Image Classification and Nearest Neighbor

Chia-Po Wei

Department of Electrical Engineering
National Sun Yat-sen University

Outline

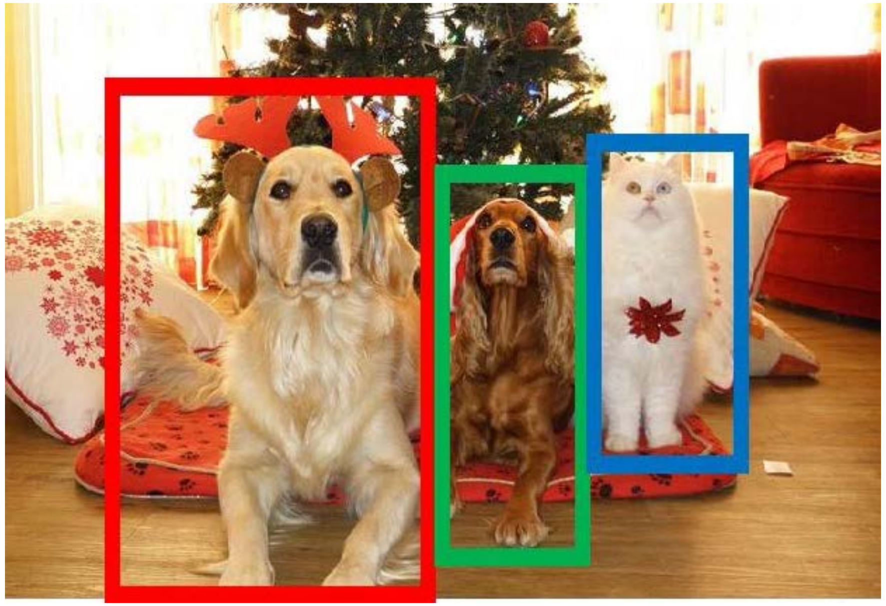
- Image Classification
  - Challenges
  - Machine Learning vs. Classical Programming
- k-Nearest Neighbors (k-NN)
  - Hyperparameters
  - Cross-Validation

Tasks in Computer Vision

Classification

CAT

Object Detection



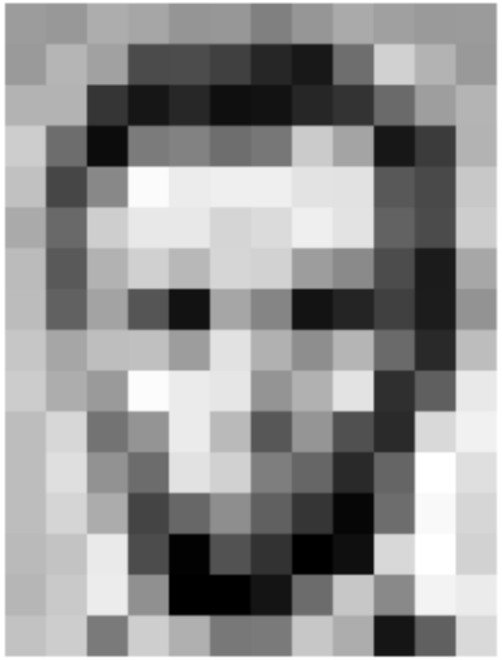
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Image Classification

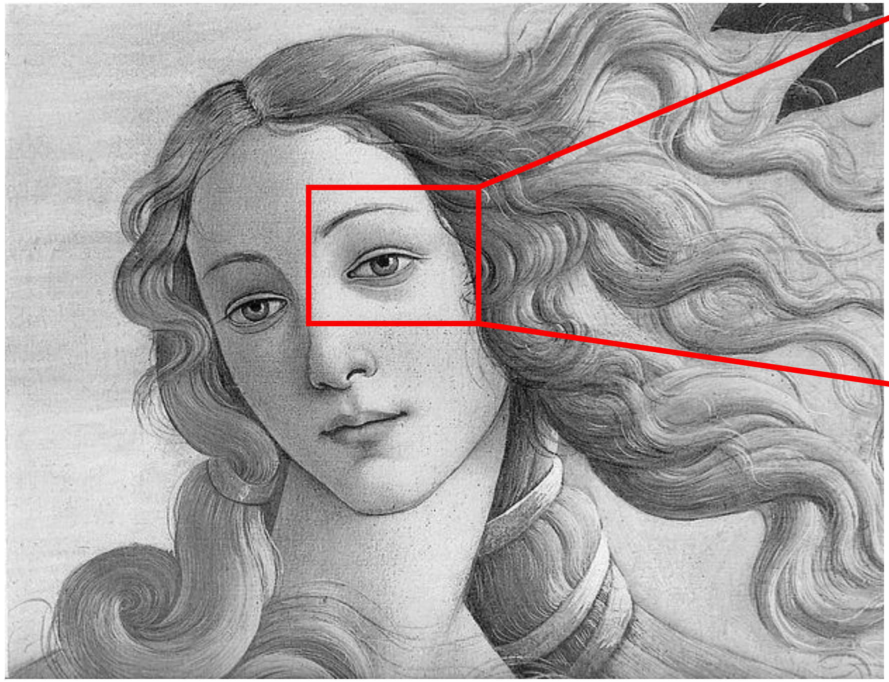
- The task of image classification is to predict the class label of a test image.
- Example:



Test image

prob.	label
0.7	Lincoln
0.1	Washington
0.1	Jefferson
0.1	Obama

Images as Matrices

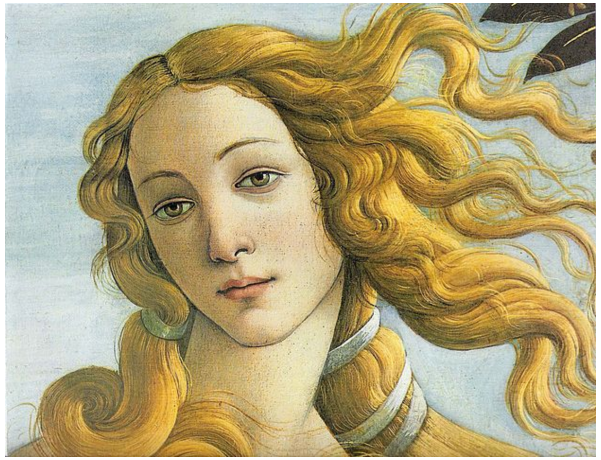


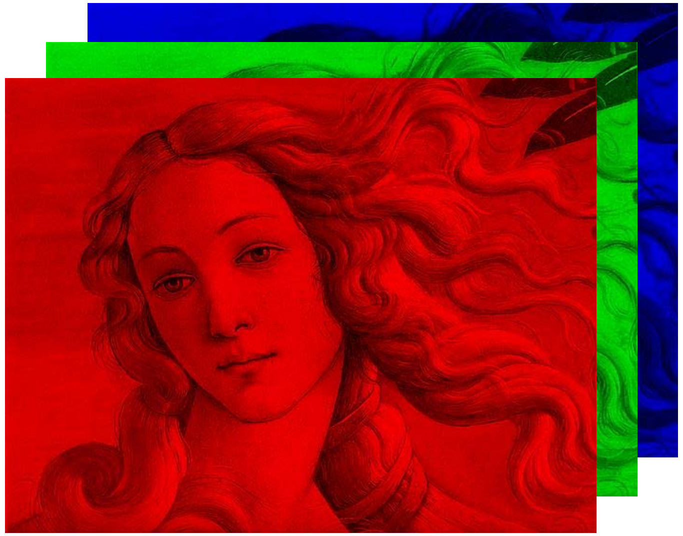
179	167	150	165	156	145	145	183	184
175	172	157	166	169	180	168	168	165
126	164	170	150	121	135	127	112	132
151	130	137	188	196	186	170	178	165
170	166	161	170	155	164	167	173	187
150	154	155	157	161	173	147	164	165
170	149	135	141	151	173	157	170	170
165	174	186	186	166	176	166	168	173
149	165	174	169	152	188	200	191	196
191	175	155	164	164	189	181	153	161

- An image is just a matrix of pixels
- E.g. The left image is 480x600
- The value of each pixel in an image is between 0~255

Color Images

```
img = cv2.imread('birth_of_venus.jpg')
print(img.shape) #(480,600,3)
```



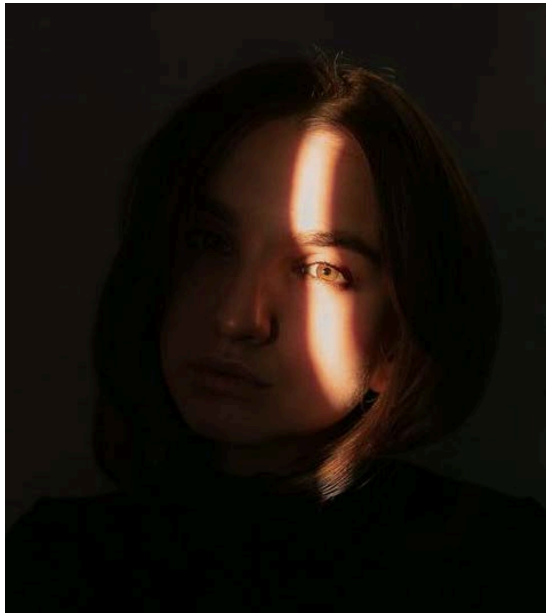


- A color image consists of three RGB channels
- The sequence of the channels can be RGB or BGR, depending on the library you use for reading the image.

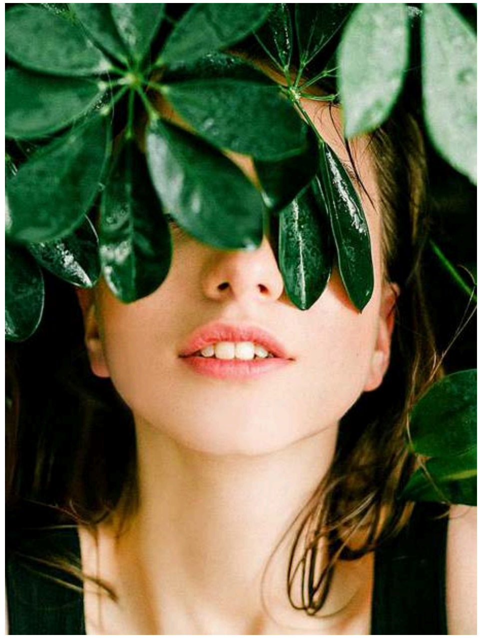
Challenges: Viewpoint

Challenges: Illumination



Challenges: Occlusion

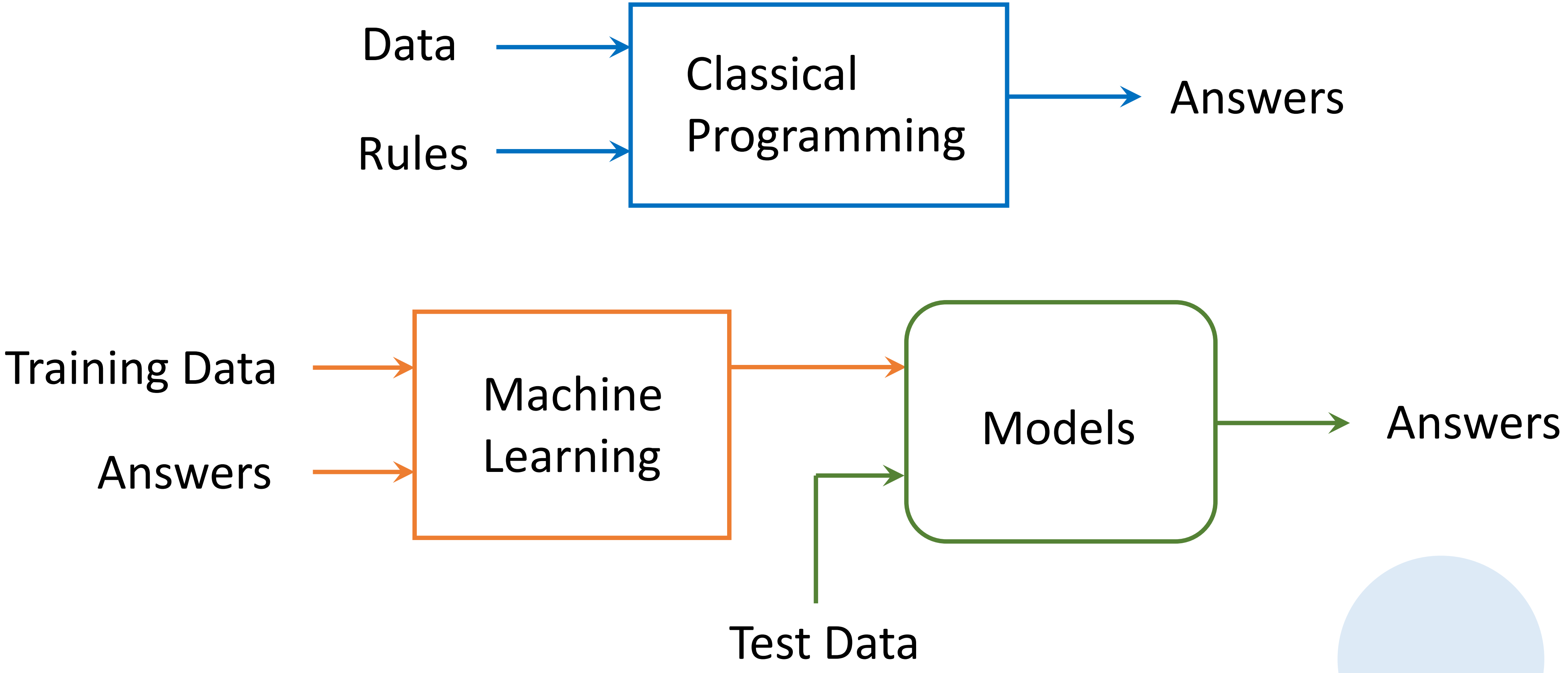


Challenges: Deformation

Challenges: Intra-class Variation

Challenges: Background Clutter

Machine Learning vs. Classical Programming



Machine Learning: Data-Driven Approach to Classification

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on test images

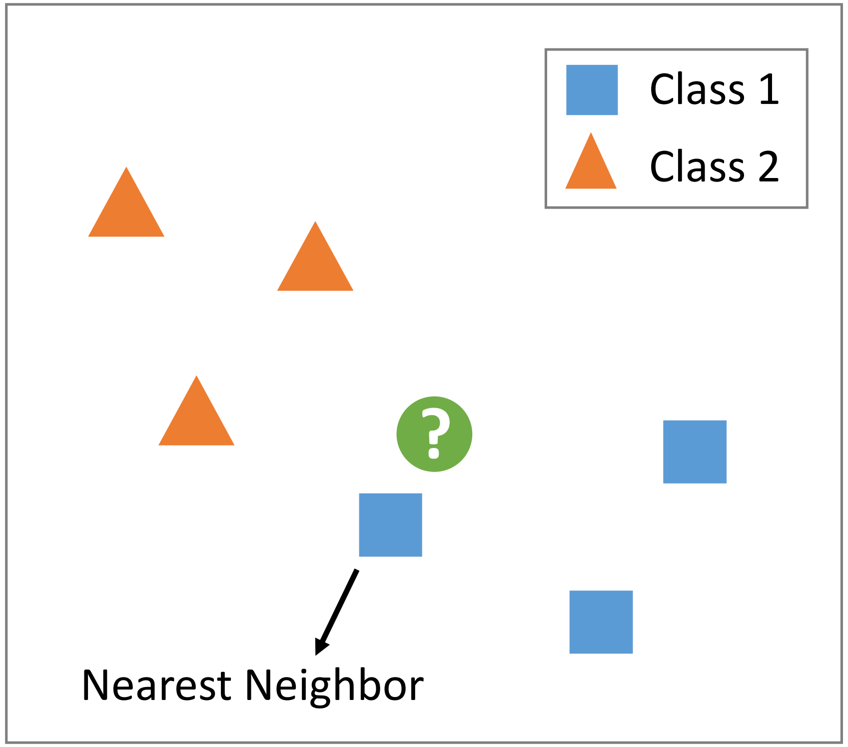
```
def train(images, labels):
  # Use machine learning to train a model for classification
  return classification_model
```

```
def predict(classification_model, test_images):
  # Use the classification model to predict
  # the label of each test image
  return test_labels
```

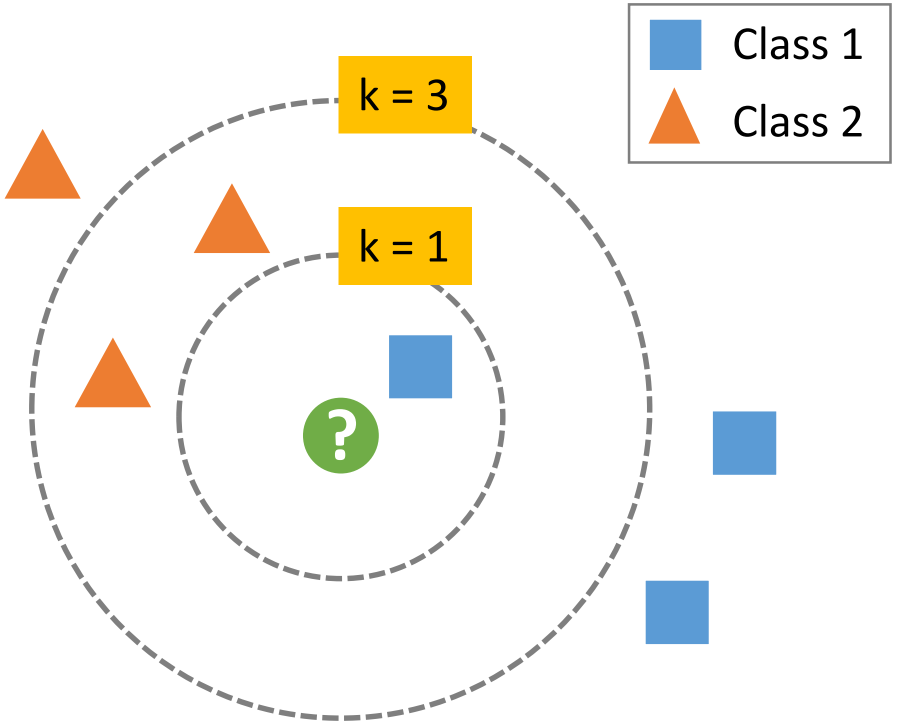
Nearest Neighbor (NN)

- Given a set of training images with class labels
- For a test image

1. Calculate the distance to each training image, and store the distance in a list
2. Sort the list of distances in ascending order
3. Assign the class label of the first point in the list to the test image

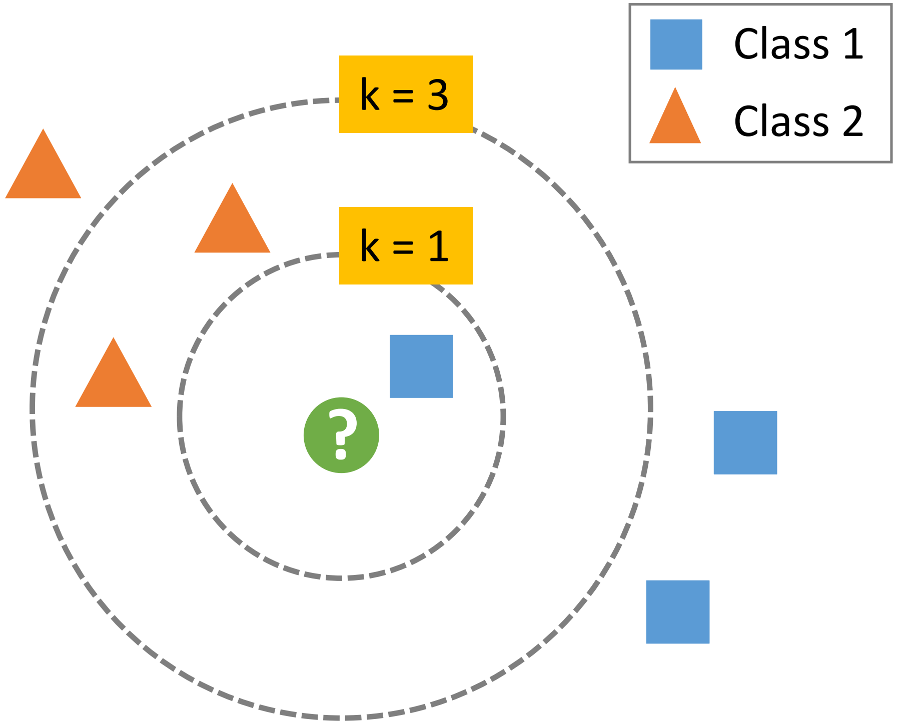


k-Nearest Neighbors (k-NN)



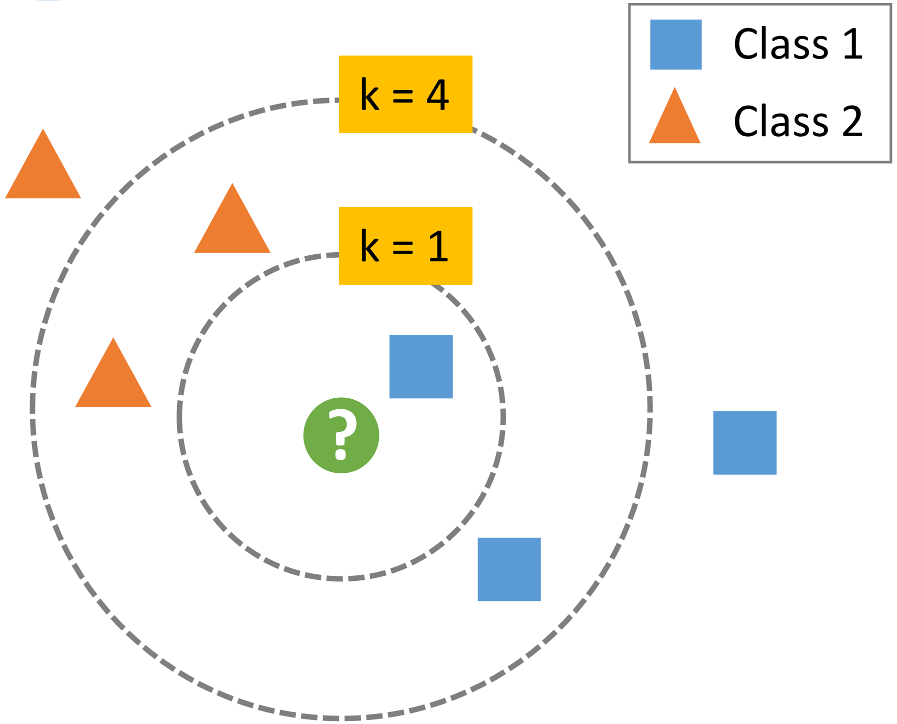
- k-NN takes **majority vote** from k closest neighbors instead of copying label from the nearest neighbor.
- In practice, k-NN performs better than NN.
- But how to determine the value of k?

k-NN Illustration



- The test sample ● should be classified either to Class 1 ■ or to Class 2 ▲
- If k = 1 (inside circle) it is assigned to Class 1 ■ because there is only 1 square.
- If k = 3 (outside circle) it is assigned to Class 2 ▲ because there are 2 triangles and only 1 square.

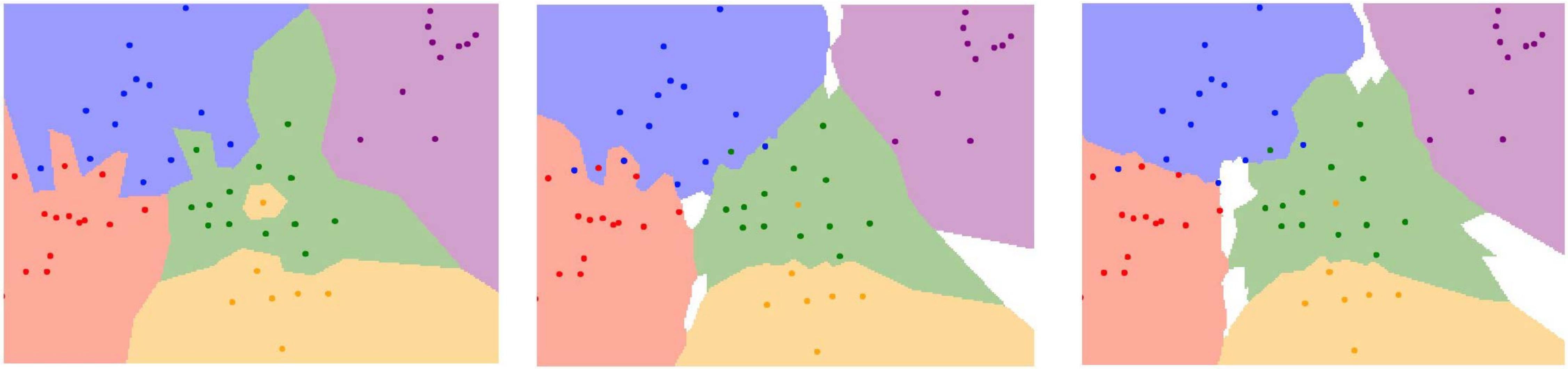
Quiz 1



1. What is the label of the green target if using 4-nn as the classifier?
2. What can we do if the class votes are tied?

Choices of k

k = 1 k = 3 k = 5

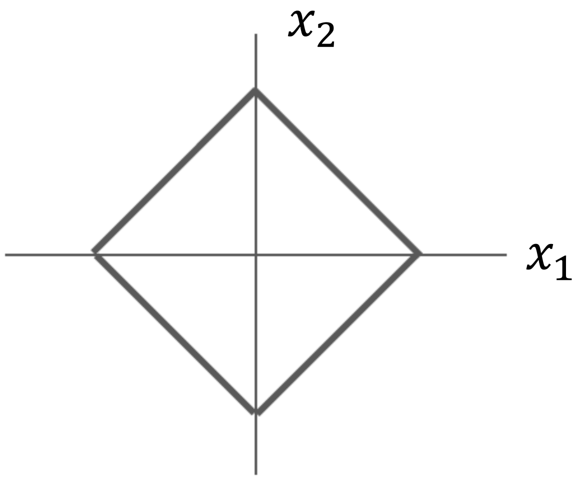


- For NN classifiers, outlier points create small islands of likely incorrect predictions
- 5-NN classifiers smooth over these irregularities, likely leading to better **generalization** on the test data (not shown).

Common Distance Metric

L1 (Manhattan) distance

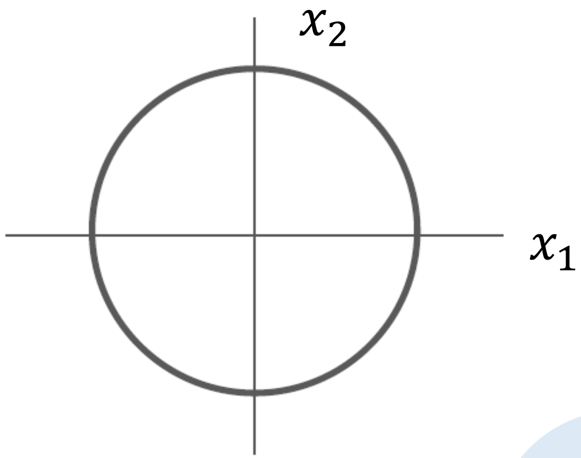
$$\text{dist}(\mathbf{I}_1, \mathbf{I}_2) = \sum_k \left| \mathbf{I}_1^k - \mathbf{I}_2^k \right|$$



$$\|\mathbf{x}\|_1 = |x_1| + |x_2| = 1$$

L2 (Euclidean) distance

$$\text{dist}(\mathbf{I}_1, \mathbf{I}_2) = \sqrt{\sum_k \left(\mathbf{I}_1^k - \mathbf{I}_2^k \right)^2}$$



$$\|\mathbf{x}\|_2 = x_1^2 + x_2^2 = 1$$

L1 Distance Metric

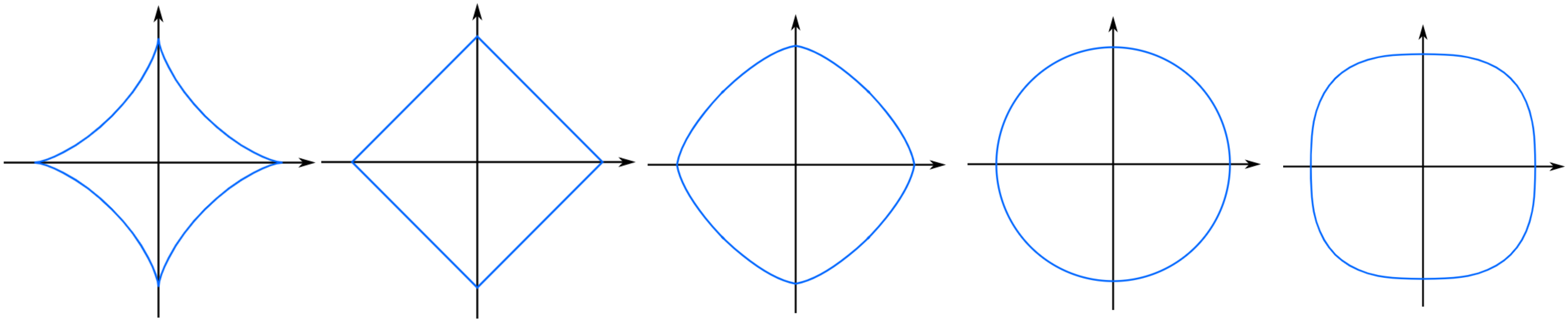
$$\mathrm{dist}(\mathbf{I}_1, \mathbf{I}_2) = \sum_k \left| \mathbf{I}_1^k - \mathbf{I}_2^k \right|$$

Test image, Training image, Absolute difference

$$\left| \begin{array}{|c|c|c|c|} \hline 23 & 155 & 166 & 45 \\ \hline 34 & 203 & 200 & 63 \\ \hline 66 & 255 & 195 & 98 \\ \hline 77 & 143 & 150 & 88 \\ \hline \end{array} - \begin{array}{|c|c|c|c|} \hline 45 & 90 & 78 & 42 \\ \hline 87 & 122 & 144 & 94 \\ \hline 76 & 156 & 167 & 83 \\ \hline 34 & 234 & 255 & 72 \\ \hline \end{array} \right| = \begin{array}{|c|c|c|c|} \hline 22 & 65 & 88 & 3 \\ \hline 53 & 81 & 56 & 31 \\ \hline 10 & 99 & 28 & 15 \\ \hline 43 & 91 & 105 & 16 \\ \hline \end{array} \xrightarrow{\text{sum}} 806$$

Minkowski Distance

$$\mathrm{dist}(\mathbf{I}_1, \mathbf{I}_2) = \left(\sum_k \left(\mathbf{I}_1^k - \mathbf{I}_2^k\right)^p\right)^{1/p}$$



$p = 2^{-0.5}$
$= 0.707$

$p = 2^0$
$= 1$

$p = 2^{0.5}$
$= 1.414$

$p = 2^1$
$= 2$

$p = 2^{1.5}$
$= 2.828$

Hyperparameters

- For k-NN
  - What is the best value of k to use?
  - What is the best distance metric to use? (L1, L2, or dot products)
- These choices are called hyperparameters.
  - Come up in the design of Machine Learning algorithms
  - Choices regarding the algorithm that we set rather than learn

Choosing Hyperparameters

Method 1: Split data into train and test. Choose hyperparameters that work best on test.

The problem of Method 1 is that we have no idea how the algorithm will perform on new data.

Method 2: Split data into train, val, and test. Choose hyperparameters on val and evaluate on test.

Cross-Validation for Hyperparameter Tuning

Method 3 (**Cross-Validation**): Split training data into folds. Try one fold as val and the remaining folds as train. Average the results of each fold as val to choose hyperparameters.

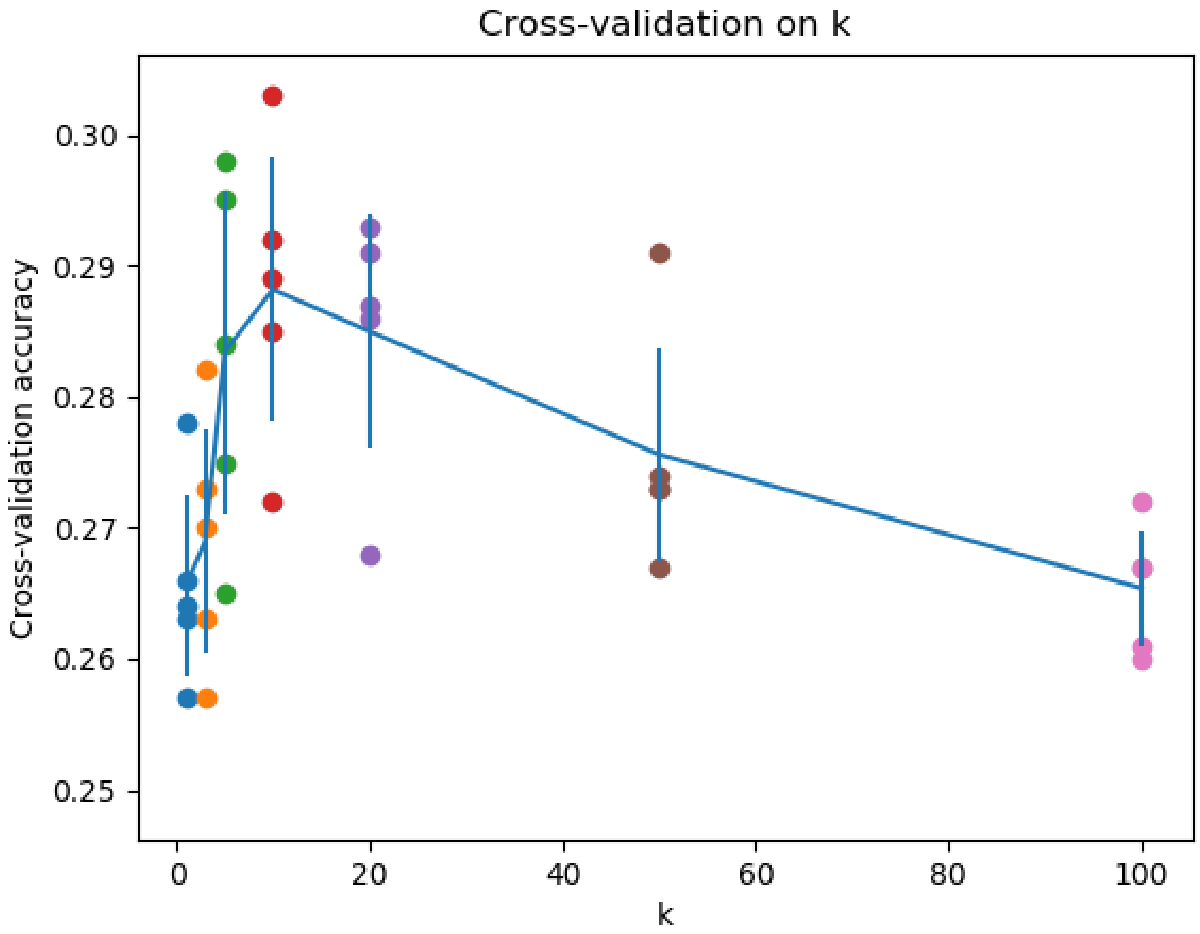
- Below is an example of 3-fold cross-validation. Blue folds are for training, while green folds are for validation.

fold 1	fold 2	fold 3	test

fold 1	fold 2	fold 3	test

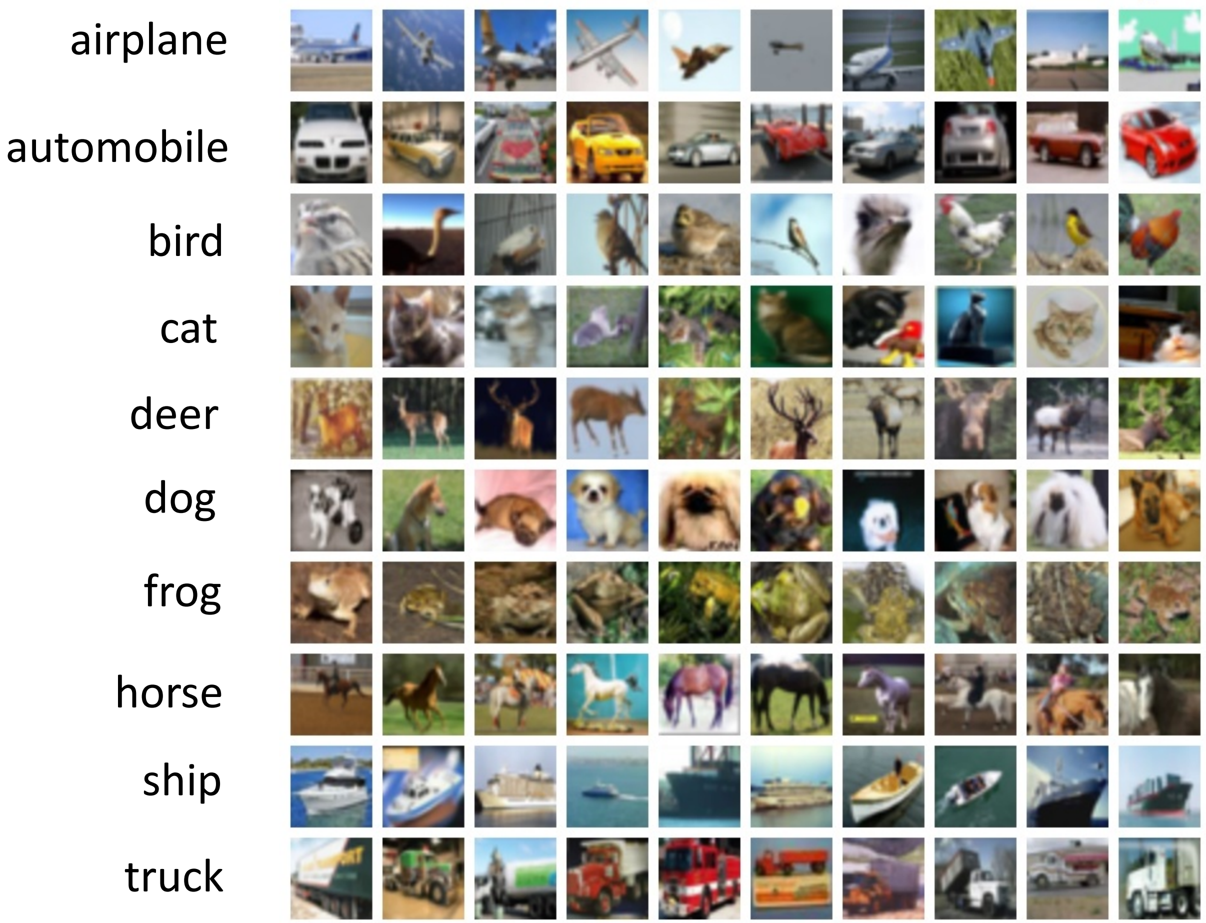
fold 1	fold 2	fold 3	test

Results of 5-fold Cross-Validation



- Each point is a single outcome.
- Each k has 5 points because of 5-fold cross-validation.
- The line goes through the mean, while bars indicated the standard deviation.

CIFAR-10 Dataset



- 60,000 color images in 10 classes
- Each class has 6,000 images
- Each color image is 32x32
- 50,000 images for training
- 10,000 images for testing

- No overlap between automobiles and trucks

Class Labels

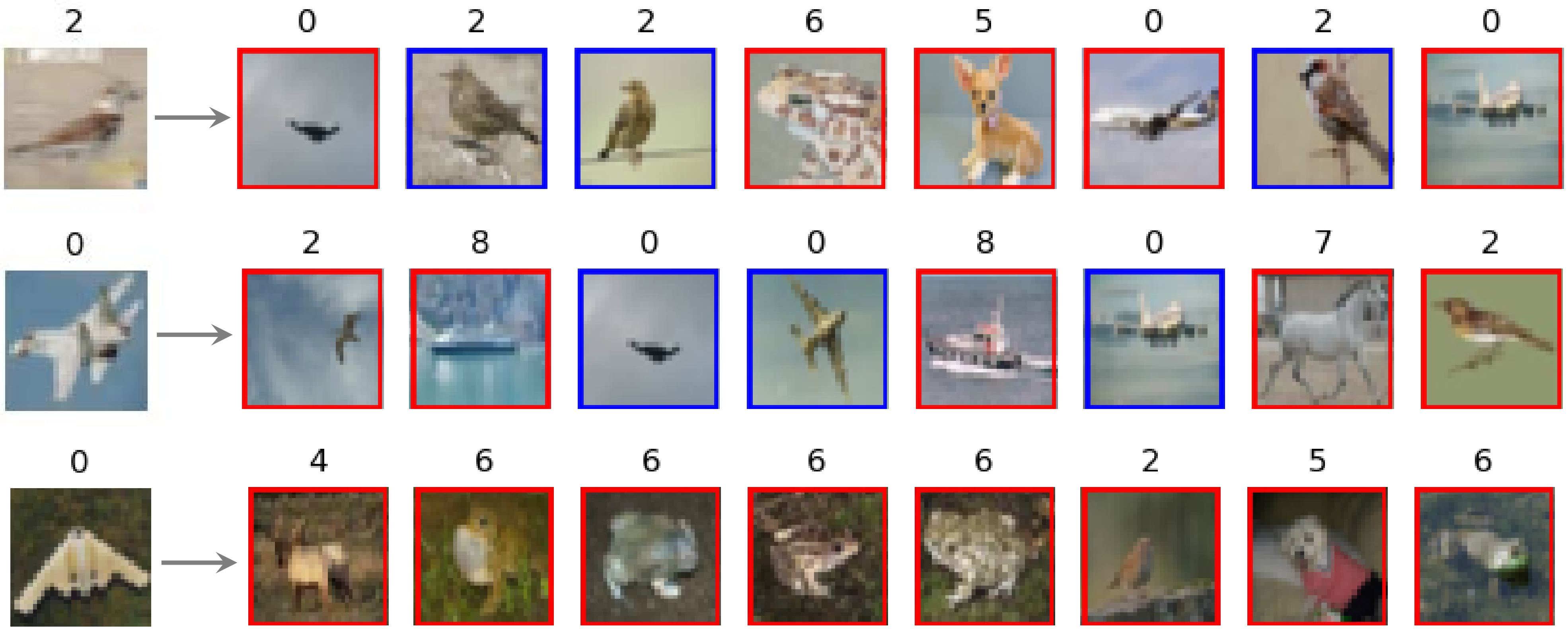
Original (k-NN)

class	label
airplane	0
automobile	1
bird	2
cat	3
deer	4
dog	5
frog	6
horse	7
ship	8
truck	9

One-Hot Encoding (softmax)

class	label
airplane	[1 0 0 0 0 0 0 0 0 0]
automobile	[0 1 0 0 0 0 0 0 0 0]
bird	[0 0 1 0 0 0 0 0 0 0]
cat	[0 0 0 1 0 0 0 0 0 0]
deer	[0 0 0 0 1 0 0 0 0 0]
dog	[0 0 0 0 0 1 0 0 0 0]
frog	[0 0 0 0 0 0 1 0 0 0]
horse	[0 0 0 0 0 0 0 1 0 0]
ship	[0 0 0 0 0 0 0 0 1 0]
truck	[0 0 0 0 0 0 0 0 0 1]

Classification Examples of kNN



- Red boxes indicate incorrect results, while blue boxes indicate correct results.
- The number above each image represents the class label of the image.

Quiz 2

1. What is the class label of the following test image if using NN as the classifier?

Test image

Frontal 1 (f1)

Label 1

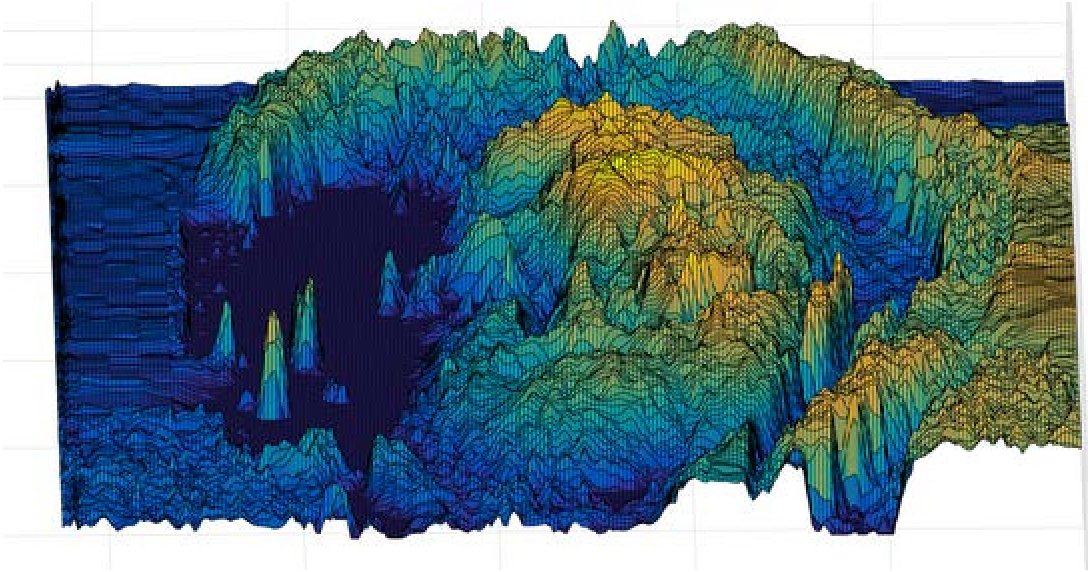
Profile 1 (p1)

Label 2

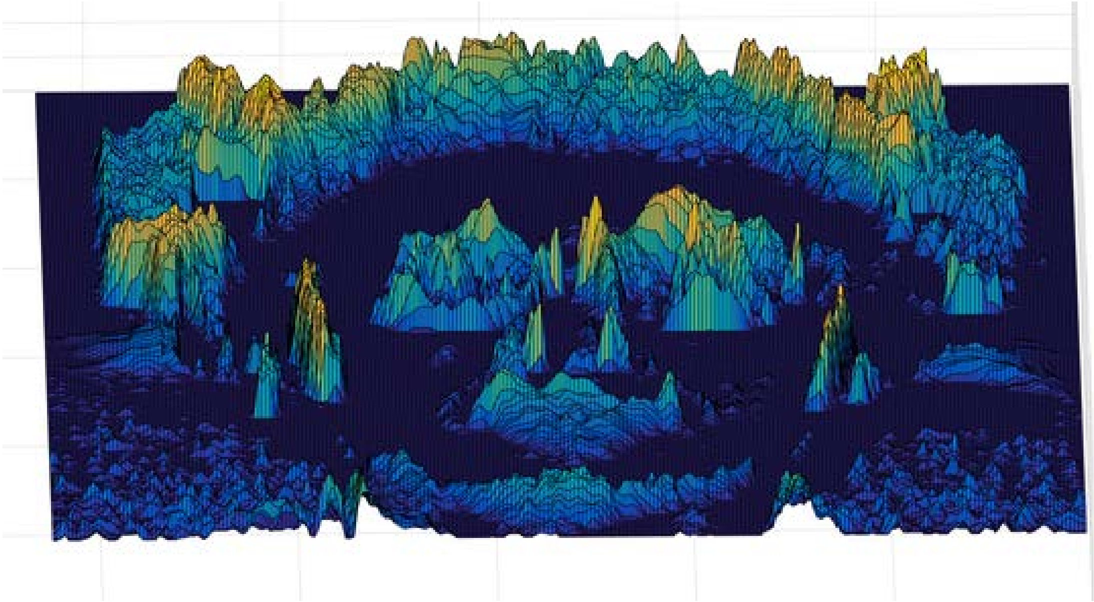
Frontal 2 (f2)

Quiz 2 (cont.)

abs(f1 – p1)



abs(f1 - f2)



- sum(abs(f1-p1)) / N = 57.7
- sum(abs(f1-f2)) / N = 11.1
- N is the number of pixels

Pros and Cons of kNN

- Pros
  - Simple to implement and understand
  - No need to train a model
- Cons
  - Need to store all the training data
  - Heavy computational cost at test time as the number of training images increases
  - Using pixel differences to compare images is inadequate