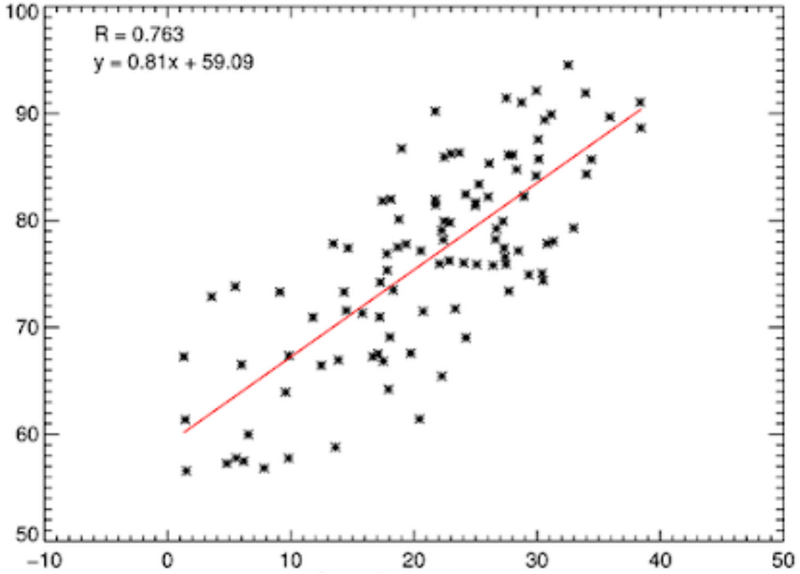
- **Clustering**: Automatically partition the data into groups.
- **Dimensionality Reduction**: Project high dimension data into lower dimension space, so it can be more easily visualized.

# Overview

1. **Dimensionality Reduction with PCA**

# Dimensionality Reduction

- **mpg**: continuous
- **cylinders**: multi-valued discrete
- **displacement**: continuous
- **horsepower**: continuous
- **weight**: continuous
- **acceleration**: continuous
- **model year**: multi-valued discrete
- **origin**: multi-valued discrete
- **car name**: string (unique for each instance)




**How do we automatically detect and remove redundant dimensions?**

We could've done this naively by looking at R-squared metrics in Project 2.

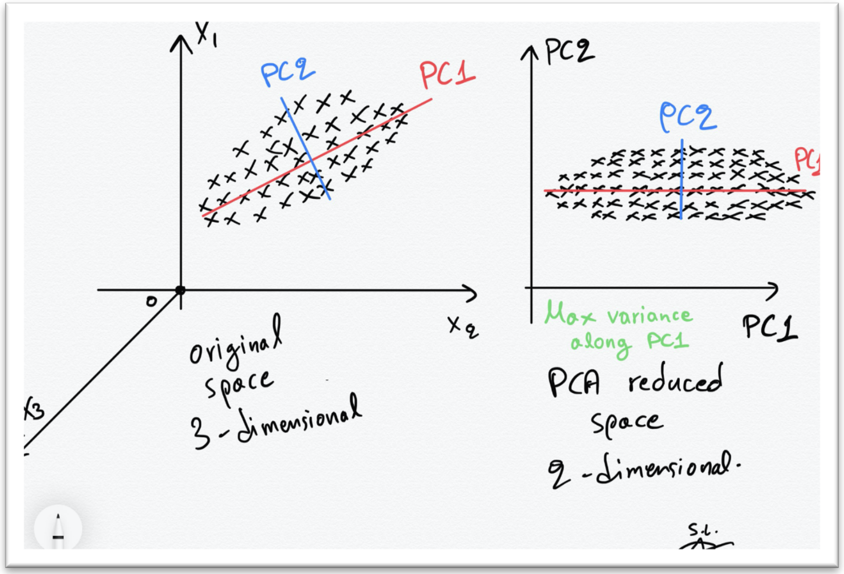
# Principal Component Analysis

Dimensionality reduction is the task of taking a dataset in high dimensions (say 10,000) and reducing it to low dimensions (say 2) while retaining the "important" characteristics of the data.
→ Unsupervised learning context = important is difficult to define.

## Principal Component Analysis (PCA)

→ Extracts "important" information from a multivariate dataset and expresses the information as a set of new variables
  → ***"Principal Components"***
    Number is less than or equal to # of features.

→ Allows us to summarize and to visualize information in a dataset of high dimensions.
  → 2D to 1D
  → 3D to 2D




**Credit:** Towards Data Science

# PCA: Solution + Algorithm

PCA operates by looking for a vector ***u*** in your feature space with the maximal variance.

**Procedure**

1. Normalize Features (This is a pre-processing step!)
   → Ensure each feature has a zero mean and is scales accordingly.

2. Compute Covariance Matrix
   → Quantifies variance between features.

3. Compute Eigenvectors
   → Vectors that allow us to transform our dimensionality.

4. Keep the first ***k*** Eigenvectors and project to get new features ***z***.
   → Translated into principal components.

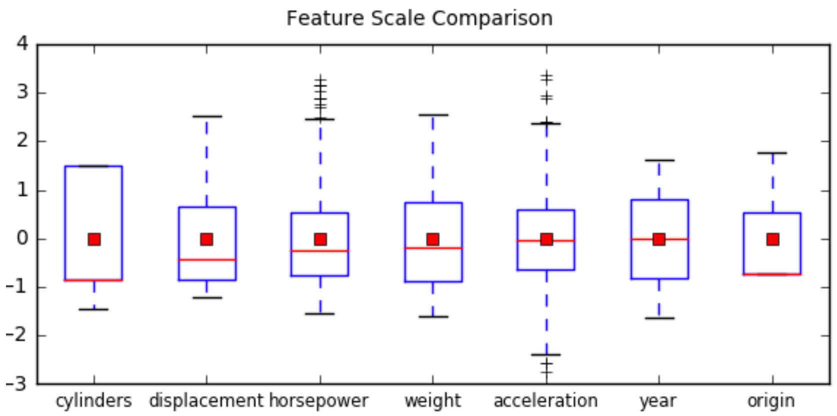
# Normalization and Scaling

**Normalization** is PCA's pre-processing step that involves computing the mean of all features to orient around zero.
→ Also known as "Mean Normalization".
→ Each $x_j^i$ is replaced with $x_j - \mu_j$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

## Scaling

You should make your features are on the same scale
→ $X_1$ = Size of House
→ $X_2$ = Number of Bedrooms




We need to scale features to have a *comparable* range of values.
→ This is precisely what we did in our Linear Regression assignment.

# PCA: Example

**Step 1: Apply mean normalization.**

Data =

| $x$ | $y$ |
|---|---|
| 2.5 | 2.4 |
| 0.5 | 0.7 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3.0 |
| 2.3 | 2.7 |
| 2 | 1.6 |
| 1 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

$$x_i - \bar{x}$$
$$y_i - \bar{y}$$
$$\forall i = 1,\ldots,10 \longrightarrow$$

DataAdjust =

| $x$ | $y$ |
|---|---|
| .69 | .49 |
| -1.31 | -1.21 |
| .39 | .99 |
| .09 | .29 |
| 1.29 | 1.09 |
| .49 | .79 |
| .19 | -.31 |
| -.81 | -.81 |
| -.31 | -.31 |
| -.71 | -1.01 |

Clark, 2002. PCA.
http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

# PCA: Example

**Step 2: Calculate Covariance Matrix**

Covariance measures how two variables vary from the mean with respect to one another.

$$var(X) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

$$cov(X) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

Covariance matrix captures covariance values between all dimensions.

$$C = \begin{pmatrix} \overset{1}{cov(x, x)} & \overset{2}{cov(x, y)} \\ \overset{3}{cov(y, x)} & \overset{4}{cov(y, y)} \end{pmatrix}$$

$$C = \begin{pmatrix} 0.6166 & \boxed{0.6154} \\ \boxed{0.6154} & 0.7166 \end{pmatrix}$$

**non-diagonal values are positive, indicating that x increases as y increases.**

# PCA: Example

**Step 3: Calculate the Eigenvectors and Eigenvalues of the Covariance matrix.**

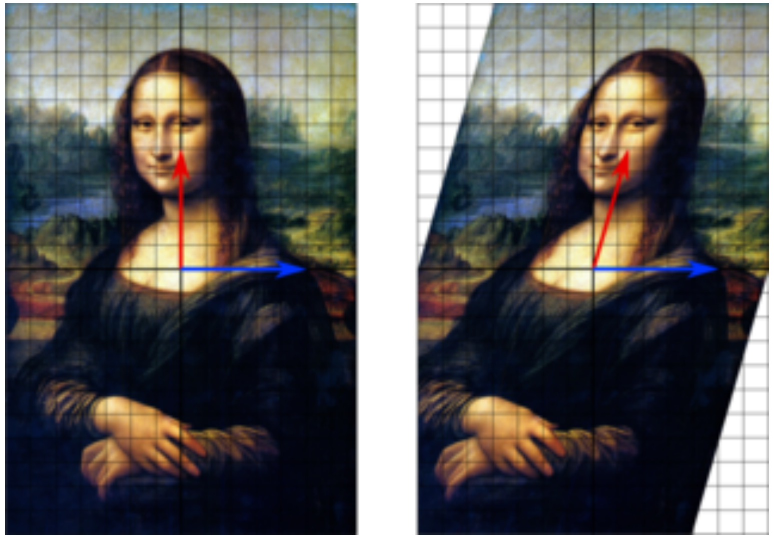
Eigenvector v of a linear transformation is a vector that, upon transformation, does not change direction.

$$Av = \lambda v$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Here, the associated eigenvalue is 4.

→ Not that all eigenvectors of a matrix are orthogonal (i.e. perpendicular) to each other. We can re-express the data using eigenvectors as new axes.



**Blue Arrow = Eigenvector**

→ Doesn't change direction

→ Eigenvector is 1 because length doesn't change.

# PCA: Example

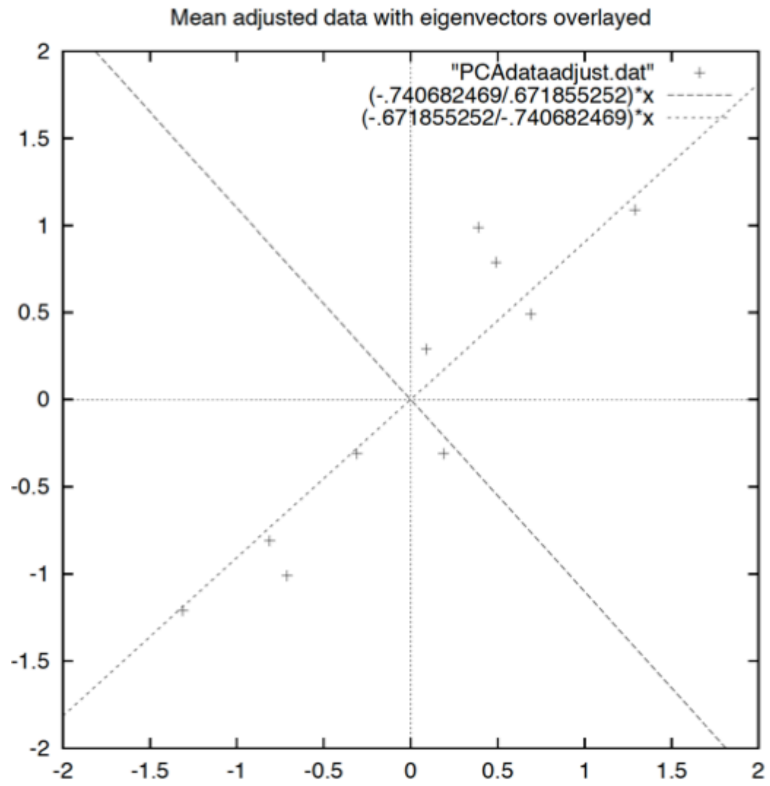
**Step 3: Calculate the Eigenvectors and Eigenvalues of the Covariance matrix.**

The Unit Eigenvectors

$$eigenvectors = \begin{pmatrix} -0.7352 & -0.6779 \\ 0.6779 & -0.7352 \end{pmatrix}$$

The Corresponding Eigenvalues

$$eigenvalues = \begin{pmatrix} 0.049 \\ 1.284 \end{pmatrix}$$



# PCA: Example

**Step 4: Choose Principal Components**

The Eigenvector with the highest eigenvalue is the principal component of the dataset.

$$eigenvectors = \begin{pmatrix} -0.7352 & \boxed{-0.6779} \\ 0.6779 & \boxed{-0.7352} \end{pmatrix} \qquad eigenvalues = \begin{pmatrix} 0.049 \\ \boxed{1.284} \end{pmatrix}$$

Form a matrix of k eigenvectors, ordered by eigenvalues from largest to smallest.

$$W = [eig_1, eig_2, \ldots, eig_k]$$

$$W = \begin{pmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{pmatrix}$$

**If $k < m$, then you are essentially discarding some dimensions. e.g.,**

$$W = \begin{pmatrix} -0.6779 \\ -0.7352 \end{pmatrix}$$

# PCA: Example

**Step 5: Derive the new Dataset**

Multiply the transposition of W on the left of the mean-adjusted dataset, transposed.

**Eigenvectors (Step 4)** **Mean Normalized Data (Step 1)**

$$\begin{pmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} \begin{pmatrix} 0.69 & -1.31 & 0.39 & 0.09 & 1.29 & 0.49 & 0.19 & -0.81 & -0.31 & -0.71 \\ 0.49 & -1.21 & 0.99 & 0.29 & 1.09 & 0.79 & -0.31 & -0.81 & -0.31 & -1.01 \end{pmatrix}$$

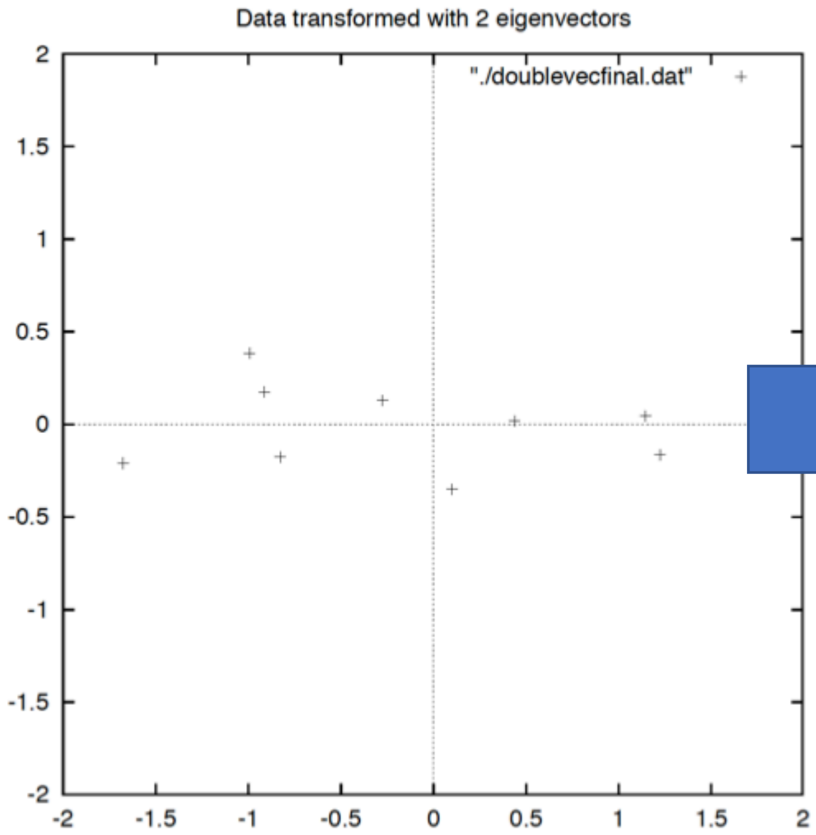
=

| $x$ | $y$ |
|---|---|
| -.827970186 | -.175115307 |
| 1.77758033 | .142857227 |
| -.992197494 | .384374989 |
| -.274210416 | .130417207 |
| -1.67580142 | -.209498461 |
| -.912949103 | .175282444 |
| .0991094375 | -.349824698 |
| 1.14457216 | .0464172582 |
| .438046137 | .0177646297 |
| 1.22382056 | -.162675287 |

# PCA: Example

**Step 5: Derive the new Dataset**

| $x$ | $y$ |
|---|---|
| -.827970186 | -.175115307 |
| 1.77758033 | .142857227 |
| -.992197494 | .384374989 |
| -.274210416 | .130417207 |
| -1.67580142 | -.209498461 |
| -.912949103 | .175282444 |
| .0991094375 | -.349824698 |
| 1.14457216 | .0464172582 |
| .438046137 | .0177646297 |
| 1.22382056 | -.162675287 |




Transformed Data (Single eigenvector)

| $x$ |
|---|
| -.827970186 |
| 1.77758033 |
| -.992197494 |
| -.274210416 |
| -1.67580142 |
| -.912949103 |
| .0991094375 |
| 1.14457216 |
| .438046137 |
| 1.22382056 |

# PCA with Sklearn

PCA-Example > pca.py > ...

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

**Versions**

- Python 3.7
- Matplotlib=3.1.2
- Numpy =1.17.2
- Pandas=0.25.1
- Scikit-learn=0.23.2

**pca.py on Canvas**

# PCA with Sklearn

```python
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# load dataset into Pandas DataFrame
df = pd.read_csv(url, names=['sepal length','sepal width','petal length','petal width','target'])
```

| | sepal length | sepal width | petal length | petal width | target |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

**The Iris Dataset**

# PCA with Sklearn

| | sepal length | sepal width | petal length | petal width |
|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 |

Standardization →

| | sepal length | sepal width | petal length | petal width |
|---|---|---|---|---|
| 0 | -0.900681 | 1.032057 | -1.341272 | -1.312977 |
| 1 | -1.143017 | -0.124958 | -1.341272 | -1.312977 |
| 2 | -1.385353 | 0.337848 | -1.398138 | -1.312977 |
| 3 | -1.506521 | 0.106445 | -1.284407 | -1.312977 |
| 4 | -1.021849 | 1.263460 | -1.341272 | -1.312977 |

```
features = ['sepal length', 'sepal width', 'petal length', 'petal width']
# Separating out the features
x = df.loc[:, features].values
# Separating out the target
y = df.loc[:,['target']].values
# Standardizing the features
x = StandardScaler().fit_transform(x)
```

**Standardization**

# PCA with Sklearn

| | sepal length | sepal width | petal length | petal width |
|---|---|---|---|---|
| 0 | -0.900681 | 1.032057 | -1.341272 | -1.312977 |
| 1 | -1.143017 | -0.124958 | -1.341272 | -1.312977 |
| 2 | -1.385353 | 0.337848 | -1.398138 | -1.312977 |
| 3 | -1.506521 | 0.106445 | -1.284407 | -1.312977 |
| 4 | -1.021849 | 1.263460 | -1.341272 | -1.312977 |

PCA
(2 components)

| | principal component 1 | princial component 2 |
|---|---|---|
| 0 | -2.264542 | 0.505704 |
| 1 | -2.086426 | -0.655405 |
| 2 | -2.367950 | -0.318477 |
| 3 | -2.304197 | -0.575368 |
| 4 | -2.388777 | 0.674767 |

```python
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

**Apply PCA: 4D to 2D Space**

# PCA with Sklearn

| | principal component 1 | principal component 2 |
|---|---|---|
| 0 | -2.264542 | 0.505704 |
| 1 | -2.086426 | -0.655405 |
| 2 | -2.367950 | -0.318477 |
| 3 | -2.304197 | -0.575368 |
| 4 | -2.388777 | 0.674767 |

principalDf

| | target |
|---|---|
| 0 | Iris-setosa |
| 1 | Iris-setosa |
| 2 | Iris-setosa |
| 3 | Iris-setosa |
| 4 | Iris-setosa |

df[['target']]

pd.concat(axis = 1)

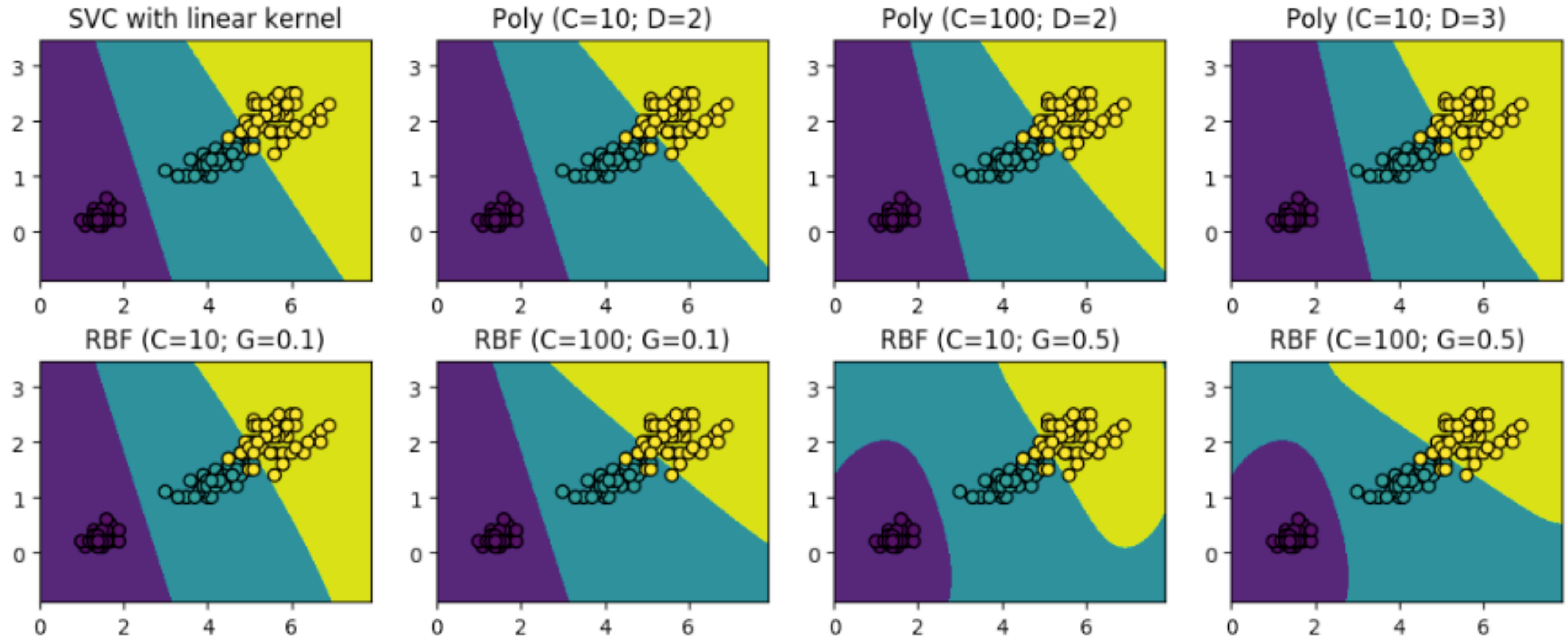
| | principal component 1 | principal component 2 | target |
|---|---|---|---|
| 0 | -2.264542 | 0.505704 | Iris-setosa |
| 1 | -2.086426 | -0.655405 | Iris-setosa |
| 2 | -2.367950 | -0.318477 | Iris-setosa |
| 3 | -2.304197 | -0.575368 | Iris-setosa |
| 4 | -2.388777 | 0.674767 | Iris-setosa |

finalDf

```
finalDf = pd.concat([principalDf, df[['target']]], axis = 1)
```

**Concatenate the Target Label into the PC dataframe**

# SVMs with Scikit-learn

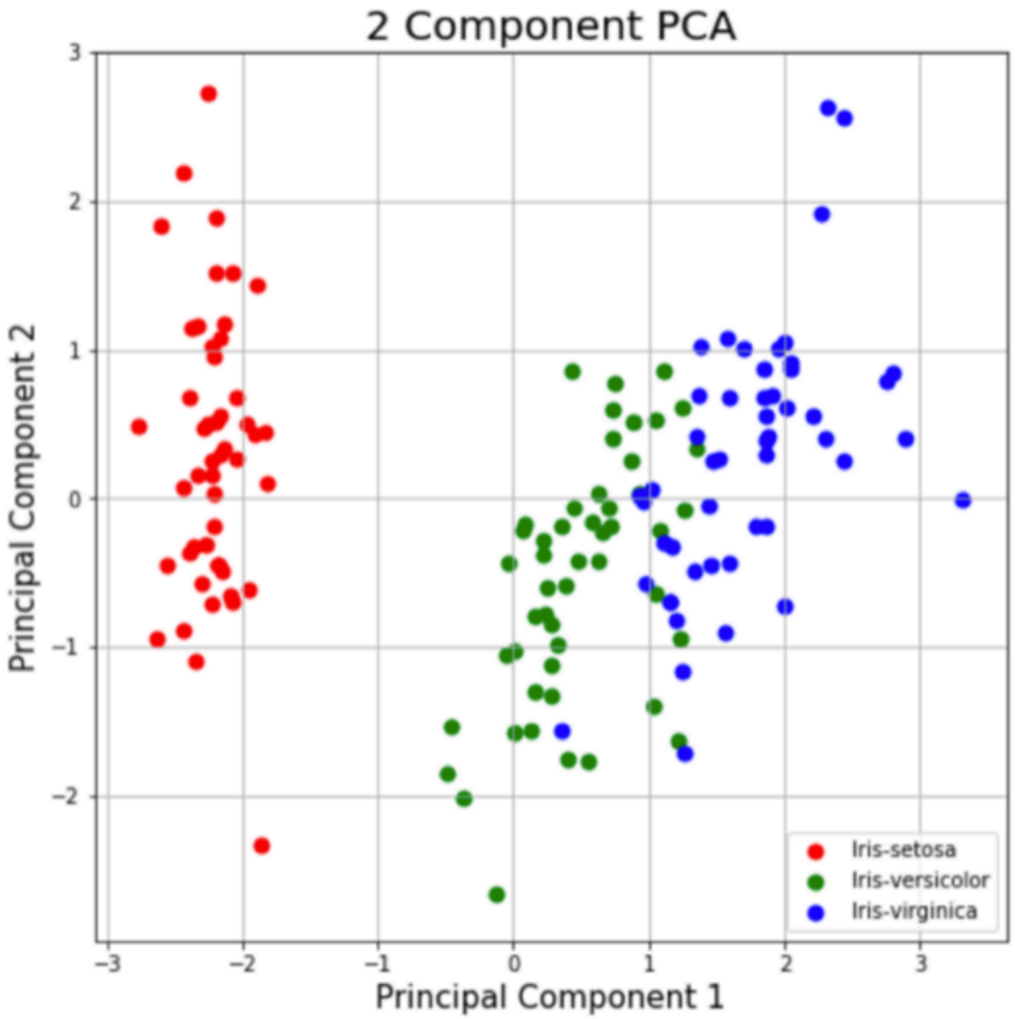



**Visualizing Boundaries**
Source code included in the svm.py file on Canvas.

# PCA with Sklearn

```python
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
plt.show()
```

**Plot PCs in 2D Space**



# PCA with Sklearn

```python
print("How much of our variance is explained?")
print(pca.explained_variance_ratio_)
print()
print()

print("Which features matter most?")
print(abs(pca.components_))
```

```
[(base) alex@MacBook-Pro:~/Desktop/COSC425/PCA-Example$ python pca.py
How much of our variance is explained?
[0.72770452 0.23030523]
Which features matter most?
[[0.52237162 0.26335492 0.58125401 0.56561105]
 [0.37231836 0.92555649 0.02109478 0.06541577]]
```

## Explained Variance

→ 72.7% for PC1
→ 23.0% for PC2

**Combined:** 95.7%

→ PCA loses only 4.3% of the "information" in our data.

## Feature Importance

→ $1^{st}$ row = PC1 → Features 1, 3, and 4 matter equally.
→ $2^{nd}$ row = PC2 → Features 2 matters *a lot*!

**OK. Now, what?**

→ Say Feature 3 had low importance in both PC1 and PC2.
   → Remove it from the feature set entirely.

# Next Time

**We will address:**

1. Neural Nets