Numerical Methods and Machine Learning for Image Processing

Week 5, Class 2: Classification and Regression 2

October 26, 2021

Damon M. Chandler and Yi Zhang

Last time: Classification and Regression, Part 2a

Support vector classification (SVC)

1. Motivation and basic idea
2. SVC on created data
3. SVC on banknote data

Today: Classification and Regression, Part 2b

1. Dimensionality reduction and SVC image classification
    - PCA—Principal Components Analysis
    - SVC on MINST database
2. Homework 2
    - Classification into three levels of importance
    - Use features that you measure via code

Today: Classification and Regression, Part 2b

1. Dimensionality reduction and SVC image classification
   - PCA—Principal Components Analysis
   - SVC on MINST database
2. Homework 2
   - Classification into three levels of importance
   - Use features that you measure via code

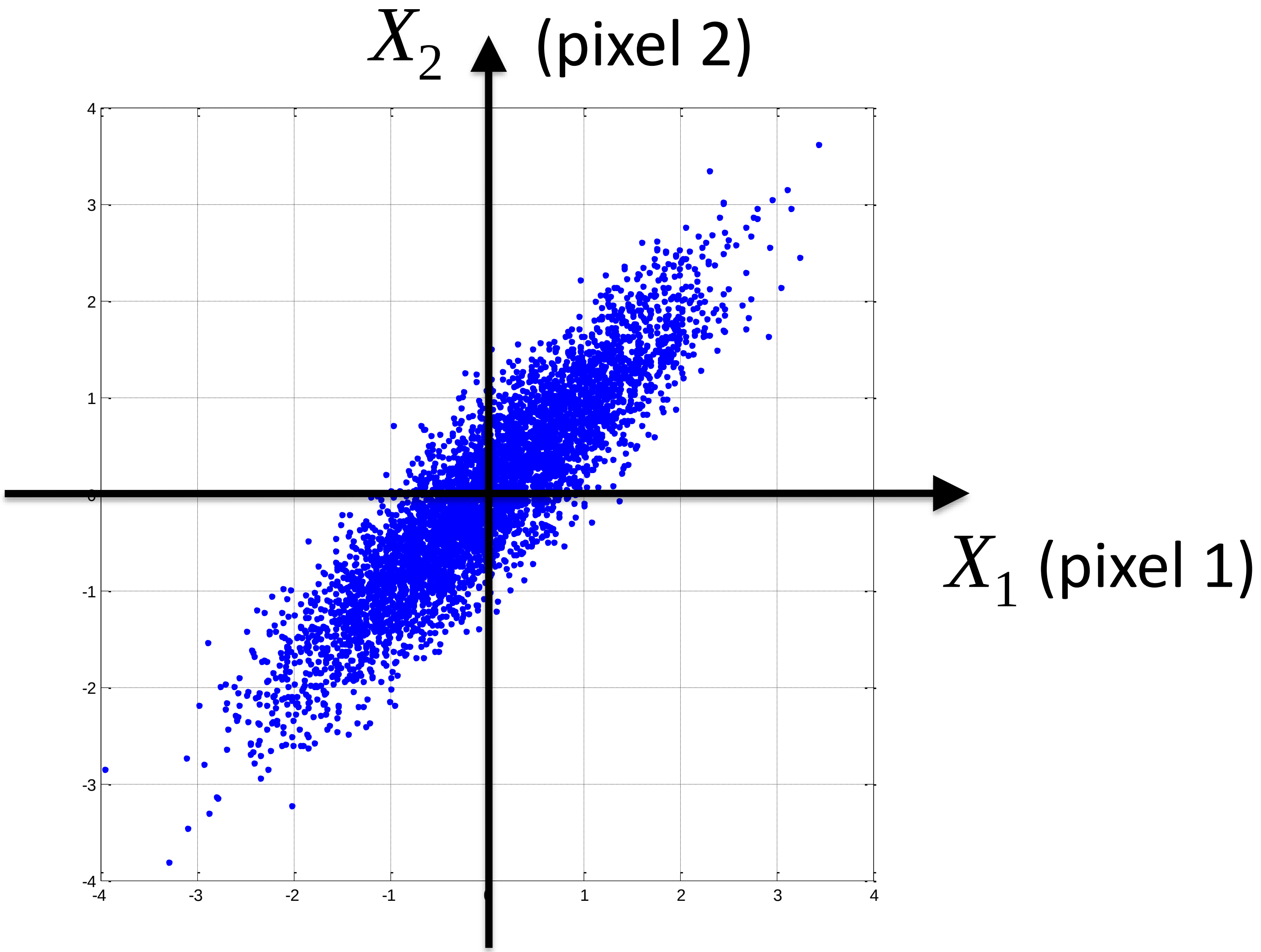
Example: Jointly Gaussian Source (2x1 images)

Suppose we wanted to reduce this 2D data down to 1D.

X1 and X2 both are important, so it's not good to discard one of them.

Can we choose **two new basis vectors**, such that it's OK to use only one of the basis vectors?

Yes! This is a procedure called ***Principal Component Analysis***.



Example: Jointly Gaussian Source (2x1 images)

- 2x1 blocks are realizations of a two-dimensional random vector $\mathbf{X} = [X_1, X_2]^T$
- Pixels X_1 and X_2 are identically distributed and jointly Gaussian (assume zero-mean) with autocorrelation matrix:

$$\mathbf{R}_{\mathbf{X}} = E\{\mathbf{X}\mathbf{X}^T\} = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$$

$$\mathbf{R}_{\mathbf{X}} = E\{\mathbf{X}\mathbf{X}^T\} = E\left\{\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}\begin{bmatrix} X_1 & X_2 \end{bmatrix}\right\} = E\left\{\begin{bmatrix} X_1X_1 & X_1X_2 \\ X_2X_1 & X_2X_2 \end{bmatrix}\right\}$$

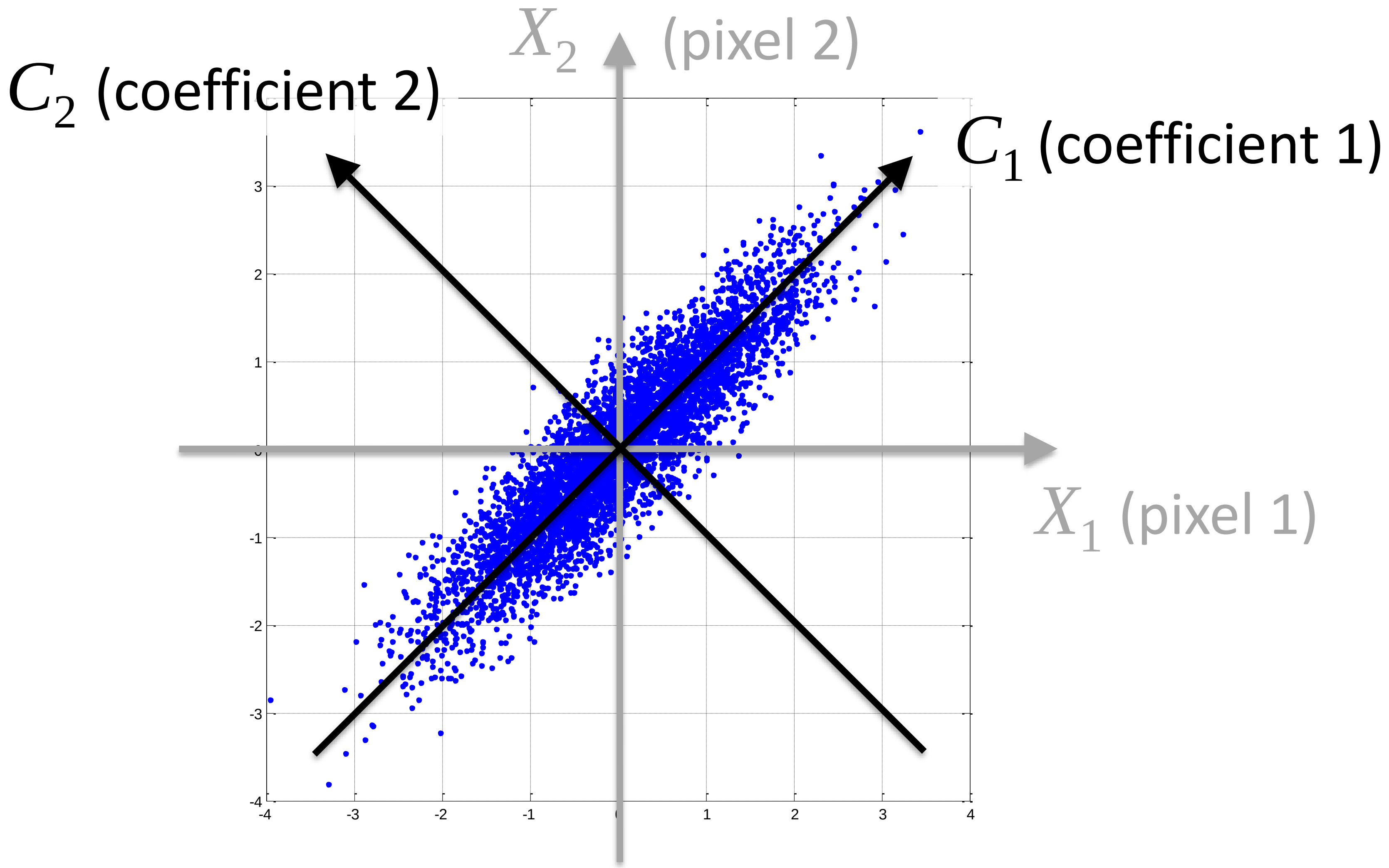
$$= \begin{bmatrix} E\{X_1X_1\} & E\{X_1X_2\} \\ E\{X_2X_1\} & E\{X_2X_2\} \end{bmatrix} = \begin{bmatrix} \mathrm{var}(X_1) & \mathrm{cov}(X_1, X_2) \\ \mathrm{cov}(X_1, X_2) & \mathrm{var}(X_2) \end{bmatrix}$$

Example: Jointly Gaussian Source (2x1 images)

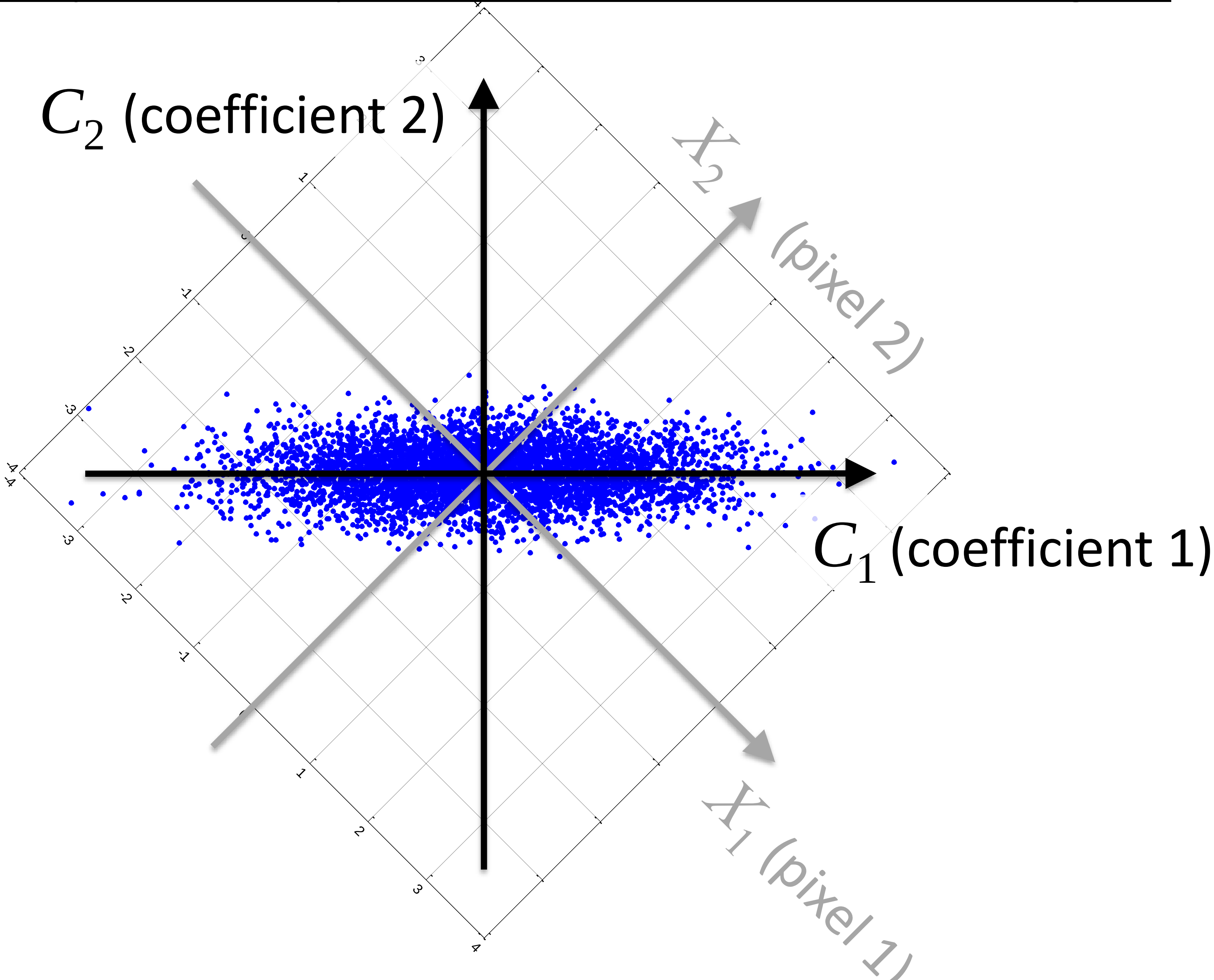
- We seek an **orthogonal transform**, $\mathbf{A}$, that can remove the correlation

$$\mathbf{C} = \mathbf{AX} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

Example: Jointly Gaussian Source (2x1 images)



<u>Example: Jointly Gaussian Source (2x1 images)</u>



Example: Jointly Gaussian Source (2x1 images)

- We seek an orthogonal transform, $\mathbf{A}$, that can remove the correlation

$$\mathbf{C} = \mathbf{AX} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(45°) & \sin(45°) \\ -\sin(45°) & \cos(45°) \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} X_1 + X_2 \\ X_2 - X_1 \end{bmatrix}$$

Example: Jointly Gaussian Source (2x1 images)

- What have we achieved?
  - The rotation does not remove any of the variability
  - It packs the variability into C_1 (“energy compaction”)
  - Now, if C_2 is lost or quantized away, most of the signal energy is still preserved
  - C_1 and C_2 are now **independent** Gaussian variables, so scalar quantization and 1st-order entropy coding are good!
- We have a name for this transform...

$$\mathbf{A} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} X_1 + X_2 \\ X_2 - X_1 \end{bmatrix}$$

For 2D jointly Gaussian data, PCA results in the DCT/Haar DWT/KLT—all the same for 2x1

How to find the transform matrix $\mathbf{A}$?

- Let $\mathbf{X}$ be a zero-mean random vector with autocorrelation matrix $\mathbf{R_X}$
- Our goal is to find a matrix $\mathbf{A}$ such that the components of $\mathbf{C} = \mathbf{AX}$ will be uncorrelated
- Uncorrelated → autocorrelation matrix of $\mathbf{C}$ is diagonal
- The autocorrelation matrix of $\mathbf{C}$ is

$$
\begin{aligned}
\mathbf{R_C} &= E\{\mathbf{CC}^T\} = E\{(\mathbf{AX})(\mathbf{AX})^T\} \\
&= E\{\mathbf{AXX}^T\mathbf{A}^T\} = \mathbf{A}E\{\mathbf{XX}^T\}\mathbf{A}^T \\
&= \mathbf{AR_X A}^T
\end{aligned}
$$

Goal is to find $\mathbf{A}$ such that $\mathbf{R_C}$ is diagonal

How to find the transform matrix $\mathbf{A}$?

- $\mathbf{R}_{\mathbf{C}} = \mathbf{A}\mathbf{R}_{\mathbf{X}}\mathbf{A}^T$
- Note that $\mathbf{R}_{\mathbf{X}} = E\{\mathbf{X}\mathbf{X}^T\}$ is a positive semi-definite matrix

How to find the transform matrix $\mathbf{A}$?

- $\mathbf{R_C} = \mathbf{A}\mathbf{R_X}\mathbf{A}^T$
- Note that $\mathbf{R_X} = E\{\mathbf{X}\mathbf{X}^T\}$ is a positive semi-definite matrix

Recall: A matrix **M** is positive semi-definite if it can be written as the product of another matrix times its transpose:

$$\mathbf{M} = \mathbf{Q}\mathbf{Q}^T$$

How to find the transform matrix $\mathbf{A}$?

- $\mathbf{R}_\mathbf{C} = \mathbf{A}\mathbf{R}_\mathbf{X}\mathbf{A}^T$
- Note that $\mathbf{R}_\mathbf{X} = E\{\mathbf{X}\mathbf{X}^T\}$ is a positive semi-definite matrix
- Two important properties of a positive semi-definite matrix:
  1. Its eigenvalues are always ≥ 0
  2. Its eigenvectors are orthogonal (for different eigenvalues)
- These properties make finding $\mathbf{A}$ straightforward:

$\mathbf{A}$ is the matrix whose rows are the eigenvectors of $\mathbf{R}_\mathbf{X}$

Take-home message: **PCA will compute $\mathbf{A}$ for us. And, in Python, the scikit-learn PCA class to also apply $\mathbf{A}$ to compute the transformed data for us.**

If you need to compute PCA manually, you would: (1) compute R_X, then (2) compute the eigenvectors of R_X, and (3) create A by making these eigenvectors the rows of A.

Today: Classification and Regression, Part 2b

1. Dimensionality reduction and SVC image classification
    - PCA—Principal Components Analysis
    - SVC on MINST database
2. Homework 2
    - Classification into three levels of importance
    - Use features that you measure via code

SVM classification on *banknote* data (2 features only)

```python
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import ipcv_utils.utils as ipcv_plt


#%% LOAD THE DATA
data = pd.read_csv("data/minst_train.csv")

idxs = [2, 16, 7, 3, 8, 21, 29, 20, 28]
for idx in idxs:
  img = np.array(data.iloc[idx, 1:])
  ipcv_plt.imshow(img.reshape(28, 28), cmap="gray",
    vmin=0, vmax=255, zoom=5)

data.head()

X = np.array(data.iloc[:, 1:])
y = np.array(data.iloc[:, 0])

X_trn, X_tst, y_trn, y_tst = train_test_split(X, y,
  test_size=0.5, random_state=42)

print("Training set size (length, dims):", X_trn.shape)
print("Testing set size (length, dims):", X_tst.shape)
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel7
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 785 columns

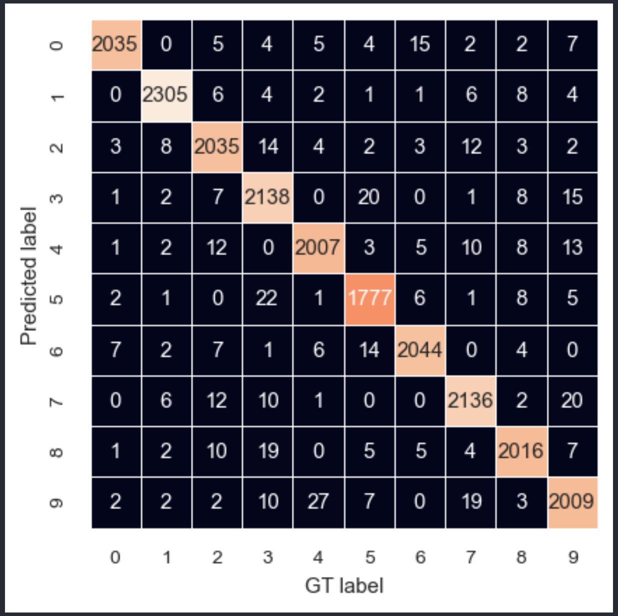
```
Training set size (length, dims): (21000, 784)
Testing set size (length, dims): (21000, 784)
```

SVM classification on *banknote* data (2 features only)

```python
#%% CREATE AND FIT THE SVC
model = SVC(kernel='rbf', C=100)
model.fit(X_trn, y_trn)

y_trn_prd = model.predict(X_trn)
print('Training accuracy: ',
  accuracy_score(y_true=y_trn, y_pred=y_trn_prd))

y_tst_prd = model.predict(X_tst)
print('Testing accuracy: ',
  accuracy_score(y_true=y_tst, y_pred=y_tst_prd))


#%% PRINT THE CLASSIFICATION RESULTS SUMMARIES
print("")
print("Classification Report")
print(classification_report(y_tst, y_tst_prd))

cm = confusion_matrix(y_tst, y_tst_prd)
sns.set(font_scale=0.75)
sns.heatmap(cm.T, square=True, annot=True, fmt='d',
  cbar=False, linewidths=0.5)
plt.xlabel('GT label')
plt.ylabel('Predicted label')
```

```
Training accuracy:  1.0
Testing accuracy:  0.9762857142857143
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      2052
           1       0.99      0.99      0.99      2330
           2       0.98      0.97      0.97      2096
           3       0.98      0.96      0.97      2222
           4       0.97      0.98      0.98      2053
           5       0.97      0.97      0.97      1833
           6       0.98      0.98      0.98      2079
           7       0.98      0.97      0.98      2191
           8       0.97      0.98      0.98      2062
           9       0.97      0.96      0.97      2082

    accuracy                           0.98     21000
   macro avg       0.98      0.98      0.98     21000
weighted avg       0.98      0.98      0.98     21000
```



SVM classification on *banknote* data (2 features only)

```python
#%% THIS TIME, DO PCA FIRST, THEN FIT SVC TO PCA-transformed DATA
pca = PCA(n_components=150,
  svd_solver='randomized', whiten=True)
pca.fit(X_trn)

X_trn_pca = pca.transform(X_trn)
X_tst_pca = pca.transform(X_tst)


#%% CREATE AND FIT THE SVC
model = SVC(kernel='rbf', C=100)
model.fit(X_trn_pca, y_trn)

y_trn_prd = model.predict(X_trn_pca)
print('Training accuracy: ',
  accuracy_score(y_true=y_trn, y_pred=y_trn_prd))
y_tst_prd = model.predict(X_tst_pca)
print('Testing accuracy: ',
  accuracy_score(y_true=y_tst, y_pred=y_tst_prd))


#%% PRINT THE CLASSIFICATION RESULTS SUMMARIES
print("")
print("Classification Report")
print(classification_report(y_tst, y_tst_prd))

cm = confusion_matrix(y_tst, y_tst_prd)
sns.set(font_scale=0.75)
sns.heatmap(cm.T, square=True, annot=True, fmt='d',
  cbar=False, linewidths=0.5)
plt.xlabel('GT label')
plt.ylabel('Predicted label')
```

Reduce to **150 features only**
(150 principal components)

```
Training accuracy:  1.0
Testing accuracy:  0.9675714285714285
```

Classification Report

	precision	recall	f1-score	support
0	0.98	0.99	0.98	2052
1	0.99	0.99	0.99	2330
2	0.95	0.97	0.96	2096
3	0.96	0.95	0.95	2222
4	0.97	0.96	0.97	2053
5	0.96	0.95	0.95	1833
6	0.98	0.98	0.98	2079
7	0.98	0.96	0.97	2191
8	0.95	0.97	0.96	2062
9	0.96	0.96	0.96	2082
accuracy			0.97	21000
macro avg	0.97	0.97	0.97	21000
weighted avg	0.97	0.97	0.97	21000

Predicted label \ GT label	0	1	2	3	4	5	6	7	8	9
0	2033	1	5	3	4	5	14	1	6	7
1	0	2301	4	1	4	1	1	9	5	2
2	4	11	2025	32	18	5	4	24	9	3
3	1	3	8	2100	0	31	0	5	11	22
4	1	3	15	2	1979	4	8	12	4	16
5	2	2	0	36	1	1739	11	2	9	7
6	7	1	5	3	7	21	2032	0	7	0
7	0	6	12	9	2	0	0	2114	2	21
8	2	2	19	21	6	23	9	4	2006	14
9	2	0	3	15	32	4	0	20	3	1990

Similar results with a lot fewer features needed!

SVM classification on *banknote* data (2 features only)

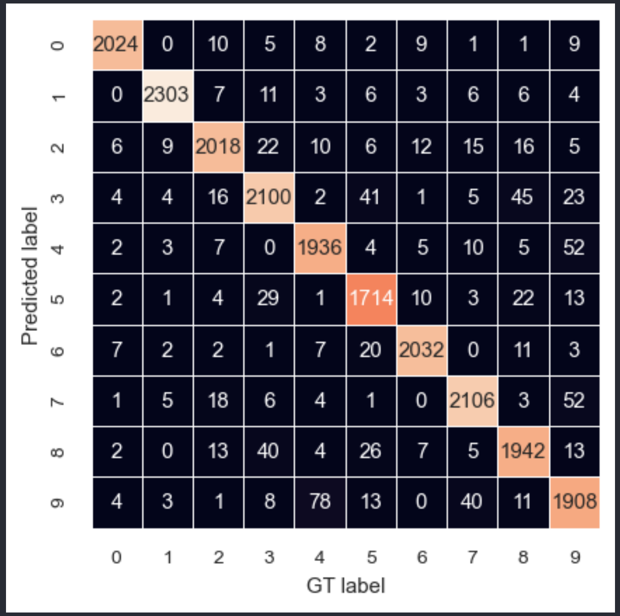
```python
#%% THIS TIME, DO PCA FIRST, THEN FIT SVC TO PCA-transformed DATA
pca = PCA(n_components=15,
  svd_solver='randomized', whiten=True)
pca.fit(X_trn)

X_trn_pca = pca.transform(X_trn)
X_tst_pca = pca.transform(X_tst)


#%% CREATE AND FIT THE SVC
model = SVC(kernel='rbf', C=100)
model.fit(X_trn_pca, y_trn)

y_trn_prd = model.predict(X_trn_pca)
print('Training accuracy: ',
  accuracy_score(y_true=y_trn, y_pred=y_trn_prd))
y_tst_prd = model.predict(X_tst_pca)
print('Testing accuracy: ',
  accuracy_score(y_true=y_tst, y_pred=y_tst_prd))


#%% PRINT THE CLASSIFICATION RESULTS SUMMARIES
print("")
print("Classification Report")
print(classification_report(y_tst, y_tst_prd))

cm = confusion_matrix(y_tst, y_tst_prd)
sns.set(font_scale=0.75)
sns.heatmap(cm.T, square=True, annot=True, fmt='d',
  cbar=False, linewidths=0.5)
plt.xlabel('GT label')
plt.ylabel('Predicted label')
```

Reduce to **15 features only**
(15 principal components)

```
Training accuracy:  0.9998095238095238
Testing accuracy:  0.9563333333333334
```

Classification Report

	precision	recall	f1-score	support
0	0.98	0.99	0.98	2052
1	0.98	0.99	0.98	2330
2	0.95	0.96	0.96	2096
3	0.94	0.95	0.94	2222
4	0.96	0.94	0.95	2053
5	0.95	0.94	0.94	1833
6	0.97	0.98	0.98	2079
7	0.96	0.96	0.96	2191
8	0.95	0.94	0.94	2062
9	0.92	0.92	0.92	2082
accuracy			0.96	21000
macro avg	0.96	0.96	0.96	21000
weighted avg	0.96	0.96	0.96	21000



Nearly similar results with only 15 features needed!

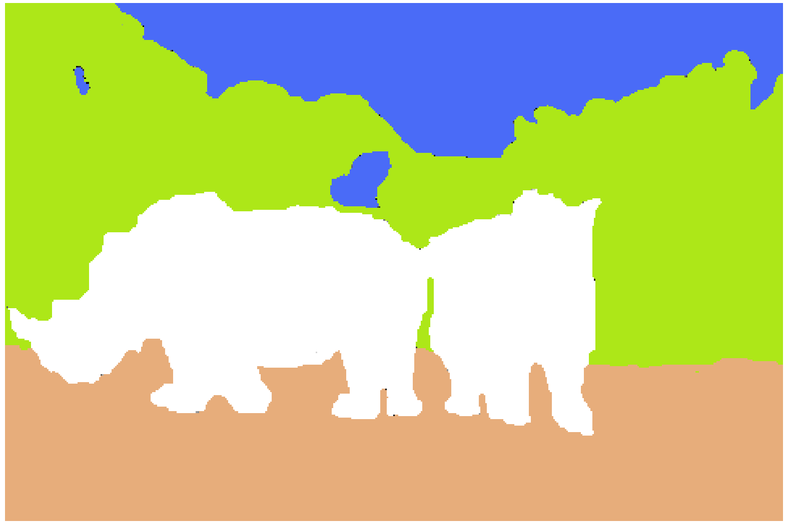
Today: Classification and Regression, Part 2b

1. Dimensionality reduction and SVC image classification
   - PCA—Principal Components Analysis
   - SVC on MINST database
2. Homework 2
   - Classification into three levels of importance
   - Use features that you measure via code

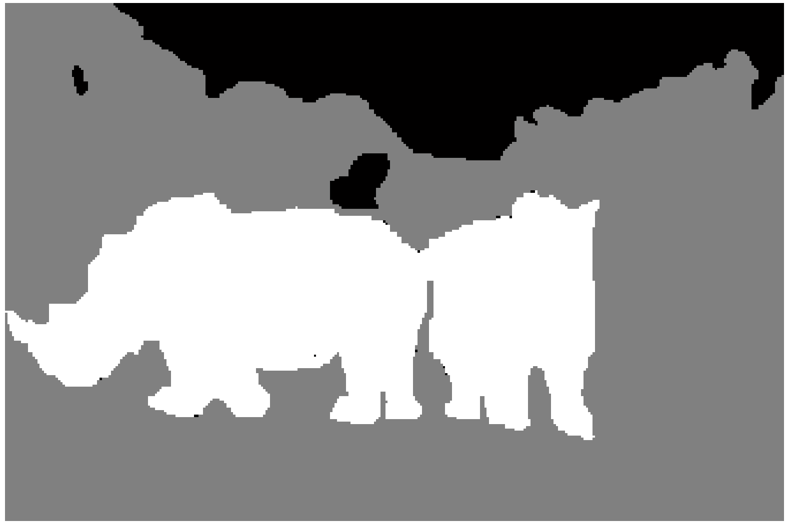
Assignment 2: Object Importance Classification

- **Importance maps** signify how visually important various object are in a photo.
- An example importance map is shown below (rightmost image).
- An importance map contains **three levels (classes)**:
  0=*unimportant*, 1=*somewhat important*, 2=*important*
- **Your goal**: Create a classification system to classify the importance of each region based on various features.

Original image



Segmentation map



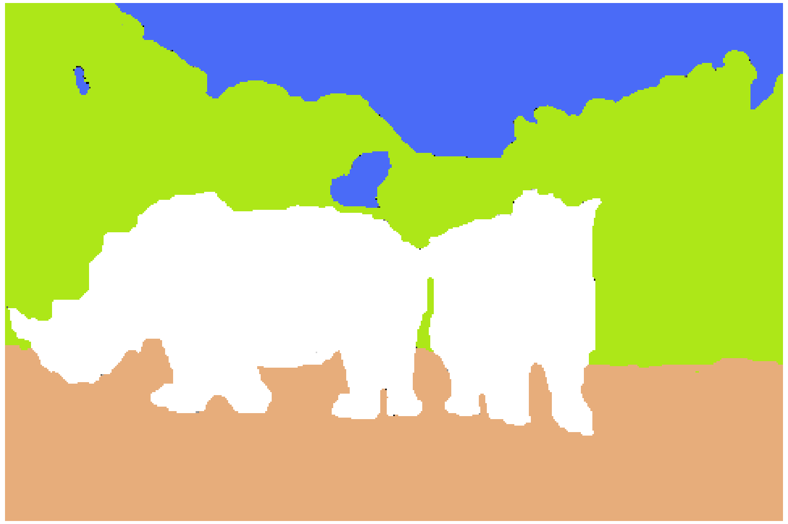
Importance map

Assignment 2: Object Importance Classification

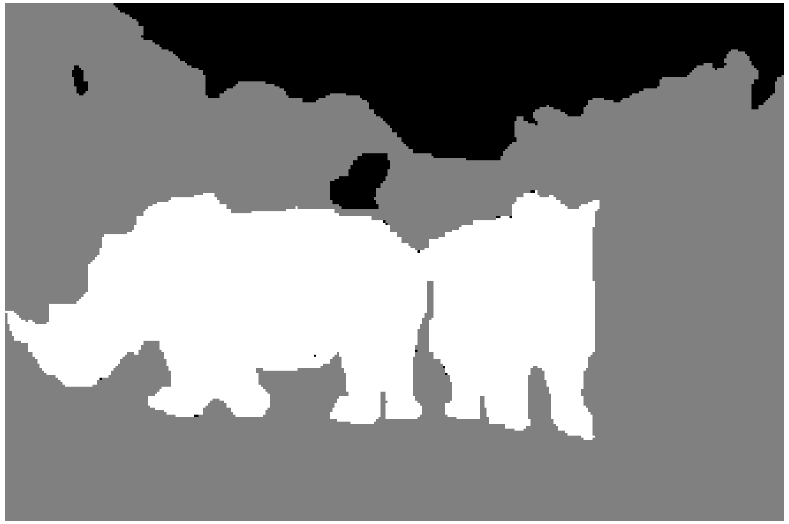
Specific steps (review the lecture video from 10/26 for a demo)**:**

1. Read the paper: *A Bayesian approach to predicting the perceived interest of objects.*
2. Download the importance map database.
3. Download the starter code.
4. Modify the starter code to measure more features that can help predict importance. You should measure at least the features mentioned in the paper, plus at least one unique feature of your own.
5. Use your features with various standard classifiers (Bayes, Decision Tree, SVM, etc.) to perform the classification.

Original image



Segmentation map



Importance map