

# Digits Recognizer

Group 2

G.X. Liu, D.W. Chen and K.I. Chung

June 21, 2017

## Introduction

Data Resource

Information

Image Examples

## Summary Statistics

Means

Principal Component Analysis

t-SNE

## Analysis

Dimensionality Reduction

One-versus-all Logistic Regression

K-Nearest Neighbor

Support Vector Machine

Neural Network

## Predict Instantly Online

<https://goo.gl/Ac1aXg>



# The Handwritten Digits

They are originally from the MNIST database.

- ▶ 60000 training data
- ▶ 10000 test data

Now, it is a competition on the Kaggle.

- ▶ 42000 training data
- ▶ 28000 test data

# Information

- ▶ Images contain  $28 \times 28$  pixels.
- ▶ There are 784 independent variables.
- ▶ The response labels the real values of images.

**Table:** Frequency of Digits

digit	0	1	2	3	4
freq.	4132	4684	4177	4351	4072
digit	5	6	7	8	9
freq.	3795	4137	4401	4063	4188

## Representation

The function,  $f$  transfers an observation with 784 variables into an  $28 \times 28$  image matrix.

$$f : \mathbb{R}^{784} \rightarrow M_{28 \times 28}$$

$$M = \begin{bmatrix} \text{pixel}_{000} & \text{pixel}_{001} & \cdots & \text{pixel}_{027} \\ \text{pixel}_{028} & \text{pixel}_{029} & \cdots & \text{pixel}_{055} \\ \vdots & \vdots & \ddots & \vdots \\ \text{pixel}_{756} & \text{pixel}_{757} & \cdots & \text{pixel}_{783} \end{bmatrix}_{28 \times 28}$$

## Images from the Training Data

0

1

2

3

4

5

6

7

8

9

## Images from the Test Data

6

9

3

1

8

9

2

4

3

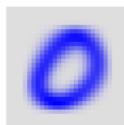
4

# Some Naked-eyed Unrecognizable Images

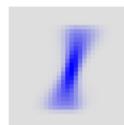


# The Mean Images of the Different Digits

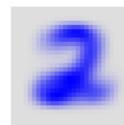
0



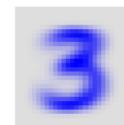
1



2



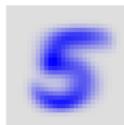
3



4



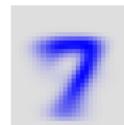
5



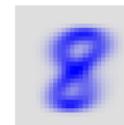
6



7



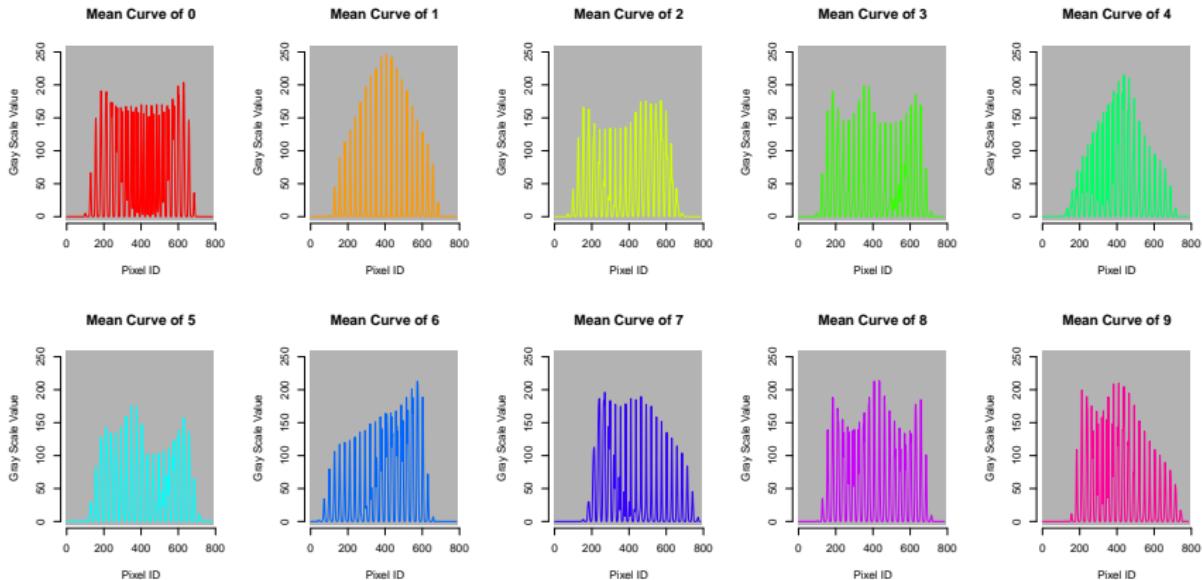
8



9



# The Mean Curves of the Different Digits



# The 20% Trimmed Mean Images of the Different Digits

0



1



2



3



4



5



6



7



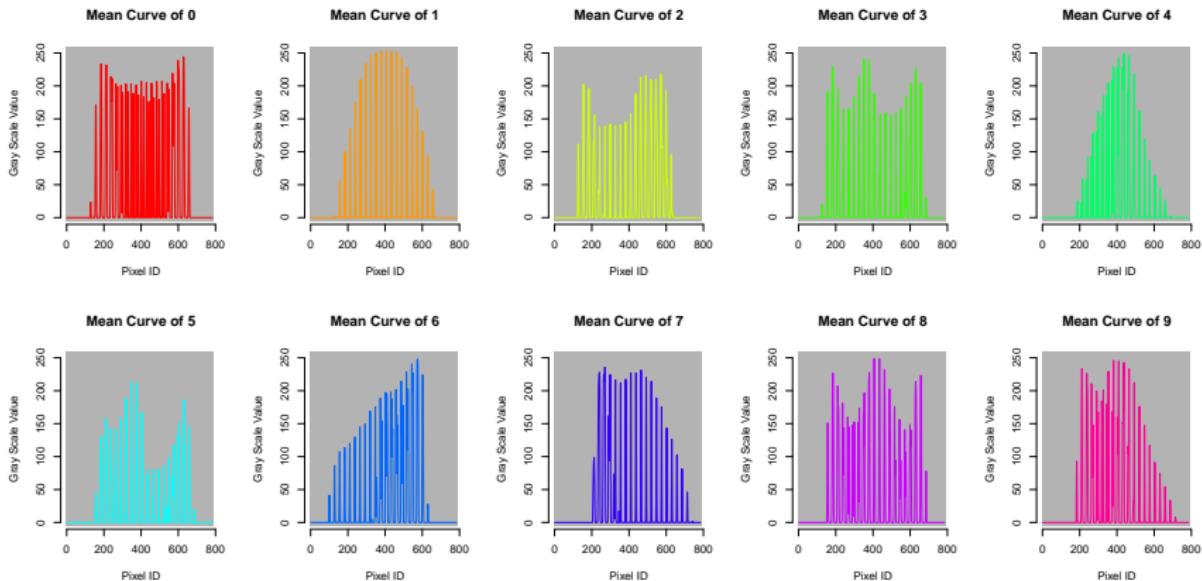
8



9



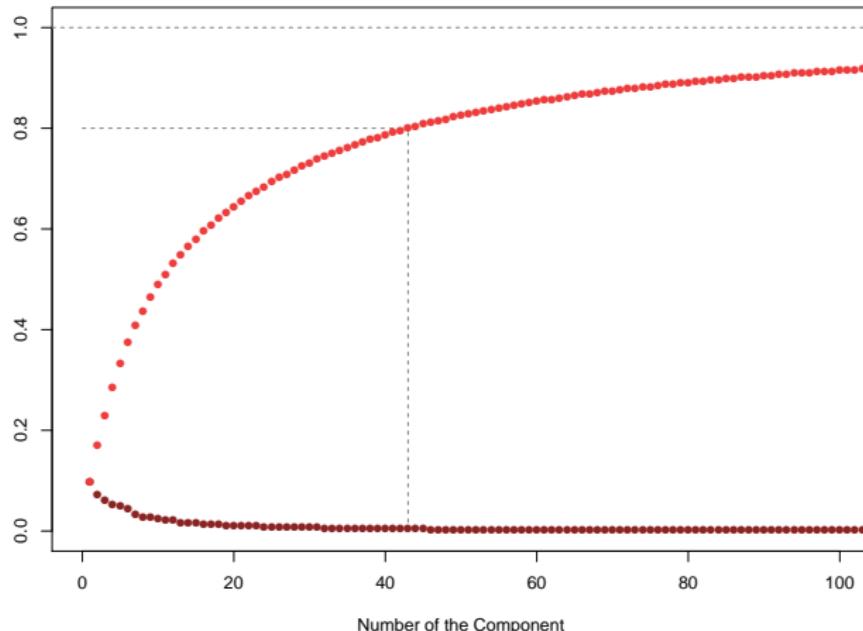
# The 20% Trimmed Mean Curves of the Different Digits



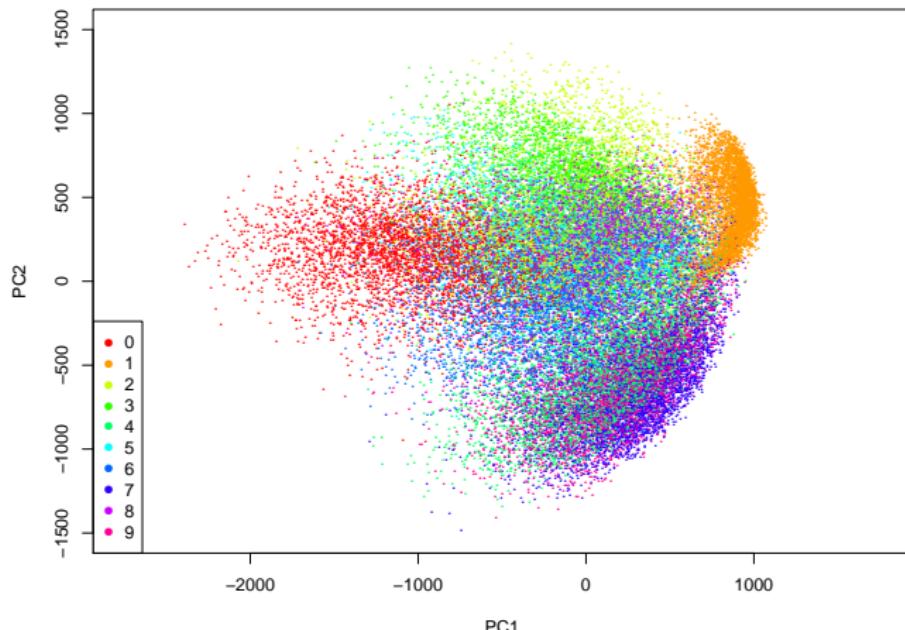
# Coefficients for the Principal Component

	PC1	PC2	...	PC43	...	PC784
$pixel_0$	0.000	0.000	...	0.000	...	0.000
$pixel_1$	0.000	0.000	...	0.000	...	-0.072
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$pixel_{462}$	0.075	-0.013	...	0.059	...	0.000
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$pixel_{783}$	0.000	0.000	...	0.000	...	0.000
Variance	5.149	3.781	...	0.221	...	0.000
Cumulative Ratio of Total Variance	0.098	0.169	...	0.800	...	1.000

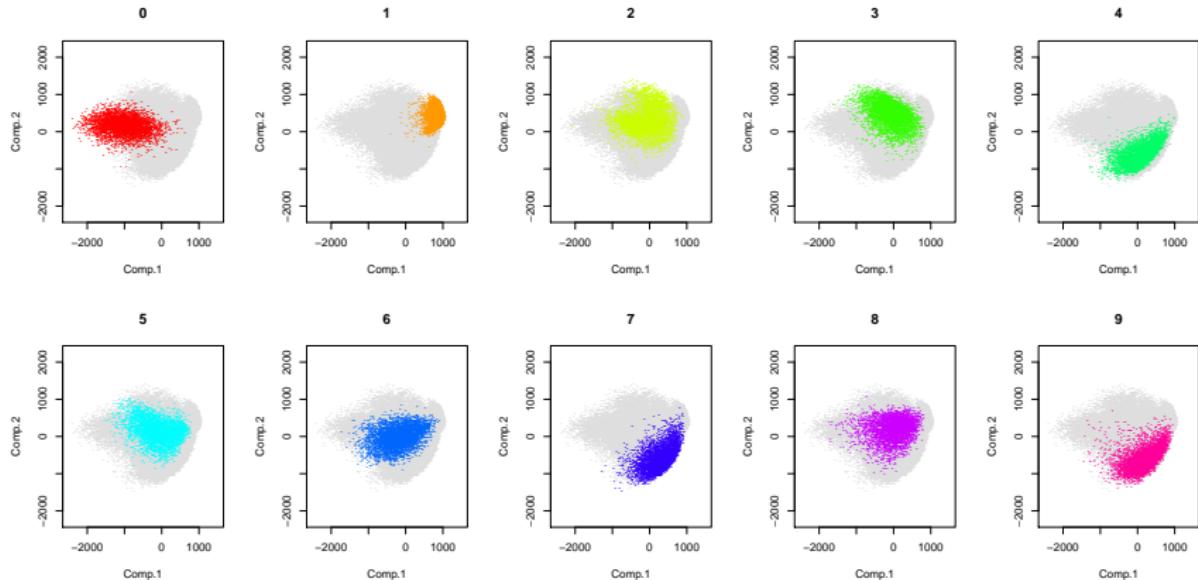
# Cumulative Ratio of Total Variance



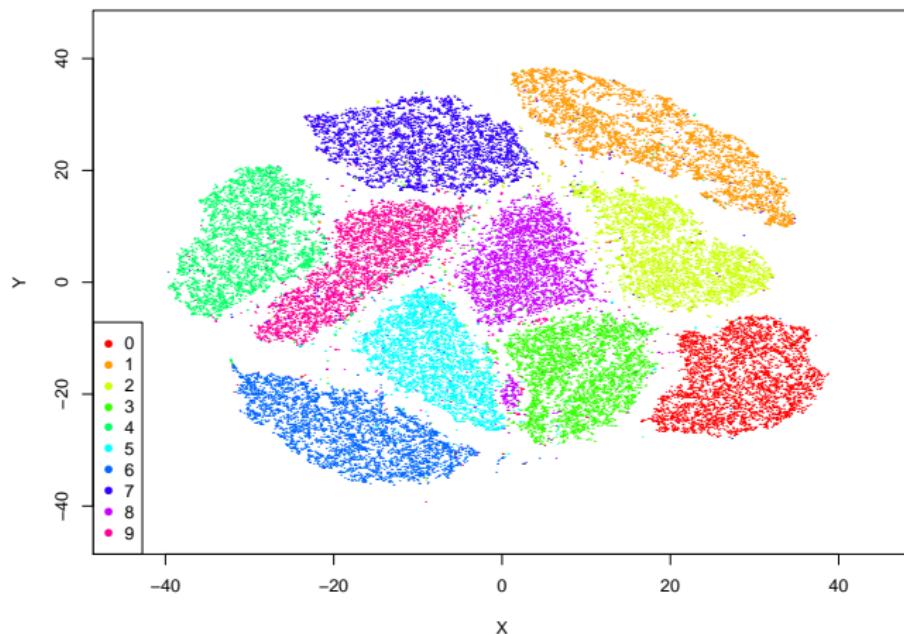
# Scatter Plot for the First Two Components



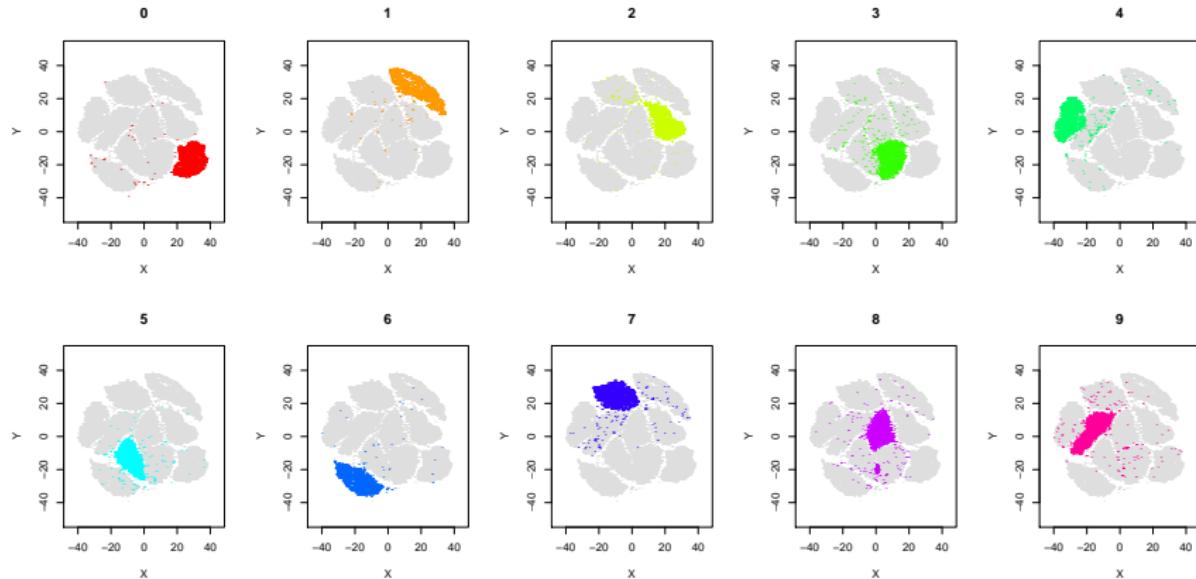
# Separated Scatter Plot for the First Two Components



## Two-dimensional t-SNE with 600-time Iterations

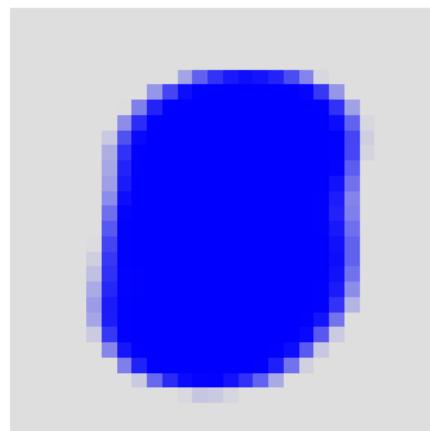


# Separated Scatter Plot for the Two-dimensional t-SNE



# Dimensionality Reduction

- ▶ Calculate the 90<sup>th</sup> percentile
- ▶ Preserve those pixels with non-zero 90<sup>th</sup> percentile
- ▶ 43.75% of pixels remain



# One-versus-all Logistic Regression

	Prediction									
	0	1	2	3	4	5	6	7	8	9
0	944	7	4	9	7	1	2	1	0	0
1	536	565	8	5	4	3	4	1	2	0
2	331	291	274	13	11	10	15	10	3	1
3	251	238	244	227	20	8	8	7	2	1
4	206	173	174	196	198	14	4	5	4	5
5	154	146	166	155	141	136	10	3	4	2
6	185	135	140	116	131	142	122	0	1	0
7	148	149	131	133	141	133	127	127	5	3
8	130	120	127	105	108	109	78	111	77	1
9	108	117	114	114	106	96	84	100	84	78

Table: Confusion Matrix (Accuracy = 27.48%)

# K-Nearest Neighbor (KNN)

Reality	Prediction									
	0	1	2	3	4	5	6	7	8	9
0	963	0	2	0	2	1	3	0	0	3
1	0	1141	5	1	0	1	1	1	2	0
2	6	7	979	4	1	0	1	15	1	2
3	1	0	5	957	0	17	2	6	5	3
4	0	8	0	0	957	0	6	2	0	32
5	4	1	0	18	1	827	11	3	4	4
6	5	0	1	0	3	4	957	0	0	0
7	0	8	5	0	3	0	0	1025	0	7
8	3	13	5	18	3	10	5	3	885	13
9	3	3	2	6	15	2	1	19	2	955

Table: Confusion Matrix (Accuracy = 96.46%)

# Support Vector Machine (SVM)

Reality	Prediction									
	0	1	2	3	4	5	6	7	8	9
0	961	0	5	1	0	1	4	0	0	2
1	0	1137	3	3	4	1	1	2	1	0
2	2	1	995	1	2	1	0	5	9	0
3	0	0	1	974	1	7	0	3	8	2
4	0	3	0	0	987	0	3	0	1	11
5	1	0	0	11	1	848	6	0	3	3
6	4	0	1	0	3	4	957	0	1	0
7	0	4	3	0	1	0	0	1034	1	5
8	2	2	2	2	2	4	0	1	942	1
9	2	2	4	6	8	1	1	4	4	976

Table: Confusion Matrix (Accuracy = 98.11%)

# Multi-layer Perceptron (784-1000-1000-10 with ReLU)

	Prediction									
	0	1	2	3	4	5	6	7	8	9
Reality	974	1	0	2	0	0	1	1	1	0
0	974	1	0	2	0	0	1	1	1	0
1	0	1128	2	0	0	0	3	0	2	0
2	1	1	1020	4	1	0	1	3	1	0
3	0	0	3	1000	0	2	0	3	0	2
4	1	0	2	1	966	0	4	1	1	6
5	2	0	0	11	1	873	4	0	0	1
6	2	3	0	1	3	5	944	0	0	0
7	0	6	11	2	0	1	0	1000	1	7
8	0	0	3	13	2	5	4	3	938	6
9	1	3	0	7	8	2	2	4	0	982

**Table:** Confusion Matrix (Accuracy = 98.25%)

# Convolution Neural Network

	Prediction									
	0	1	2	3	4	5	6	7	8	9
Reality	976	1	0	0	0	0	1	1	1	0
0	976	1	0	0	0	0	1	1	1	0
1	0	1134	0	1	0	0	0	0	0	0
2	1	2	1027	0	0	0	0	2	0	0
3	0	0	0	1005	0	3	0	0	2	0
4	0	0	0	0	980	0	0	0	0	2
5	0	0	0	5	0	886	1	0	0	0
6	1	3	0	0	2	4	946	0	2	0
7	0	3	5	1	0	0	0	1015	1	3
8	1	0	2	3	1	2	0	1	961	3
9	0	2	0	1	5	3	0	1	0	997

Table: Confusion Matrix (Accuracy = 99.27%)

# Predict Instantly Online

- ▶ <https://goo.gl/Ac1aXg>
- ▶ Train the MLP with Keras on Python
- ▶ Export the weight matrices and bias vectors as JavaScript file
- ▶ Insert the matrices into JavaScript
- ▶ **Mobilized** the model

## Q & A

Thank you for listening