



# Computer Vision 1

<https://tinyurl.com/595b9yer>

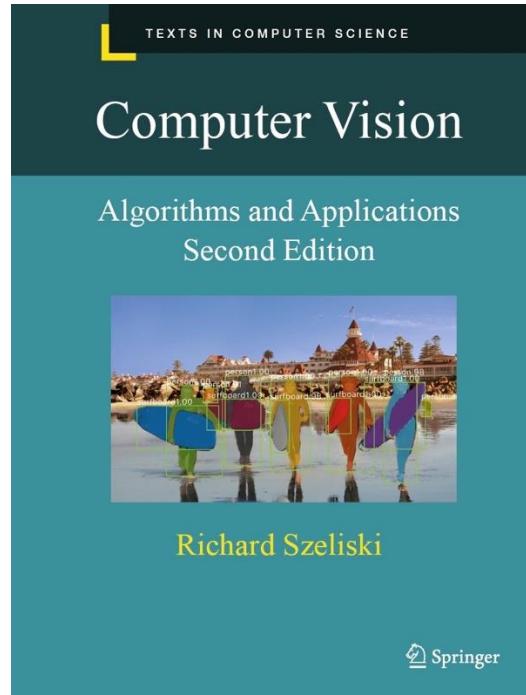
HC4b

# Image Classification

Dr. Martin Oswald, Dr. Dimitris Tzionas, Dr. Arun Mukundan,  
[m.r.oswald, d.tzionas, a.mukundan]@uva.nl

# Textbook

- Chapter 6 (until 6.2.2)



# Image Understanding

AN IMAGE IS WORTH A THOUSAND WORDS



# Image Understanding

## Classification



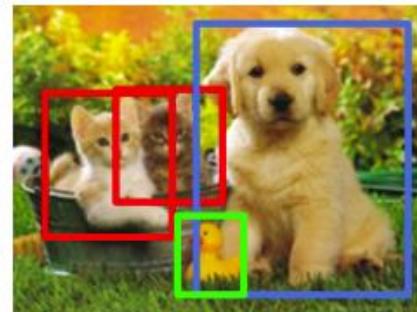
CAT

## Classification + Localization



CAT

## Object Detection



CAT, DOG, DUCK

## Instance Segmentation



CAT, DOG, DUCK

- Object Classification: classify the main object in an image if the image has one main object.
- Visual Image Retrieval: retrieve images from the library through a query image.
- Object Detection: identify whether there is an object in an image and recognize the object.
- Image Segmentation: segment an image into regions according to the pixels the object belongs to.

## Today: Image Classification!

# Today's Agenda

---

## Image Classification

- Problem Setting and Challenges
- Bag of words
- Nearest Neighbors
- Hyperparameters and Training
- Linear Classifier
- SVMs
- Classification Metrics and Performance Evaluation

# Today's Agenda

---

## Image Classification

- **Problem Setting and Challenges**
- Bag of words
- Nearest Neighbors
- Hyperparameters and Training
- Linear Classifier
- SVMs
- Classification Metrics and Performance Evaluation

# Challenges

---

**Computer Vision is hard!**

# Challenges: Similar Appearance



# Challenges: Similar Appearance



# Challenges: Similar Appearance



# Challenges: Similar Appearance



# Challenges: Similar Appearance



# Challenges: Similar Appearance



# Challenges

# Computer Vision ~~is~~ hard!

used to be ?

Ask ChatGPT:

Please tell me what is in each cell in a tabular format aligned with this picture grid.

Certainly! Here's a breakdown of the content of each cell in the format of a table:

Row	Column 1	Column 2	Column 3	Column 4
1	Muffin with blueberries	Chihuahua (side profile)	Blueberry Muffin	Chihuahua (profile looking up)
2	Chihuahuas (frontal face)	Cookie with chocolate chips	Chihuahua (face close-up)	Blueberry Muffin with sugar on top
3	Blueberry Muffin with a smiley face	Chihuahua (frontal face close-up)	Blueberry Muffin with blueberries on top	Chihuahua (slightly showing teeth)
4	Chihuahua (puppy face close-up)	Group of Cookies with chocolate chips	Group of Chihuahua puppies	Muffin with blueberries

# Image Classification

A core computer vision task.

**Input:** Image



[Image source: [Flickr](#)]

**Output:** One of a fixed set  
of categories (vocabulary).

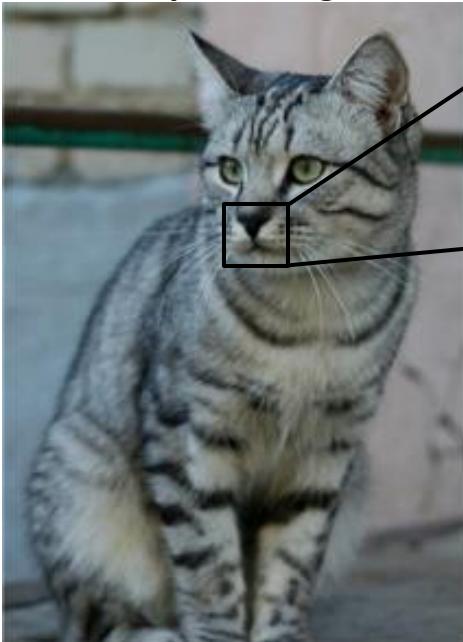


cat  
bird  
deer  
dog  
truck

# Image Classification

Difficulty: Semantic gap

Input: Image



[Image source: [Flickr](#)]

[[105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87] [ 91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85] [ 76 85 90 105 128 105 87 96 95 99 115 112 106 103 99 85] [ 99 81 81 93 120 137 127 104 95 98 102 99 96 93 101 94] [106 91 61 64 69 91 88 85 101 107 109 98 75 84 96 95] [114 108 85 55 55 69 64 54 64 87 112 129 98 74 84 91] [131 137 147 103 65 81 80 65 52 54 74 84 102 93 85 82] [120 137 144 140 95 86 70 62 65 63 63 68 73 86 101] [125 133 148 137 119 121 117 94 65 79 80 65 54 64 72 98] [127 125 131 147 133 127 126 131 111 96 89 75 61 64 72 84] [125 114 109 123 150 145 131 118 113 109 100 92 74 65 72 78] [ 89 93 90 97 100 147 131 118 113 114 113 109 106 95 77 80] [ 63 77 86 81 77 79 182 123 117 115 117 125 125 130 115 87] [ 62 65 82 89 78 71 80 101 124 128 119 101 107 114 131 139] [ 63 65 75 80 89 71 62 81 120 130 135 105 81 98 118 118] [ 87 65 71 87 106 95 69 45 76 138 126 107 92 94 105 112] [110 97 82 86 117 123 116 66 41 51 95 93 89 95 102 107] [164 146 112 80 82 128 124 104 76 48 45 66 88 101 102 109] [157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94] [130 120 134 161 139 100 109 110 121 134 114 87 65 53 69 86] [128 112 96 117 158 144 120 115 104 107 102 93 87 81 72 79] [123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99] [122 121 182 80 82 86 94 117 145 140 153 102 58 78 92 107] [122 164 148 103 71 56 78 83 93 103 119 109 102 61 69 84]]
---

What the computer sees.

An image is just a grid of numbers between [0, 255]

e.g. 800x600x3 (3 channel RGB)

# What makes image classification difficult?

- Huge dimensionality
  - A 256x256 RGB image amounts to ~200K input variables
- Image formation process
  - 2D summary of 3D world
  - Variances in lighting, orientations, scale
  - Scene layout, cluttering, occlusions

# Challenges: Intraclass Variation



# Challenges: Fine-grained Categories

Maine Coon



[\[Image source\]](#)

Ragdoll



[\[Image source\]](#)

American Shorthair



[\[Image source\]](#)

# Challenges: Background Clutter

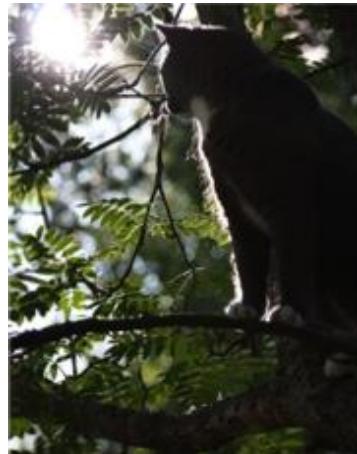


[\[Image source\]](#)

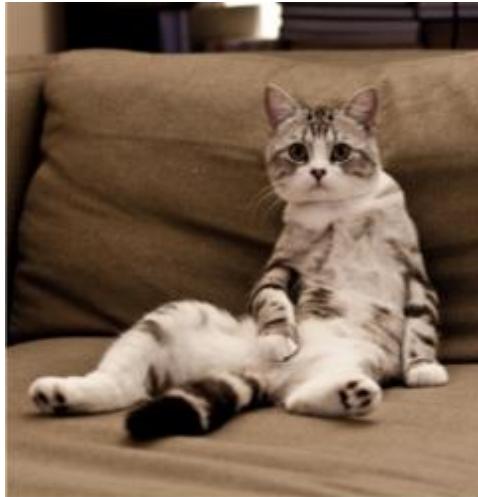
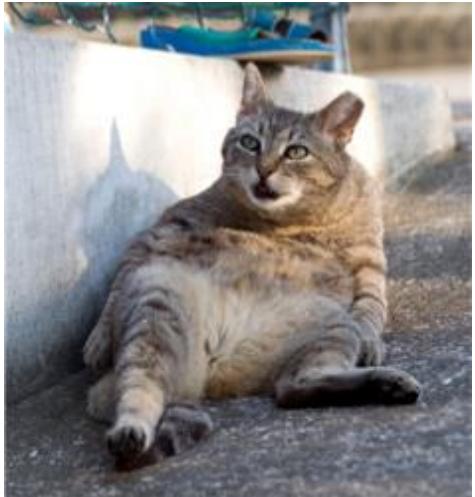


[\[Image source\]](#)

# Challenges: Illumination Changes



# Challenges: Deformation

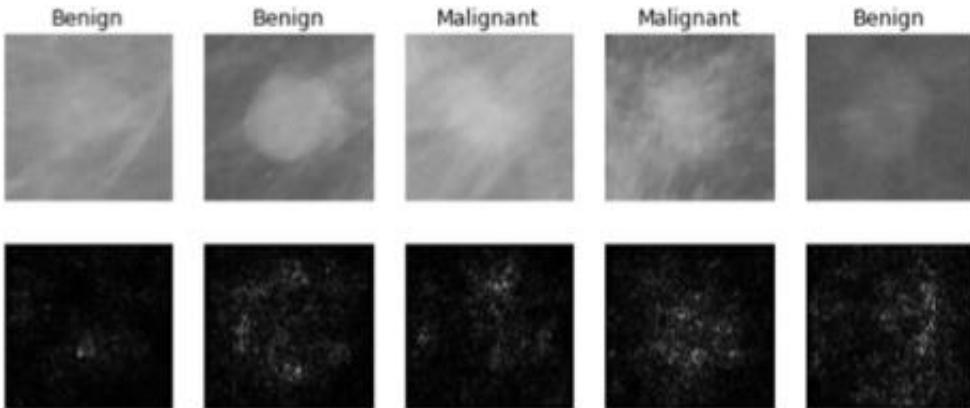


# Challenges: Occlusion



# Image Classification is very useful!

Medical Imaging



[Levy et al., 2016]

Galaxy Classification



[Dieleman et al., 2014; image sources: [1](#),[2](#),[3](#),[4](#)]

Whale recognition

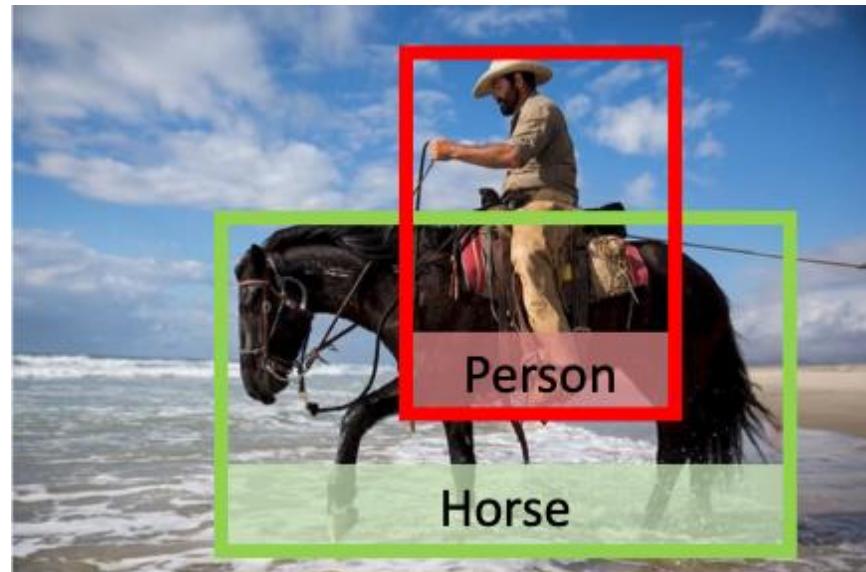


[Kaggle Challenge; [image source](#)]

# Image Classification: Building Block for other tasks!

CV

Example: Object Detection



# Image Classification: Building Block for other tasks!

CV

Example: Object Detection



**Background**

Horse

Person

Car

Truck

# Image Classification: Building Block for other tasks!

CV

Example: Object Detection



Background  
**Horse**  
**Person**  
Car  
Truck

# Image Classification: Building Block for other tasks!

CV

Example: Image Captioning



riding  
cat  
horse  
man  
when  
...  
<STOP>

What word  
to say next?

Caption:  
man

# Image Classification: Building Block for other tasks!

CV

Example: Image Captioning



riding  
cat  
horse  
man  
when  
...  
<STOP>

What word  
to say next?

Caption:  
man riding

# Image Classification: Building Block for other tasks!

CV

Example: Image Captioning



riding  
cat  
**horse**  
man  
when  
...  
<STOP>

What word  
to say next?

Caption:  
man riding horse

# Image Classification: Building Block for other tasks!

CV

Example: Image Captioning



riding  
cat  
horse  
man  
when  
...  
**<STOP>**

What word  
to say next?

Caption:  
man riding horse

# Image Classification: Building Block for other tasks!



Example: Playing Go



Where to  
play next?

(1, 1)  
(1, 2)  
...  
(1, 19)  
...  
(19, 19)

# How to Implement an Image Classifier?

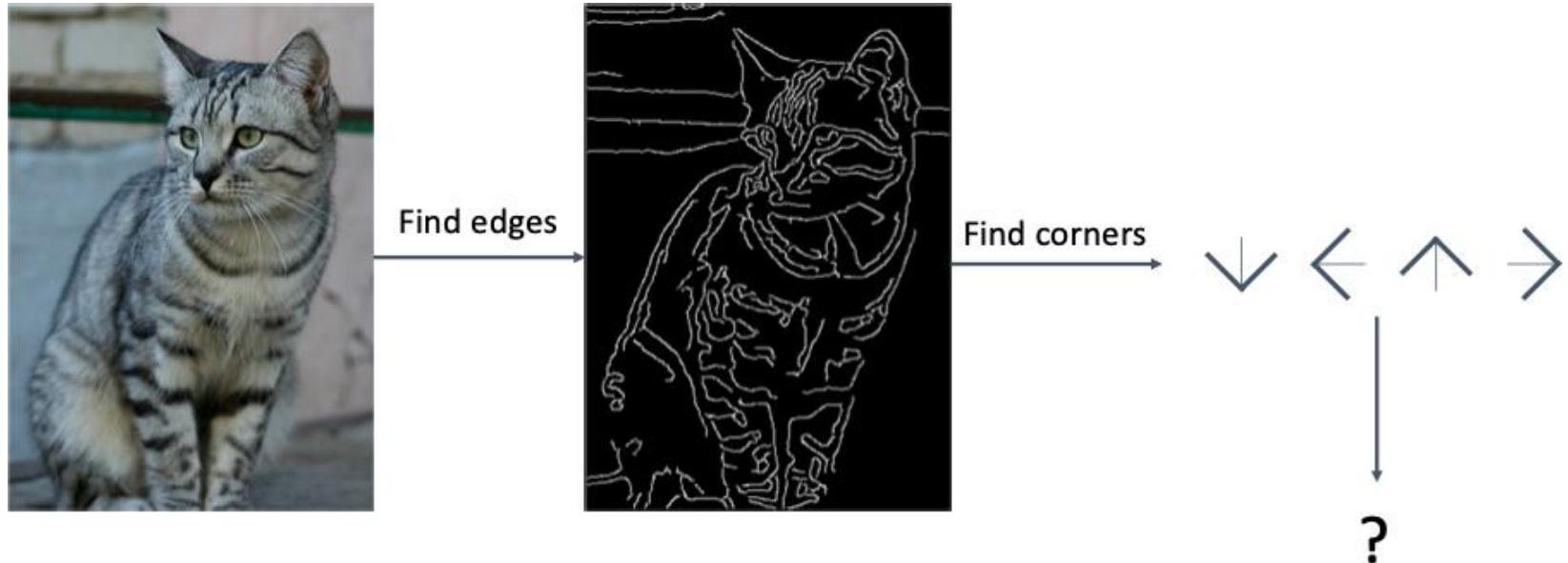
```
def classify_image(image):  
    # Some magic here?  
    return class_label
```

Unlike e.g. sorting a list of numbers,

**no obvious way** to hard-code the algorithm  
for recognizing a cat, or other classes.

# How to Implement an Image Classifier?

You could try...



# Today's Agenda

---

## Image Classification

- Problem Setting and Challenges
- **Bag of words**
- Nearest Neighbors
- Hyperparameters and Training
- Linear Classifier
- SVMs
- Classification Metrics and Performance Evaluation

# Bag of Words Classification

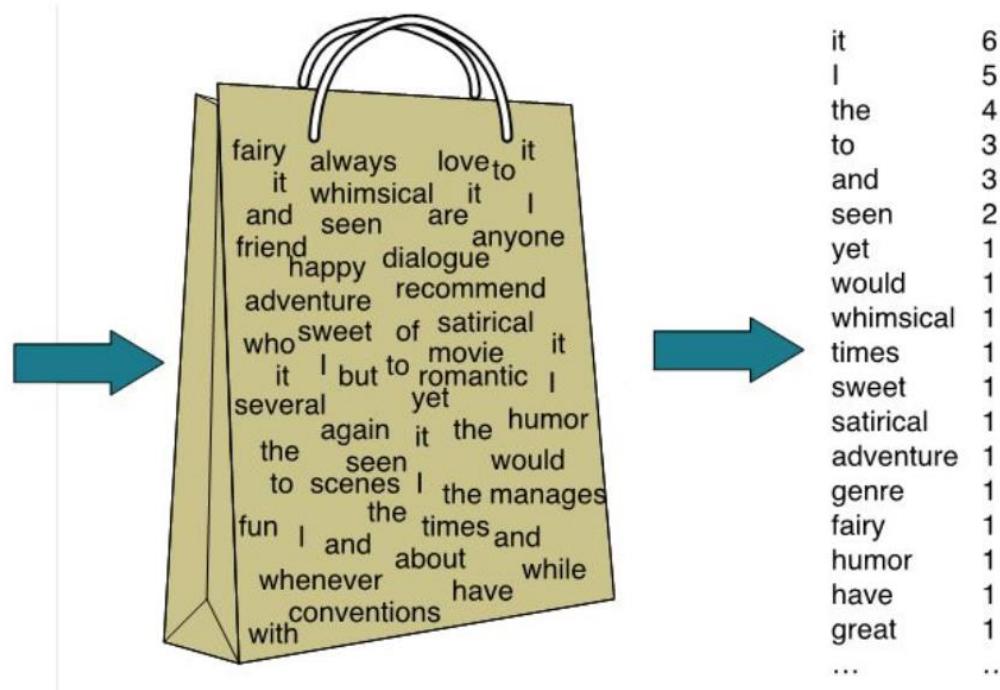
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

Text case

# Bag of Words Classification

## The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



# Bag of Words Classification

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

Text case

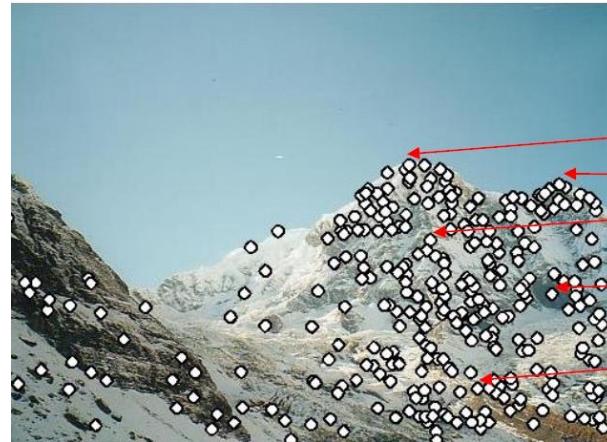
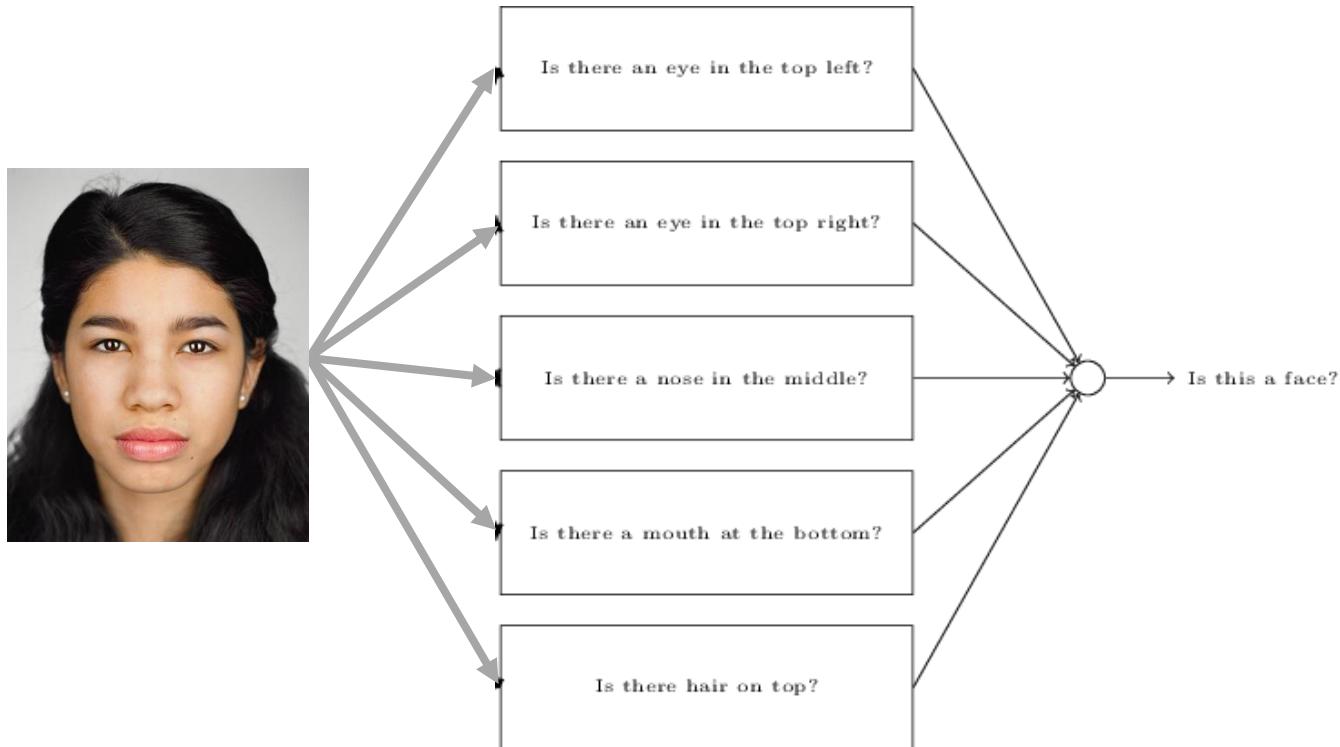
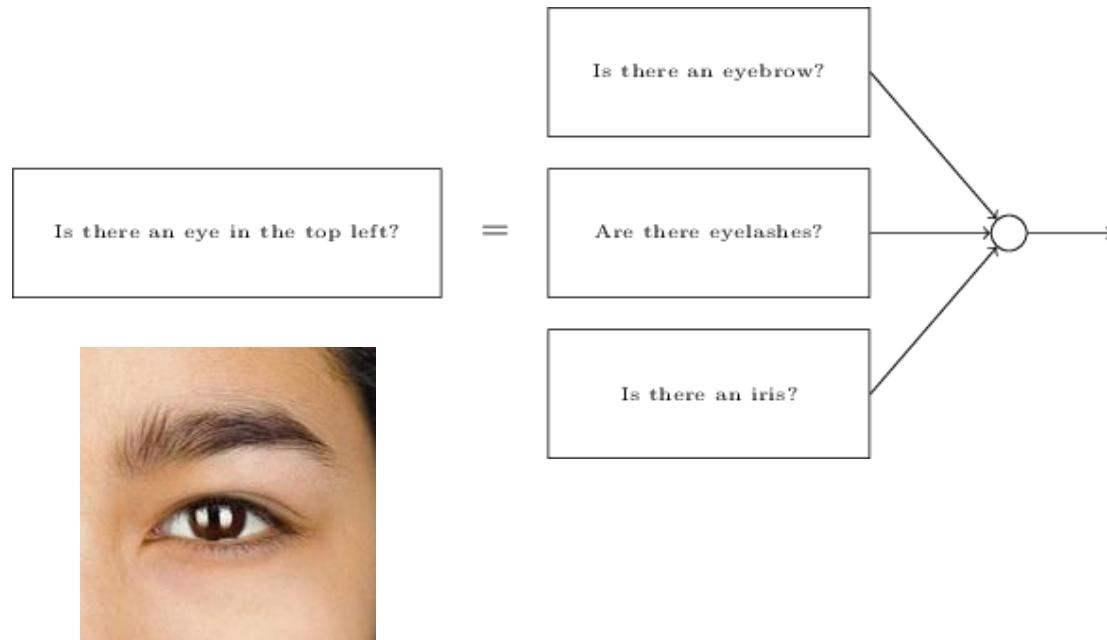


Image case

# Example



# Example

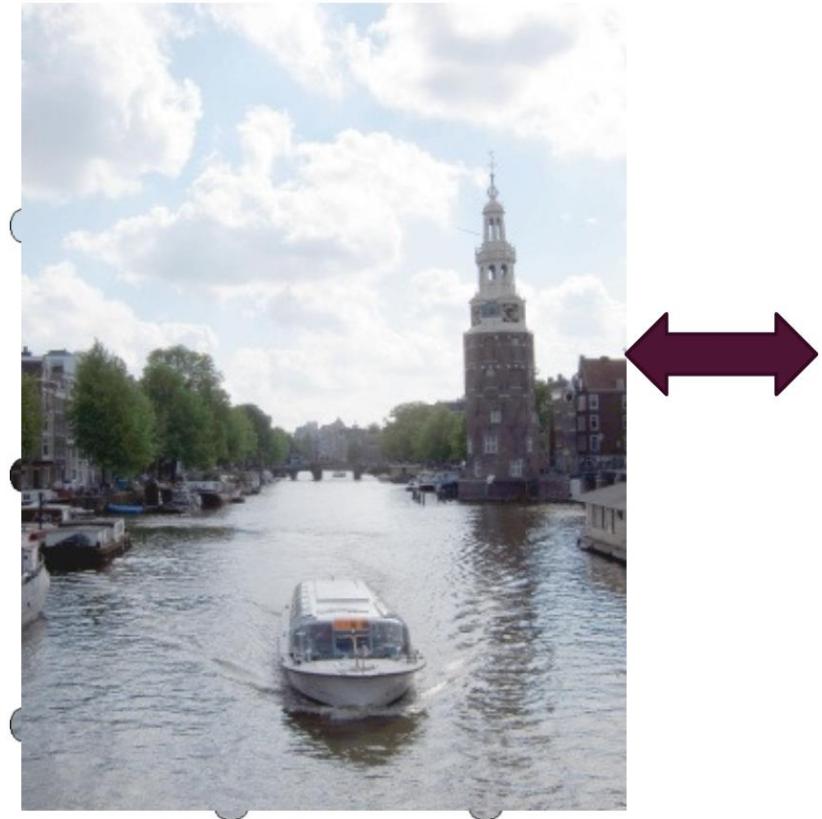


# Bag of Visual Words

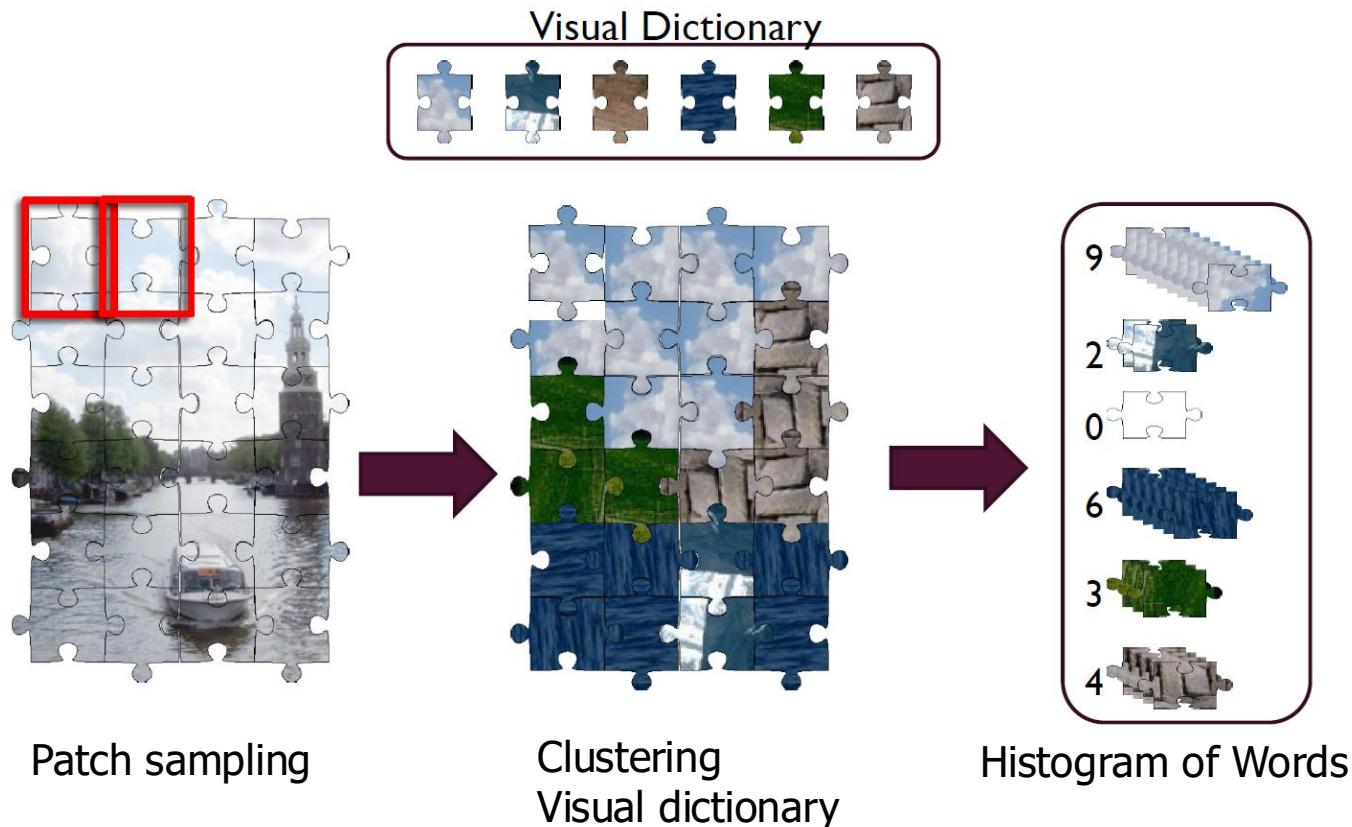


Slide credits: Jan van Gemert

# Images as Bag of Patches



# Images as Bag of Patches

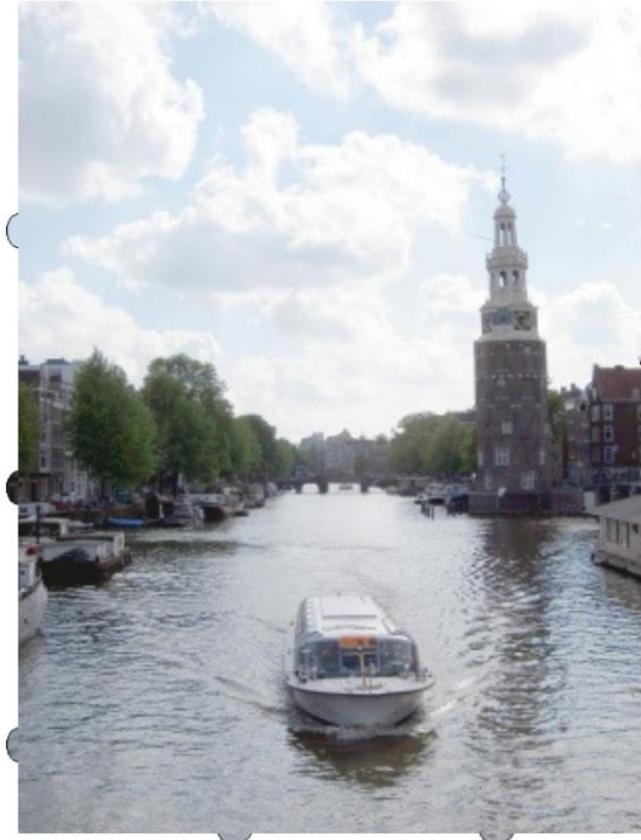


# Bag of Words: Key Ingredients

1. Patch sampling
2. Patch descriptors
3. Visual Dictionary
4. Histogram creation

There are choices to be made!

# Patch Sampling



# Patch Descriptor Extraction

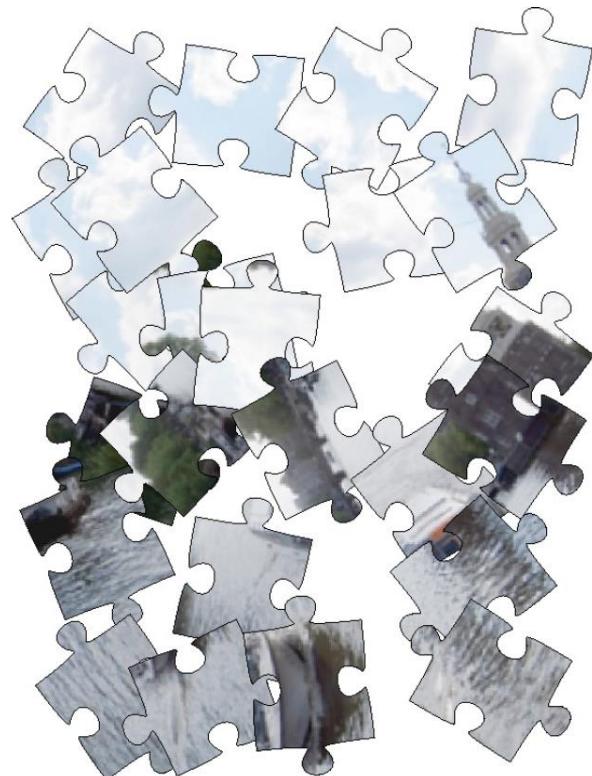
SIFT

RGB

HOG

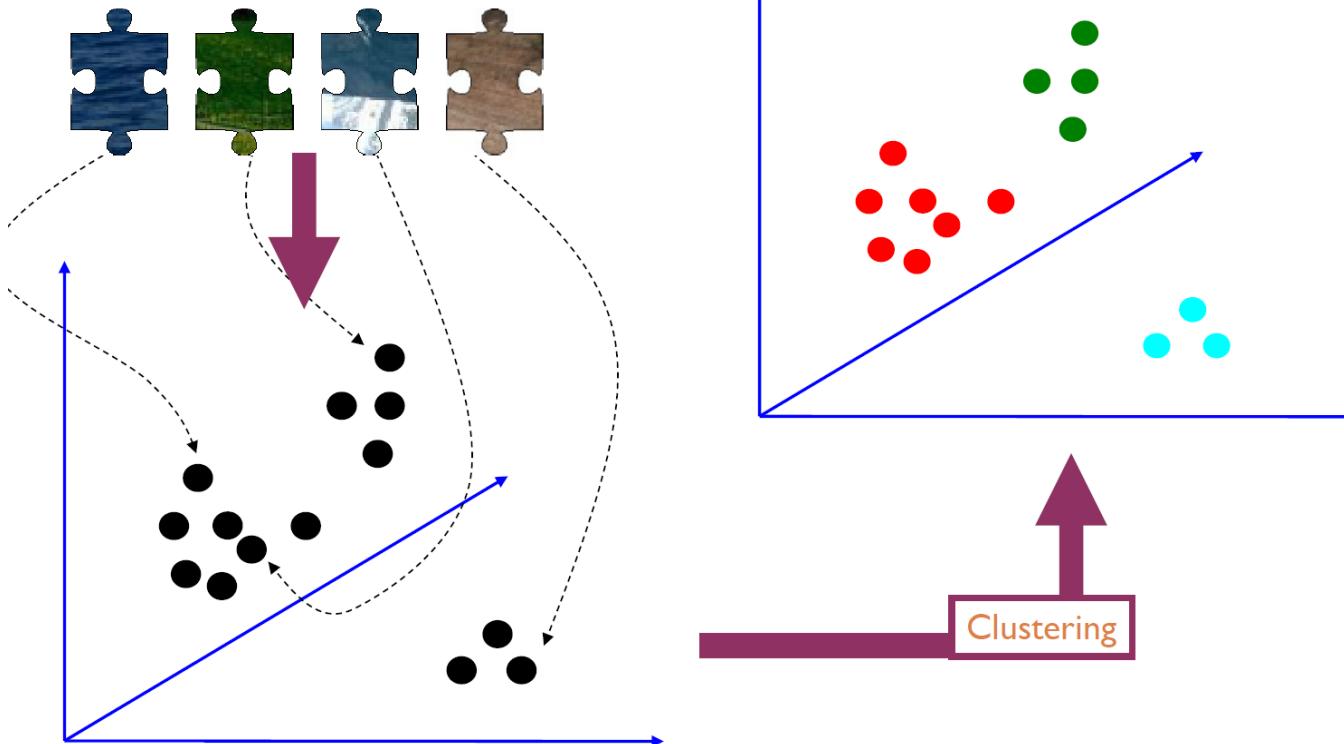
SURF

LBP



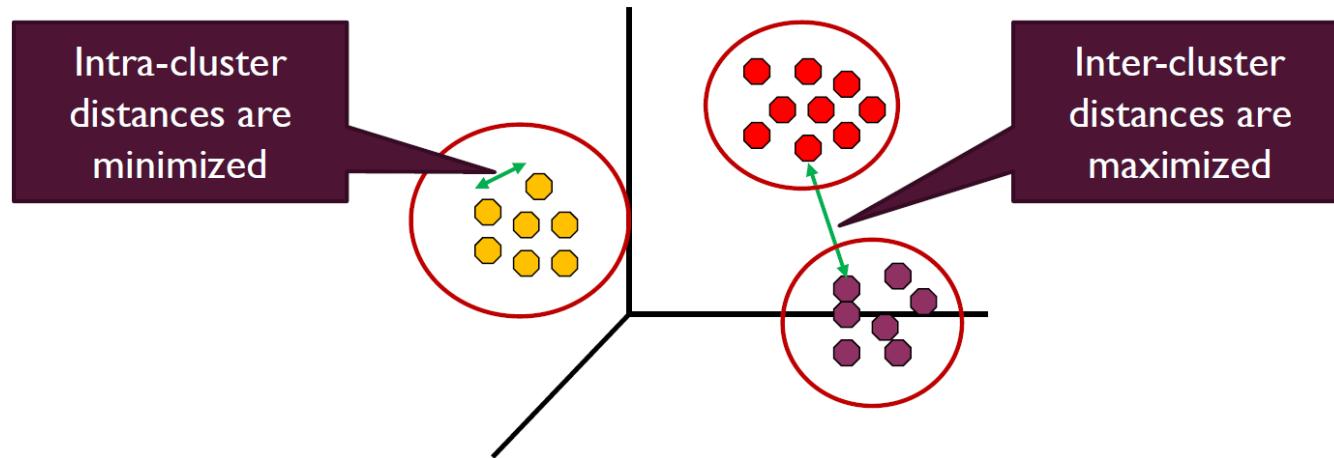
# Feature Clustering

## CLUSTERING FOR BOW

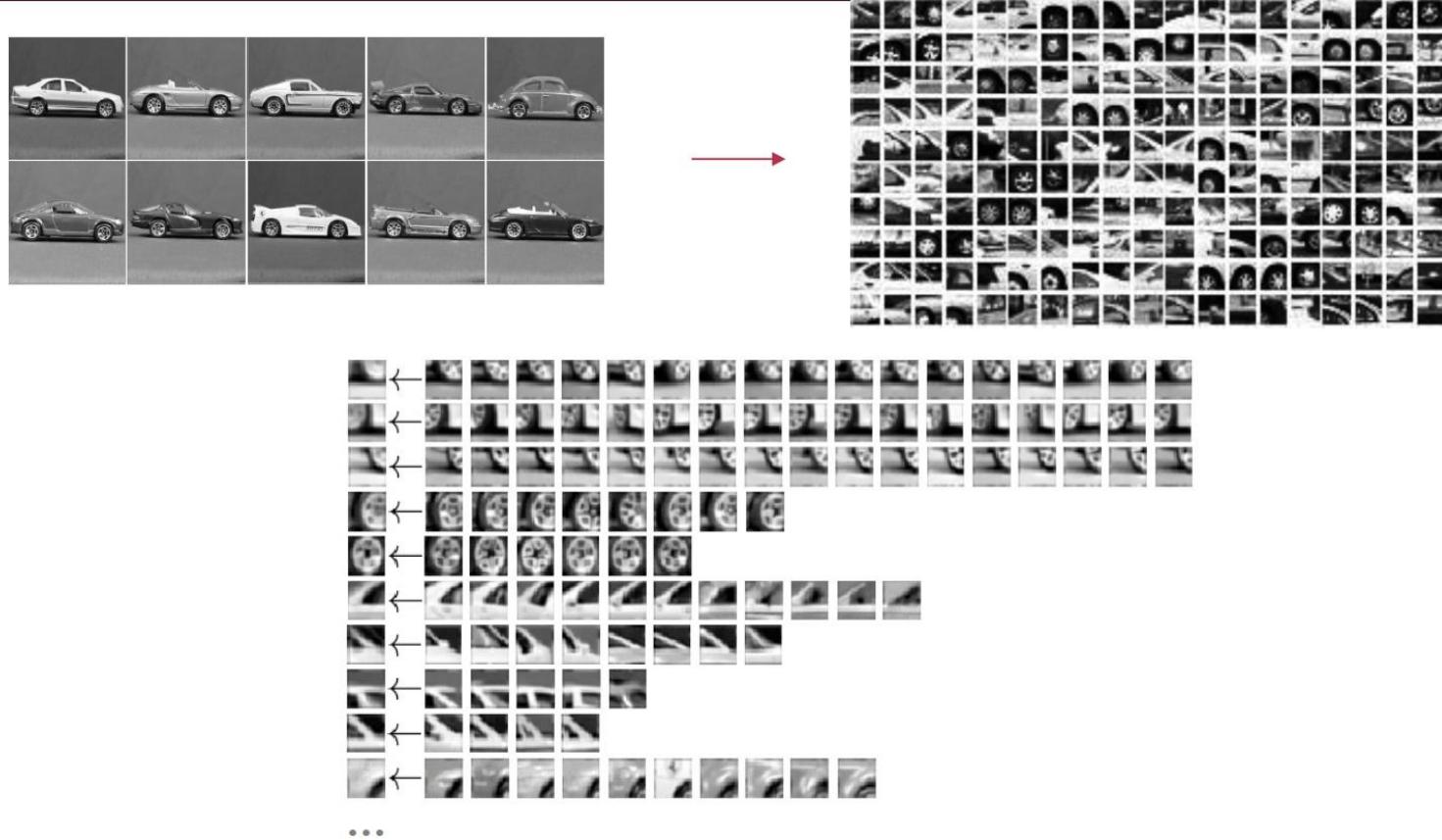


# Feature Clustering

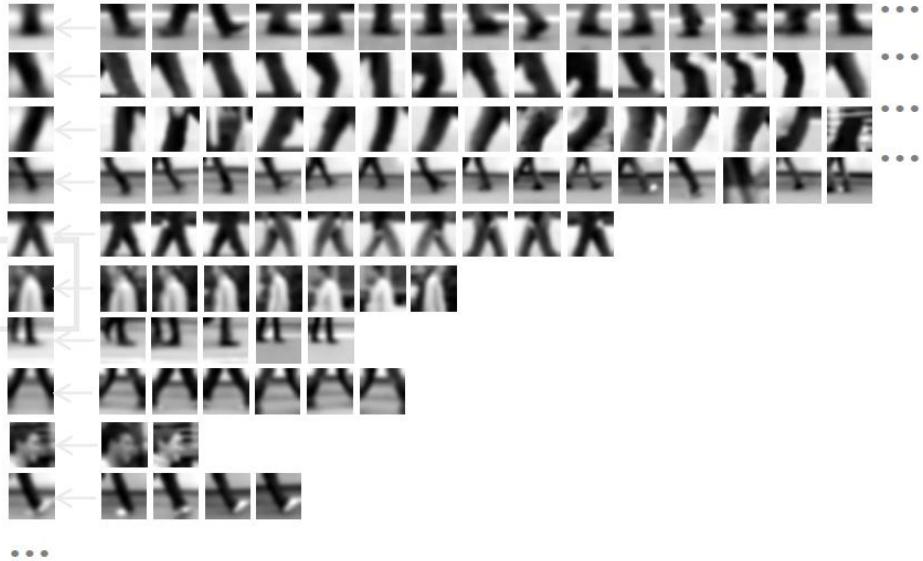
Find clusters of similar descriptors



# Codebook Example



# Codebook Example

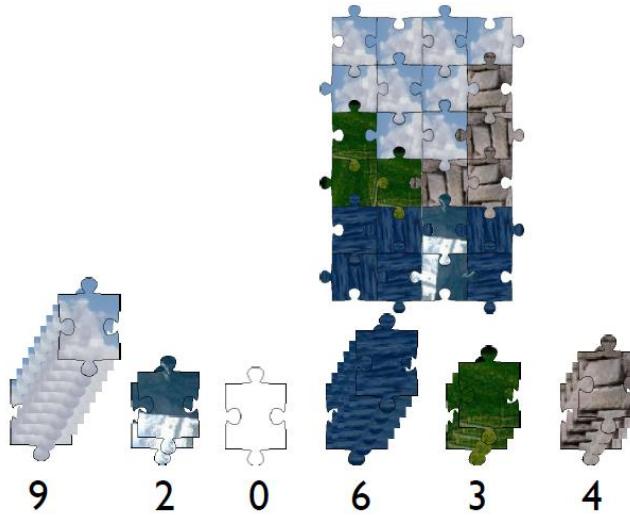


Appearance codebook

# Histogram Creation

Simplest form:

- Assign each descriptor to nearest prototype
- Count how often each prototype appears



# Today's Agenda

---

## Image Classification

- Problem Setting and Challenges
- Bag of words
- **Nearest Neighbors**
- Hyperparameters and Training
- Linear Classifier
- SVMs
- Classification Metrics and Performance Evaluation

# Machine Learning: Data-Driven Approach

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

```
def train(images, labels):
    # Machine learning!
    return model
```

```
def train(images, labels):
    # Machine learning!
    return model
```

Example training set

Airplane



Car



Bird



Cat



Deer

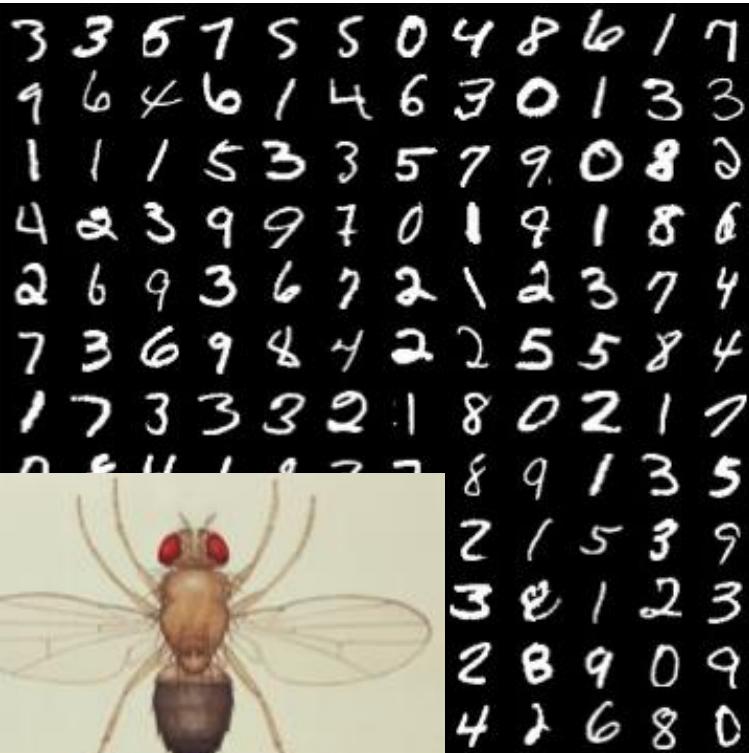


# Image Classification Datasets: MNIST



**10 classes:** Digits 0 to 9  
**28x28** grayscale images  
**50k** training images  
**10k** test images

# Image Classification Datasets: MNIST



**10 classes:** Digits 0 to 9

**28x28** grayscale images

**50k** training images

**10k** test images

**“Drosophila of computer vision”**

Results from MNIST often do not hold on more complex datasets!

# Image Classification Datasets: CIFAR10

airplane



automobile



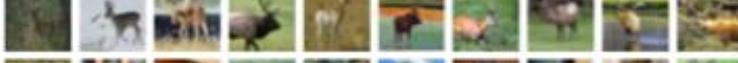
bird



cat



deer



dog



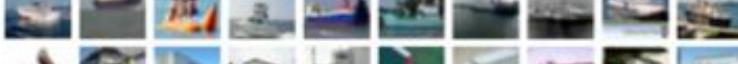
frog



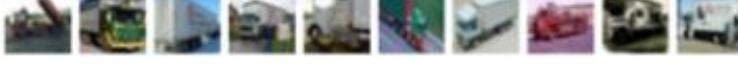
horse



ship



truck



**10 classes**

**50k training images (5k per class)**

**10k testing images (1k per class)**

**32x32 RGB images**

**“Drosophila of computer vision”**

Results from MNIST often do not hold on more complex datasets!

# Image Classification Datasets: CIFAR100



**100 classes**

**50k training images (500 per class)**

**10k testing images (100 per class)**

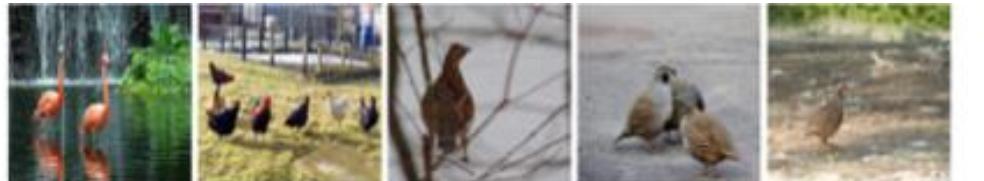
**32x32 RGB images**

**20 superclasses** with 5 classes each:

Aquatic mammals: beaver, dolphin, otter, seal, whale

Trees: Maple, oak, palm, pine, willow

# Image Classification Datasets: ImageNet



flamingo

cock

ruffed grouse

quail

partridge

**1000 classes**

**~1.3M** training images (~1.3K per class)

**50K** validation images (50 per class)

**100K** test images (100 per class)



Egyptian cat

Persian cat

Siamese cat

tabby

lynx

Performance metric: **Top 5 accuracy**

Algorithm predicts 5 labels for each image; one of them needs to be right



dalmatian

keeshond

miniature schnauzer

standard schnauzer

giant schnauzer

# Image Classification Datasets: ImageNet



**IMAGENET**

**22K** categories and **15M** images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
  - Food
  - Materials
- Structures
  - Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activity

[www.image-net.org](http://www.image-net.org)

Deng et al. 2009,  
Russakovsky et al. 2015

Slide credit: Fei-Fei Li

# Image Classification Datasets: MIT Places



**365 classes of different scene types**  
**~8M training images**  
**18.25K val images (50 per class)**  
**328.5K test images (900 per class)**

Images have variable size, often  
resize to **256x256** for training

# Nearest Neighbor Classifier

```
def train(images, labels):  
    # Machine learning!  
    return model
```



Memorize all data  
and labels

```
def train(images, labels):  
    # Machine learning!  
    return model
```



Predict the label of  
the most similar  
training image

# Distance Metric for Image Comparison

L1 distance:  $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test image				training image				pixel-wise absolute value differences			
56	32	10	18	10	20	24	17	46	12	14	1
90	23	128	133	8	10	89	100	82	13	39	33
24	26	178	200	12	16	178	170	12	10	0	30
2	0	255	220	4	32	233	112	2	32	22	108

-

=

add  
 → 456

# Nearest Neighbor Classifier

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Memorize training data

For each test image:  
Find nearest training image  
Return label of nearest image

# Nearest Neighbor Classifier

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

**Q:** With  $N$  examples,  
how fast is training?

**A:**  $\sim O(1)$  (shallow copy)

**Q:** With  $N$  examples,  
how fast is testing?

**A:**  $O(N)$

This is **bad**: We can afford slow,  
but we need fast testing!

# Nearest Neighbor Classifier

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

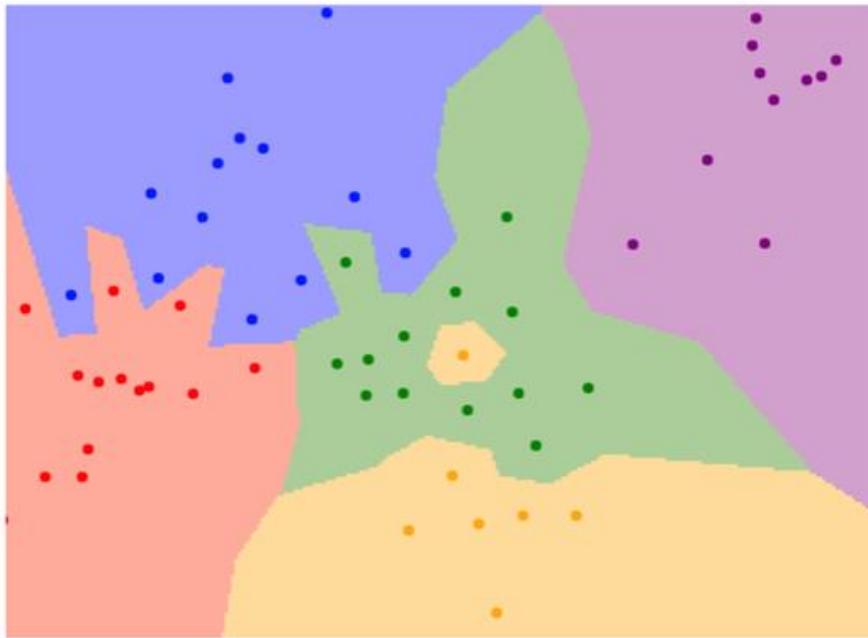
    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

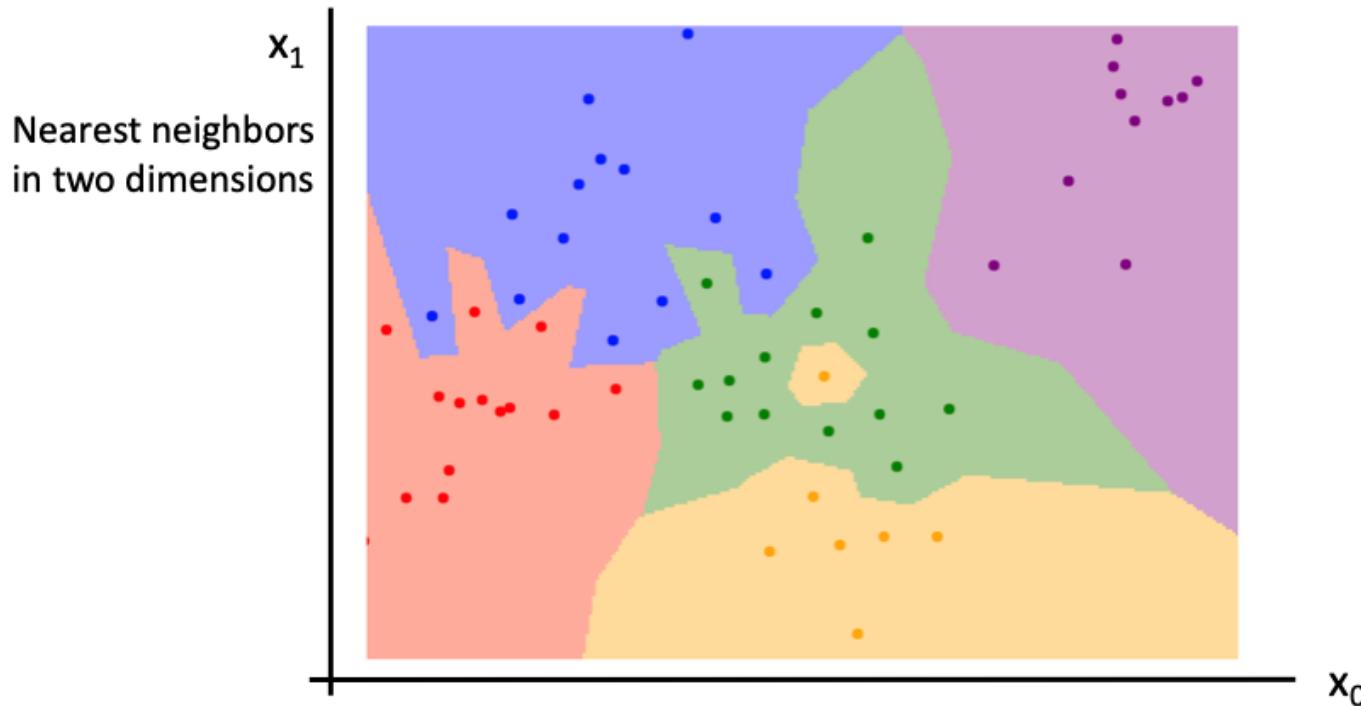
        return Ypred
```

There are many methods for fast / approximate nearest neighbors; e.g. see  
<https://github.com/facebookresearch/faiss>

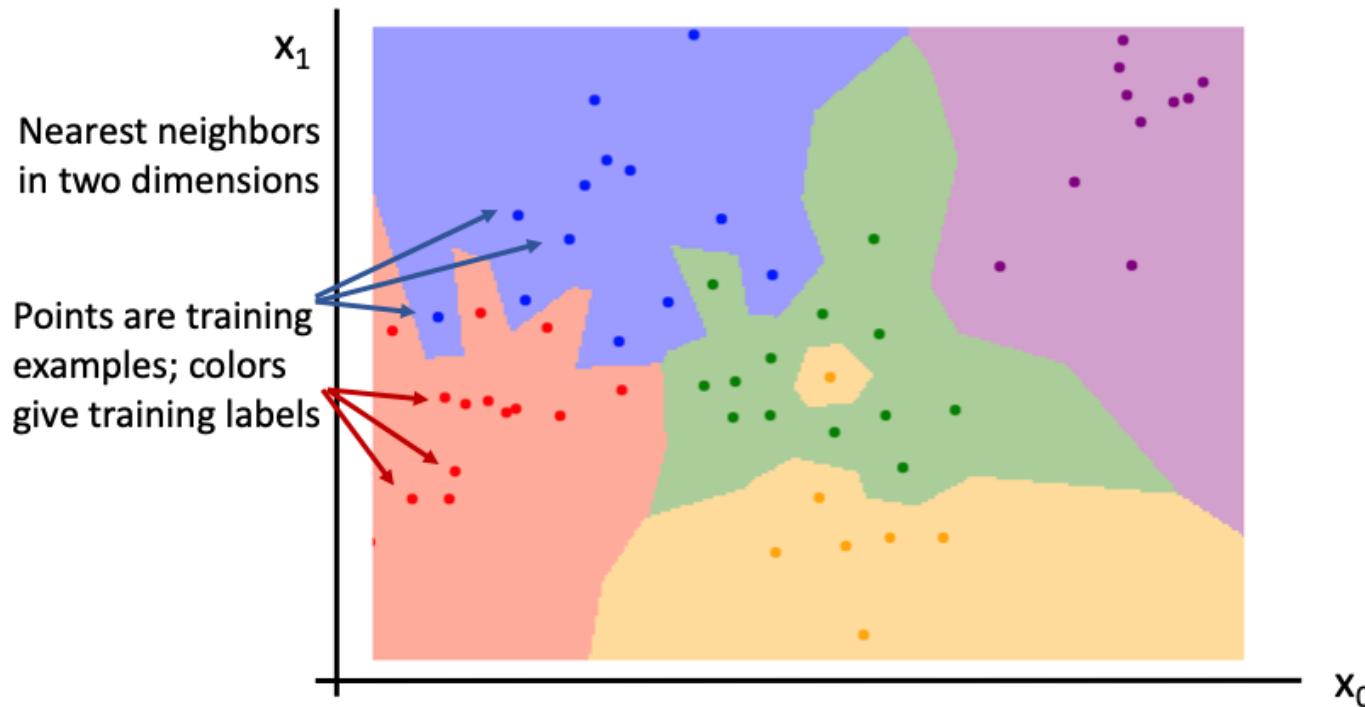
# Nearest Neighbor Classifier



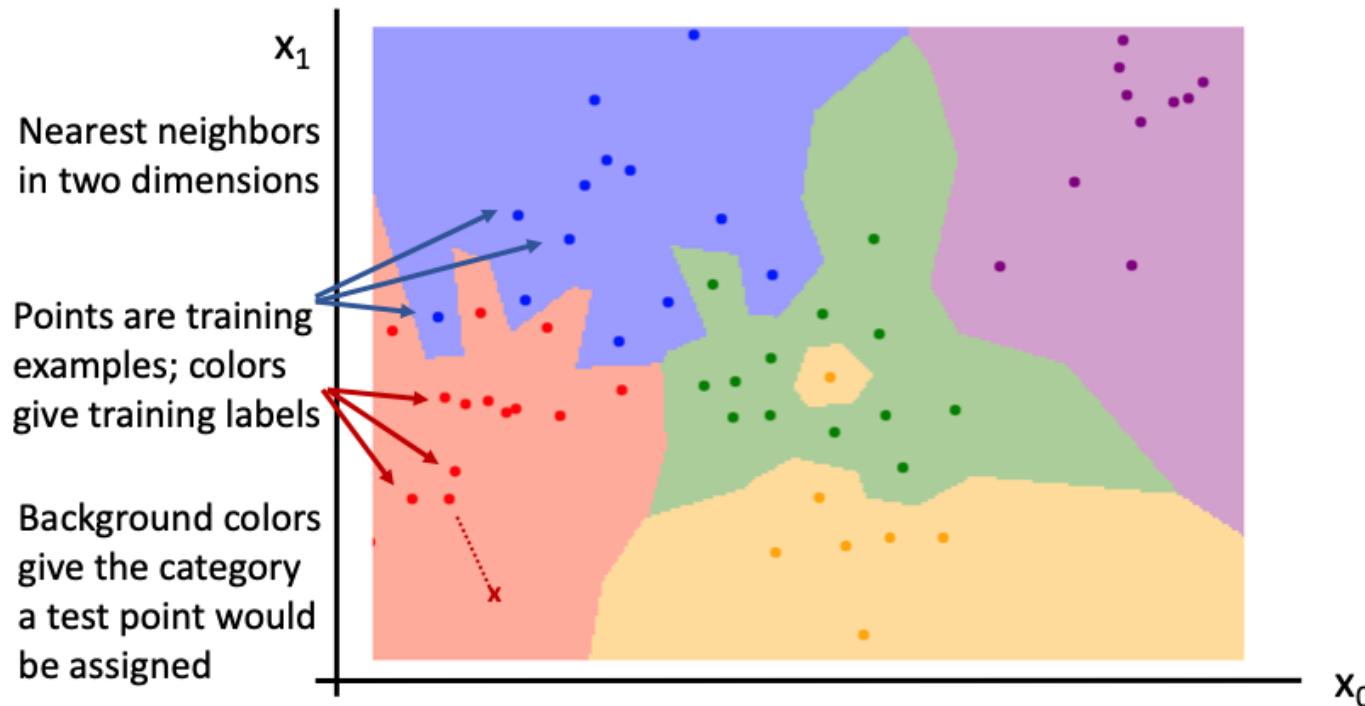
# Nearest Neighbor Classifier



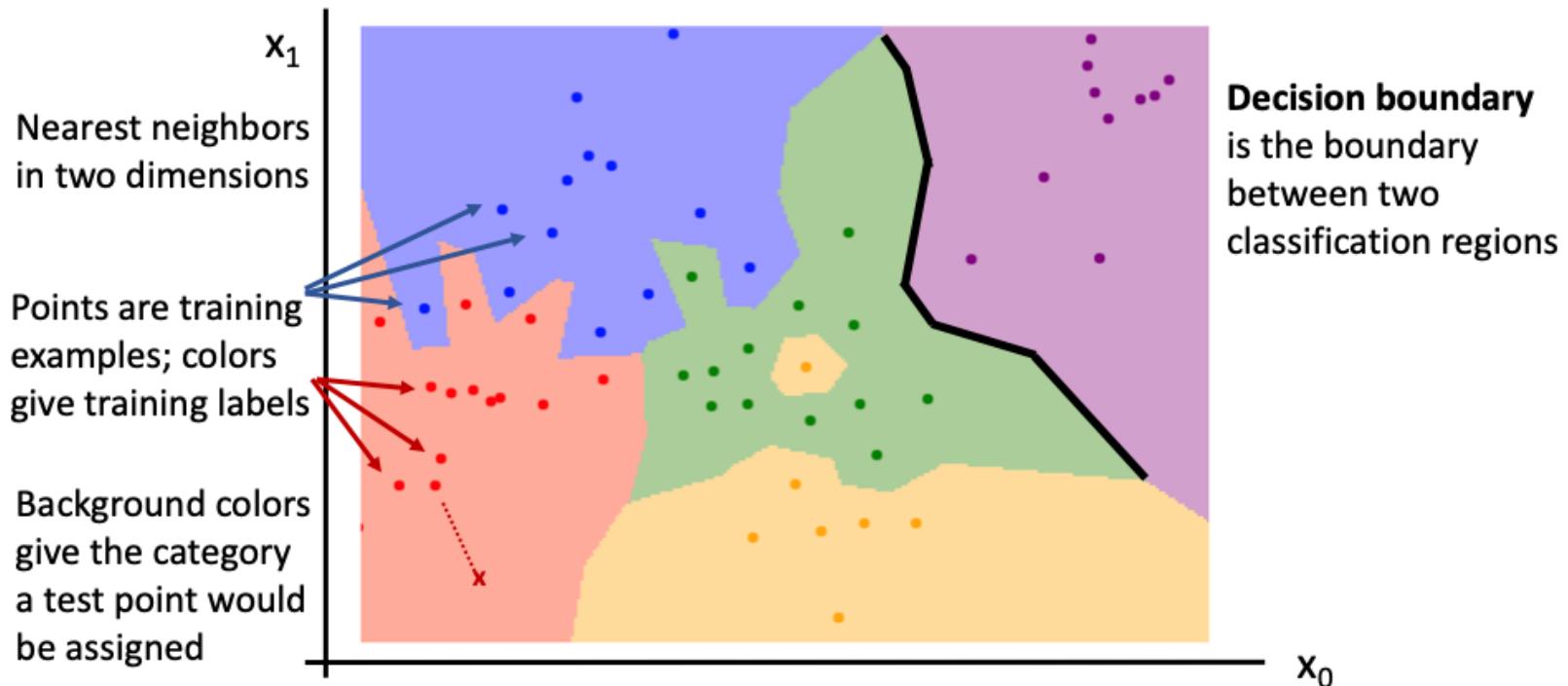
# Nearest Neighbor Classifier



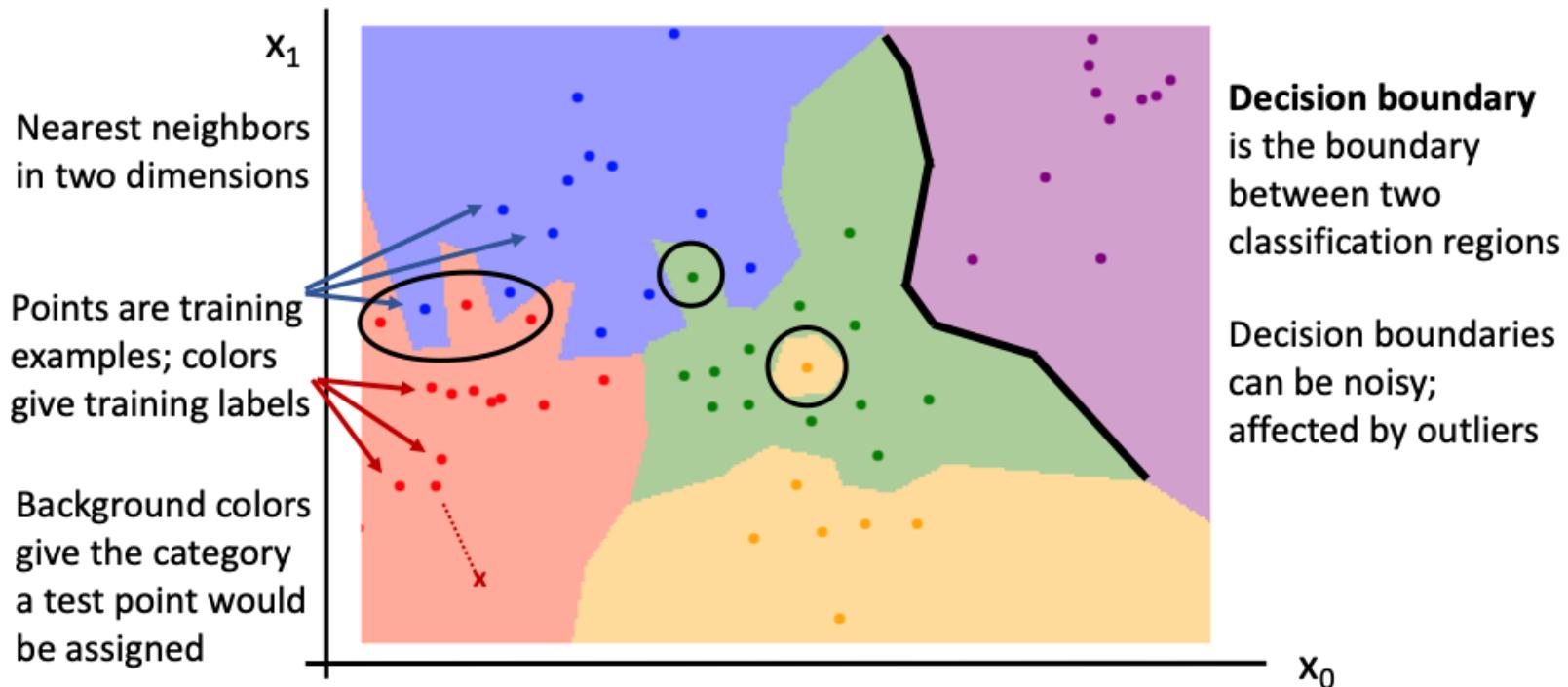
# Nearest Neighbor Classifier



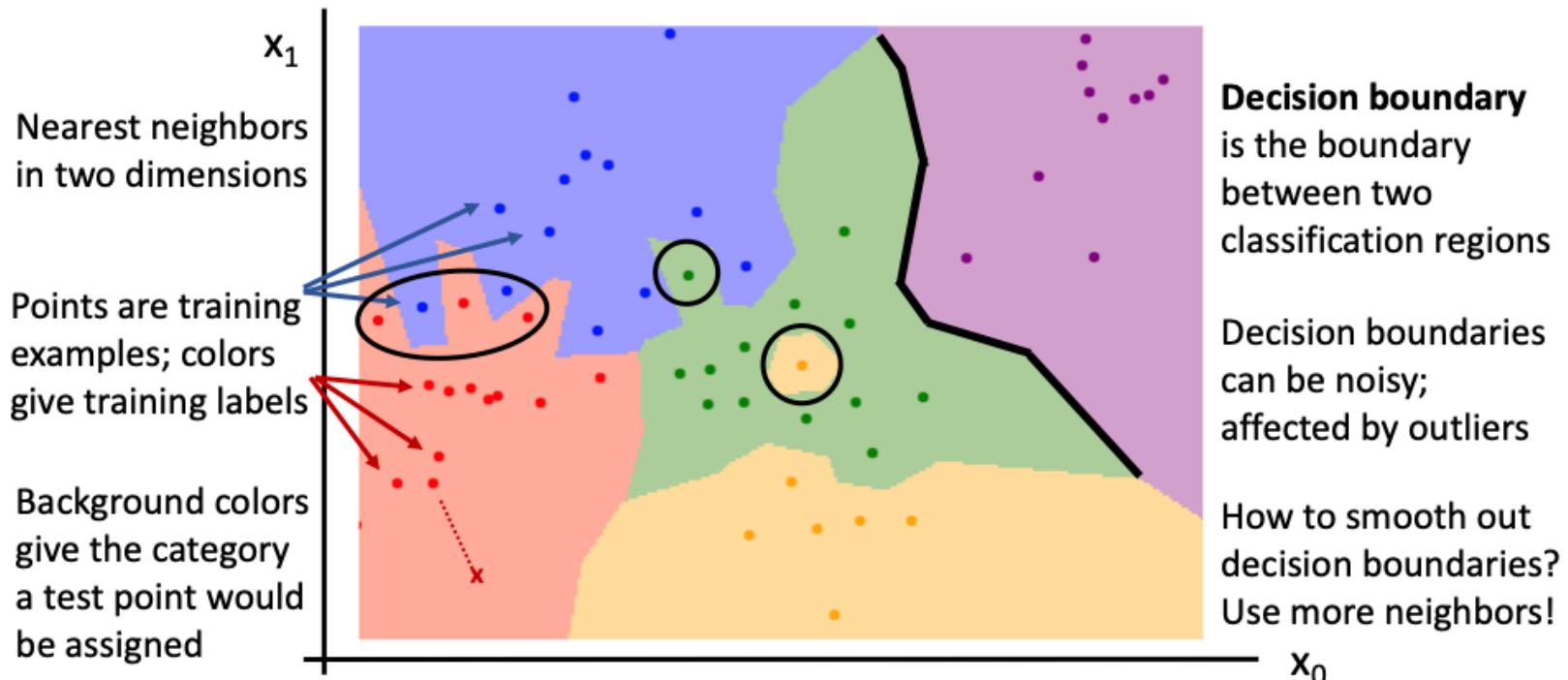
# Nearest Neighbor Classifier



# Nearest Neighbor Classifier

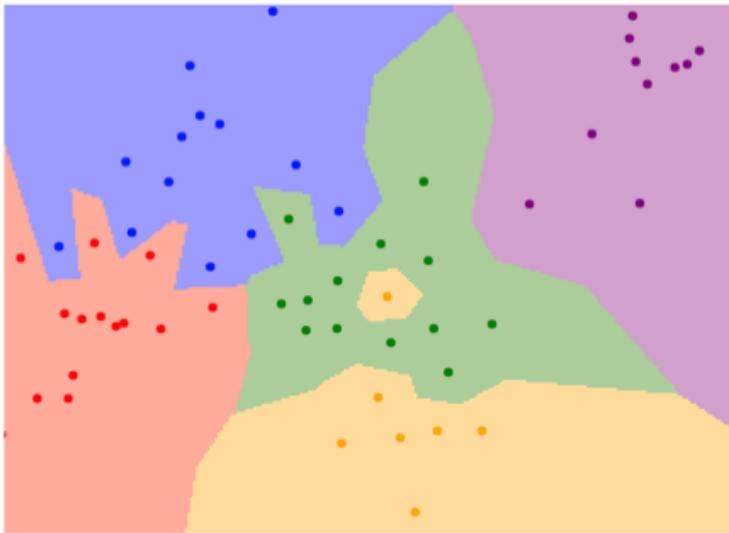


# Nearest Neighbor Classifier

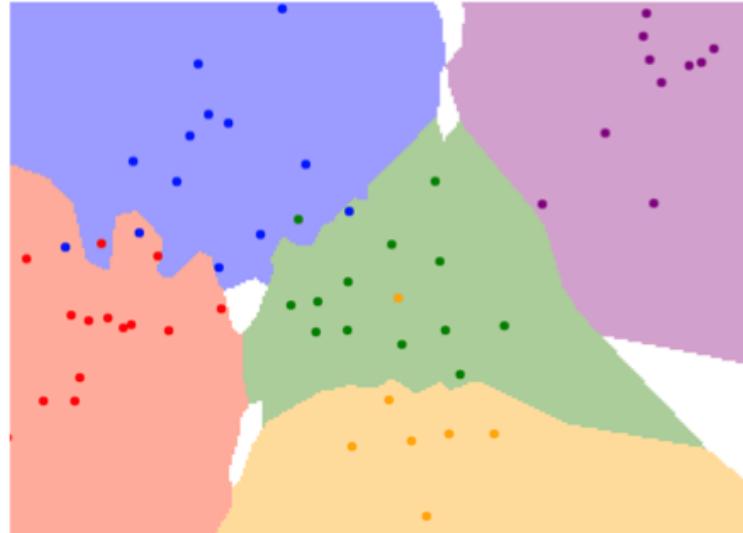


# K-Nearest Neighbors

$K = 1$

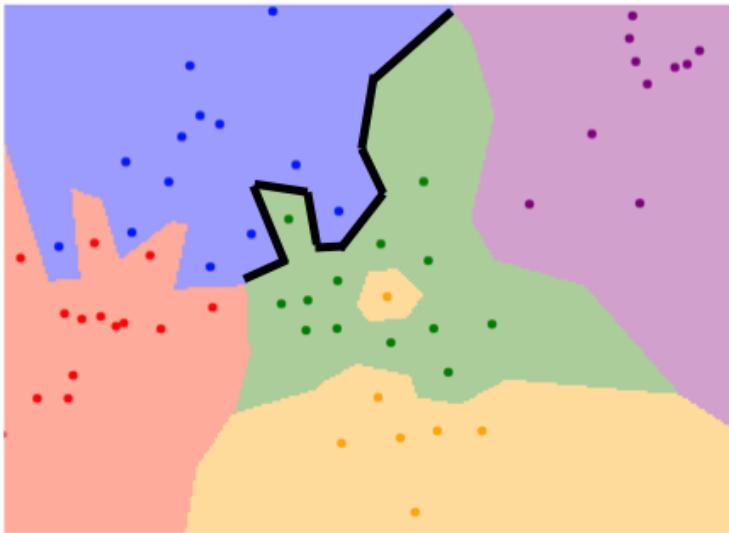


$K = 3$

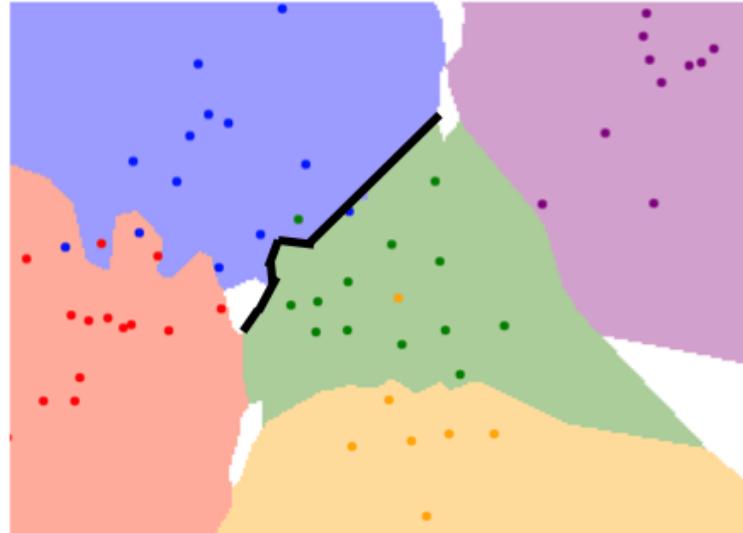


# K-Nearest Neighbors

$K = 1$

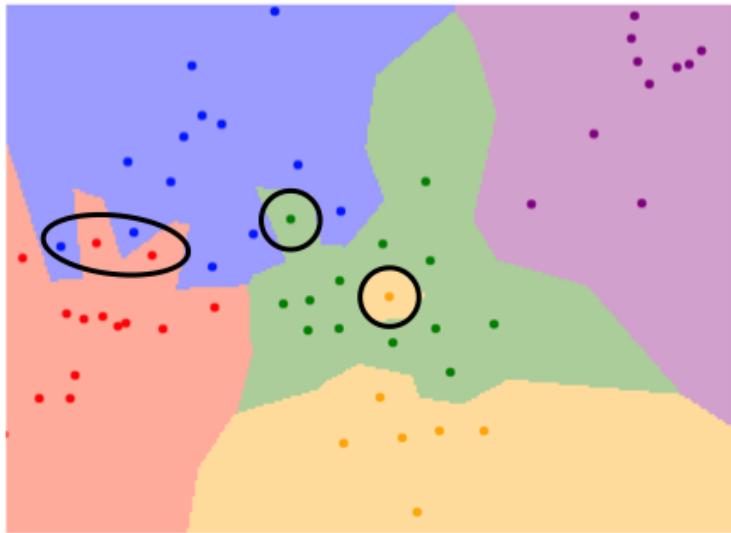


$K = 3$

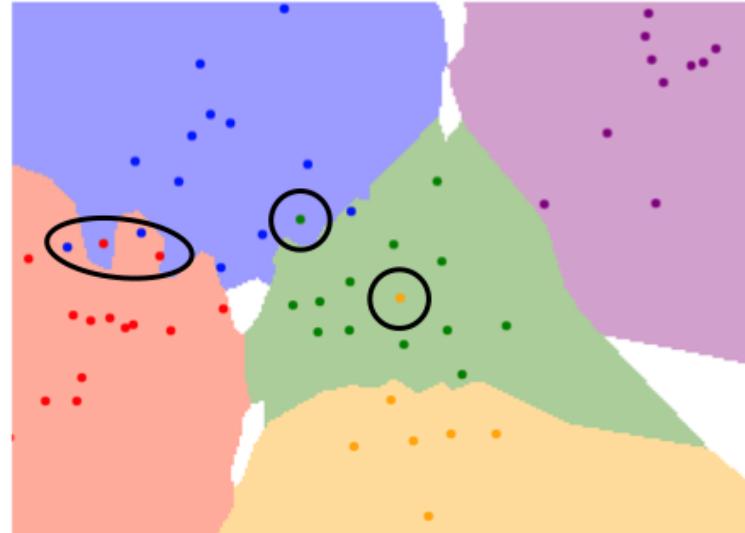


# K-Nearest Neighbors

$K = 1$

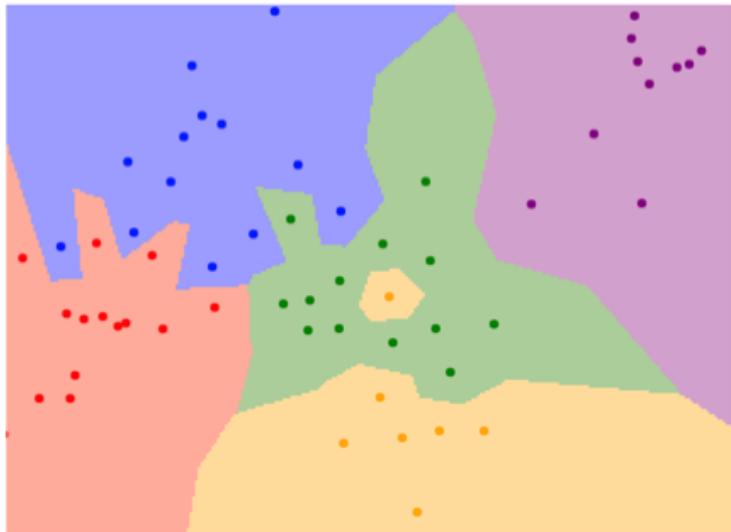


$K = 3$

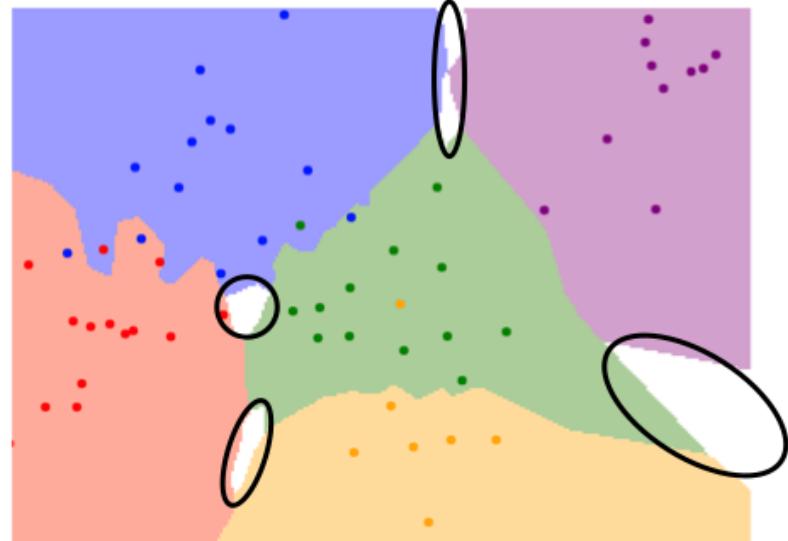


# K-Nearest Neighbors

$K = 1$



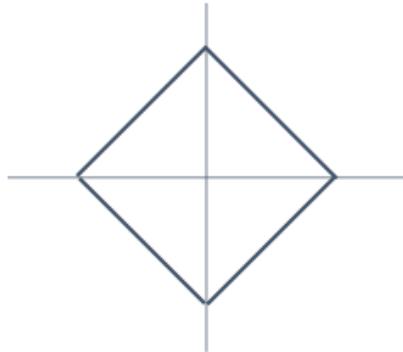
$K = 3$



# K-Nearest Neighbors: Distance Metric

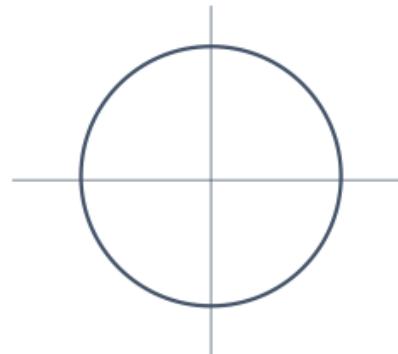
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

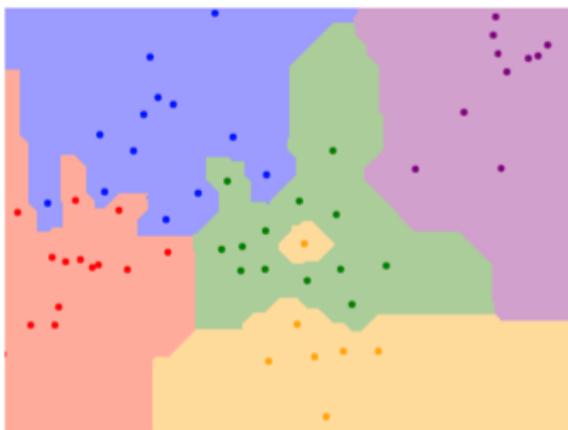
$$d_2(I_1, I_2) = \left( \sum_p (I_1^p - I_2^p)^2 \right)^{\frac{1}{2}}$$



# K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_1(I_1, I_2) = \left( \sum_p (I_1^p - I_2^p)^2 \right)^{\frac{1}{2}}$$



# K-Nearest Neighbor on raw pixels is seldom used



- Very slow at test time
- Distance metrics on pixels are not informative

Original



Boxed



Shifted



Tinted



(all 3 images have same L2 distance to the one on the left)

# K-Nearest Neighbor with ConvNet features works well!



# K-Nearest Neighbor with ConvNet features works well!

Example: Image Captioning with Nearest Neighbor



A bedroom with a bed and a couch.



A cat sitting in a bathroom sink.



A train is stopped at a train station.



A wooden bench in front of a building.

# Today's Agenda

---

## Image Classification

- Problem Setting and Challenges
- Bag of words
- Nearest Neighbors
- **Hyperparameters and Training**
- Linear Classifier
- SVMs
- Classification Metrics and Performance Evaluation

# Hyperparameters

What is the best value of K to use?

What is the best **distance metric** to use?

These are examples of **hyperparameters**: choices about our learning algorithm that we don't learn from the training data; instead we set them at the start of the learning process

Very problem-dependent. In general need to try them all and see what works best for our data / task.

# Setting Hyperparameters

**Idea #1:** Choose hyperparameters that work best on the data

Your Dataset

# Setting Hyperparameters

**Idea #1:** Choose hyperparameters that work best on the data

**BAD:**  $K = 1$  always works perfectly on training data

Your Dataset

# Setting Hyperparameters

**Idea #1:** Choose hyperparameters that work best on the data

**BAD:**  $K = 1$  always works perfectly on training data

Your Dataset

**Idea #2:** Split data into **train** and **test**, choose hyperparameters that work best on test data

train

test

# Setting Hyperparameters

**Idea #1:** Choose hyperparameters that work best on the data

**BAD:**  $K = 1$  always works perfectly on training data

Your Dataset

**Idea #2:** Split data into **train** and **test**, choose hyperparameters that work best on test data

**BAD:** No idea how algorithm will perform on new data

train

test

# Setting Hyperparameters

**Idea #1:** Choose hyperparameters that work best on the data

**BAD:** K = 1 always works perfectly on training data

Your Dataset

**Idea #2:** Split data into **train** and **test**, choose hyperparameters that work best on test data

**BAD:** No idea how algorithm will perform on new data

train

test

**Idea #3:** Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

**Better!**

train

validation

test

# Setting Hyperparameters

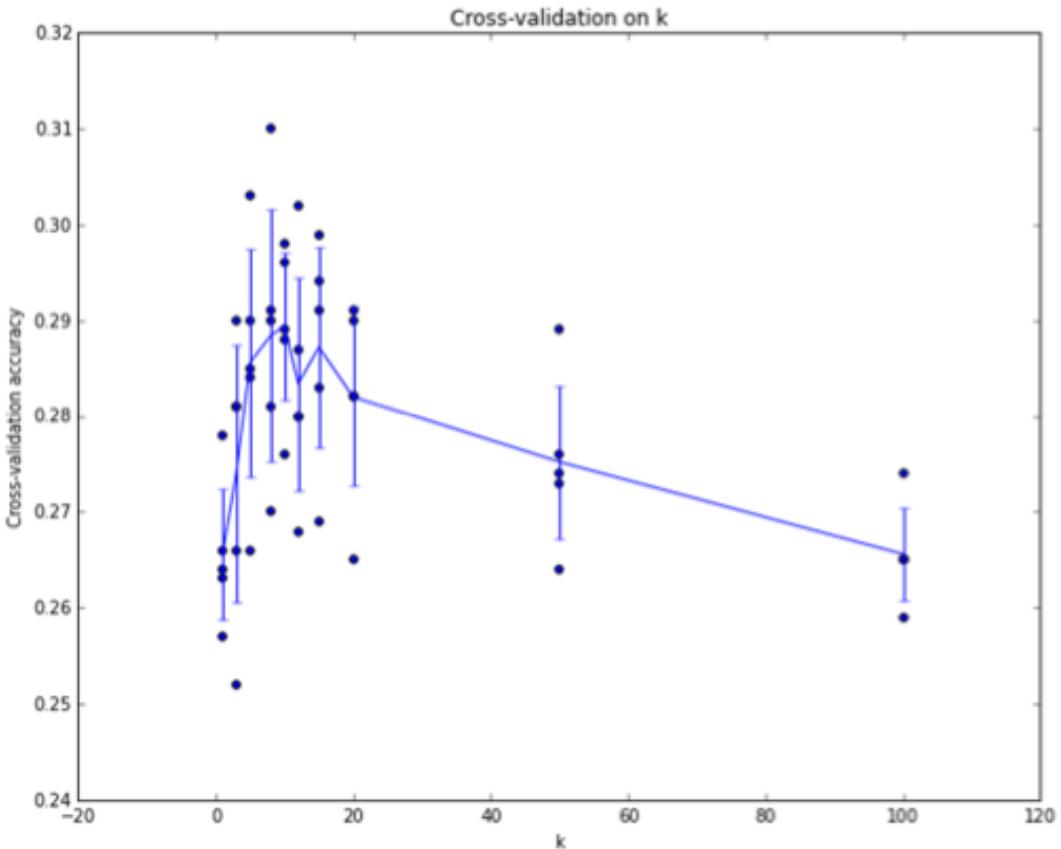
Your Dataset

**Idea #4: Cross-Validation:** Split data into **folds**, try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but (unfortunately) not used too frequently in deep learning

# Setting Hyperparameters



Example of 5-fold cross-validation for the value of  $k$ .

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation

(Seems that  $k \sim 7$  works best for this data)

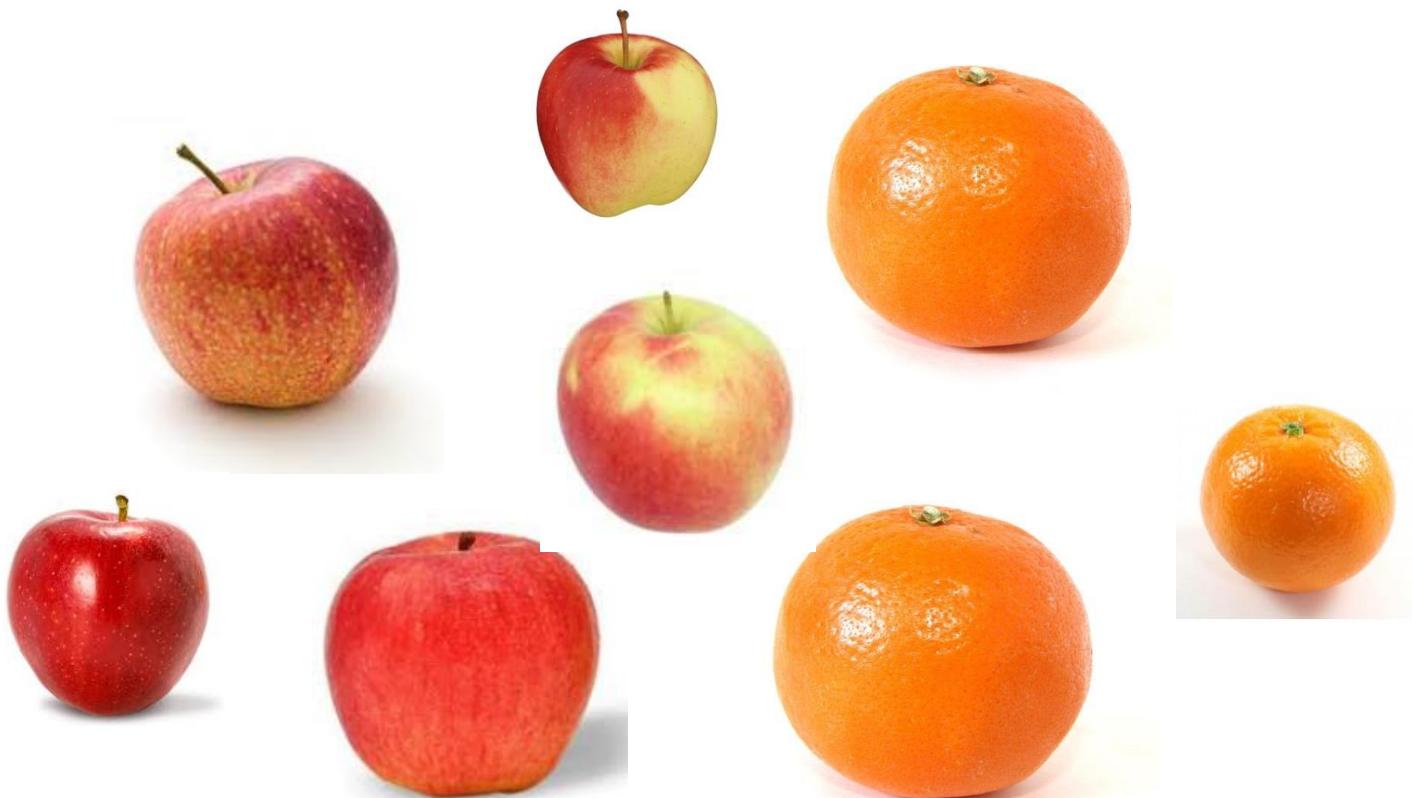
# Today's Agenda

---

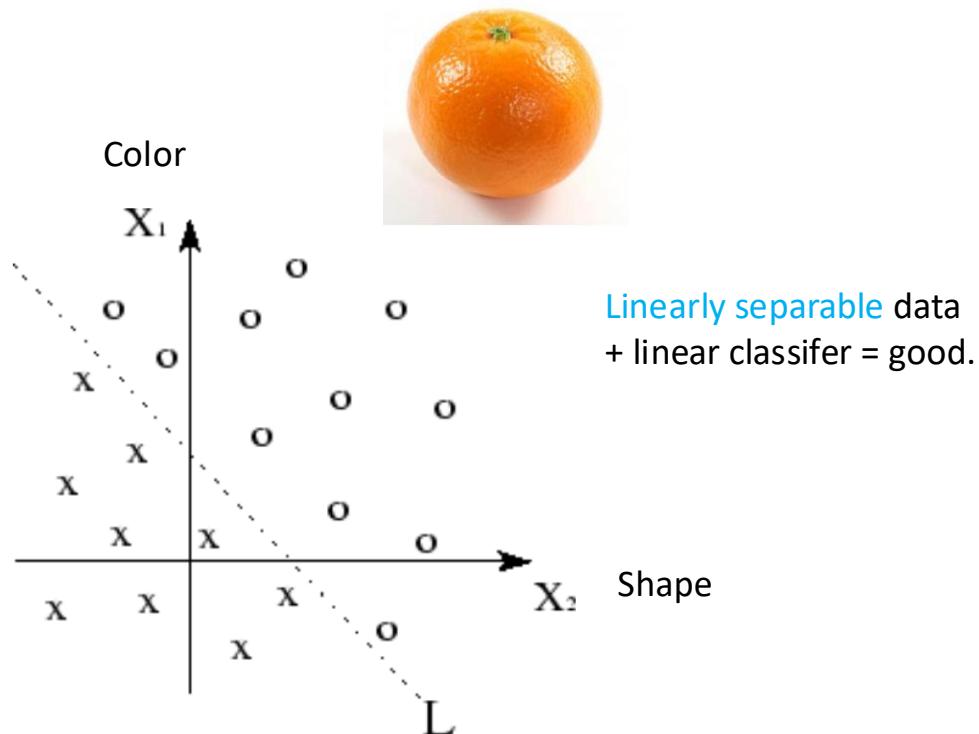
## Image Classification

- Problem Setting and Challenges
- Bag of words
- Nearest Neighbors
- Hyperparameters and Training
- **Linear Classifier**
- SVMs
- Classification Metrics and Performance Evaluation

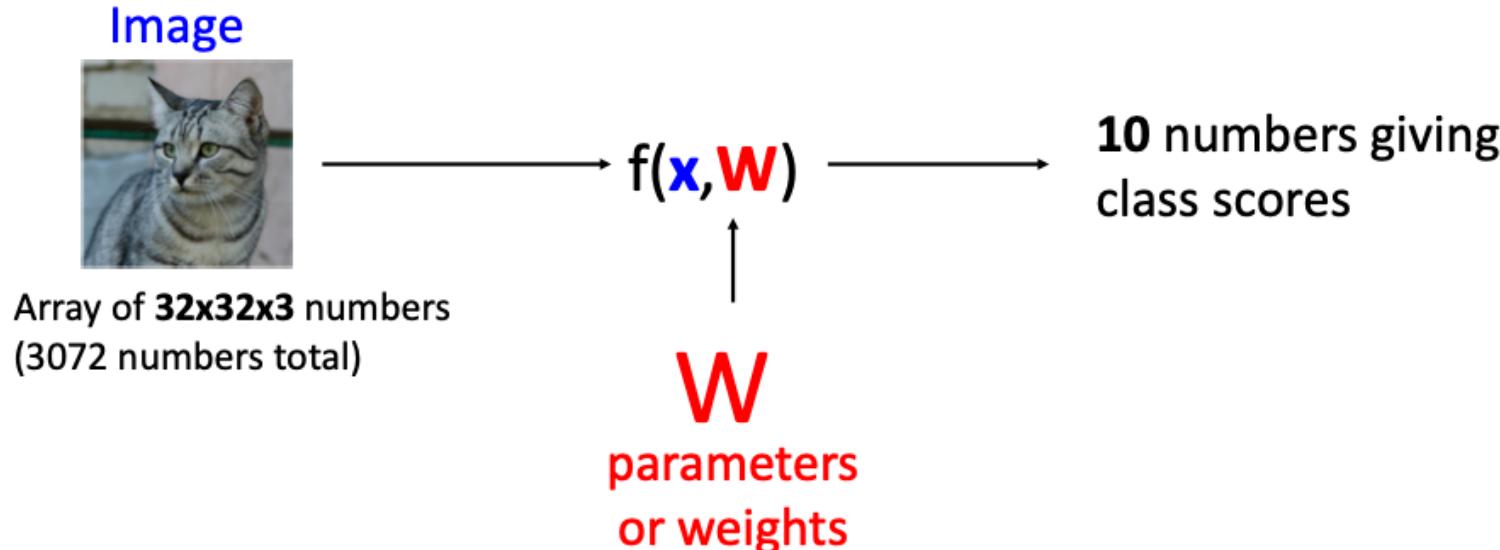
# Apple – Orange Classification



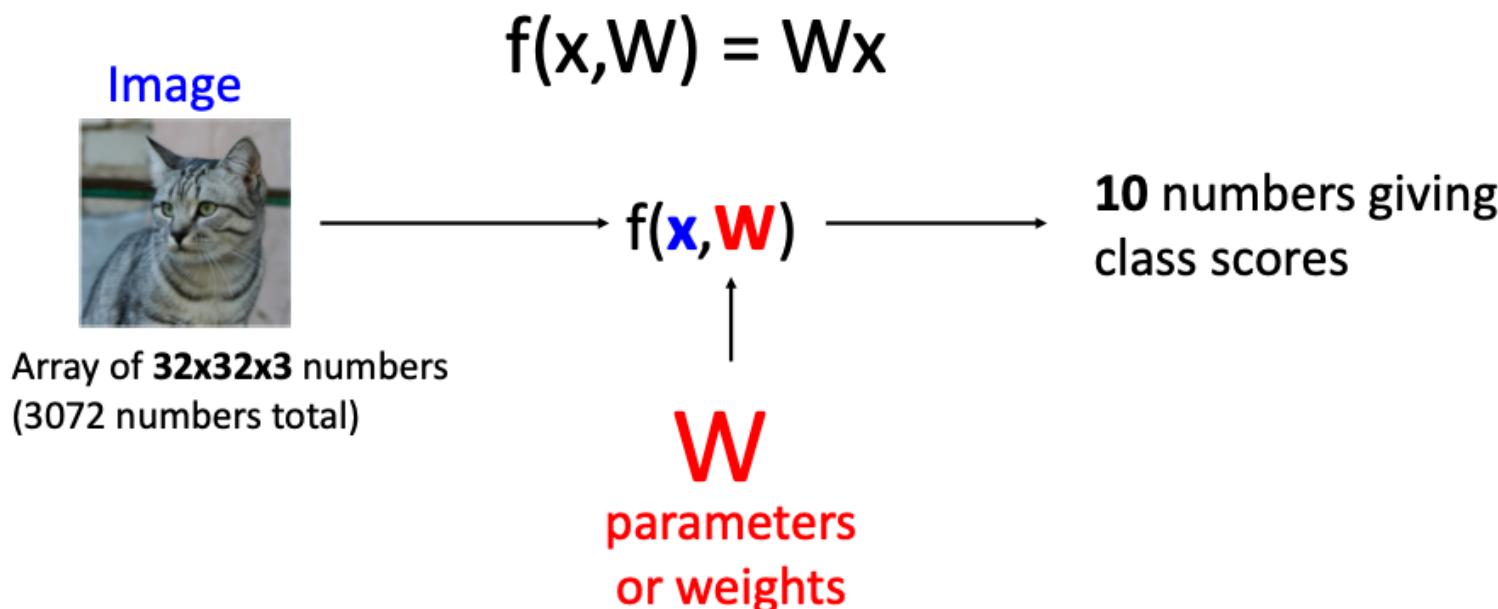
# Feature Spaces - Separability



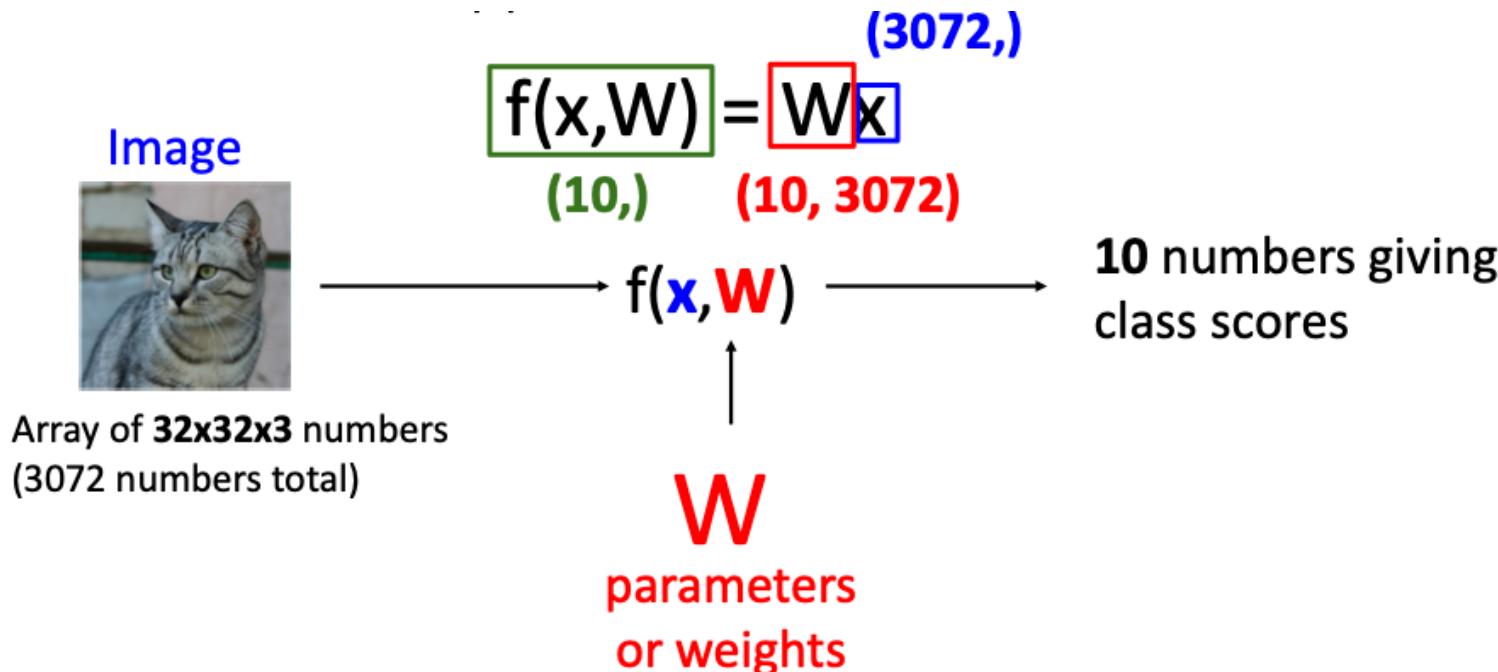
# Parametric Approach: Linear Classifier



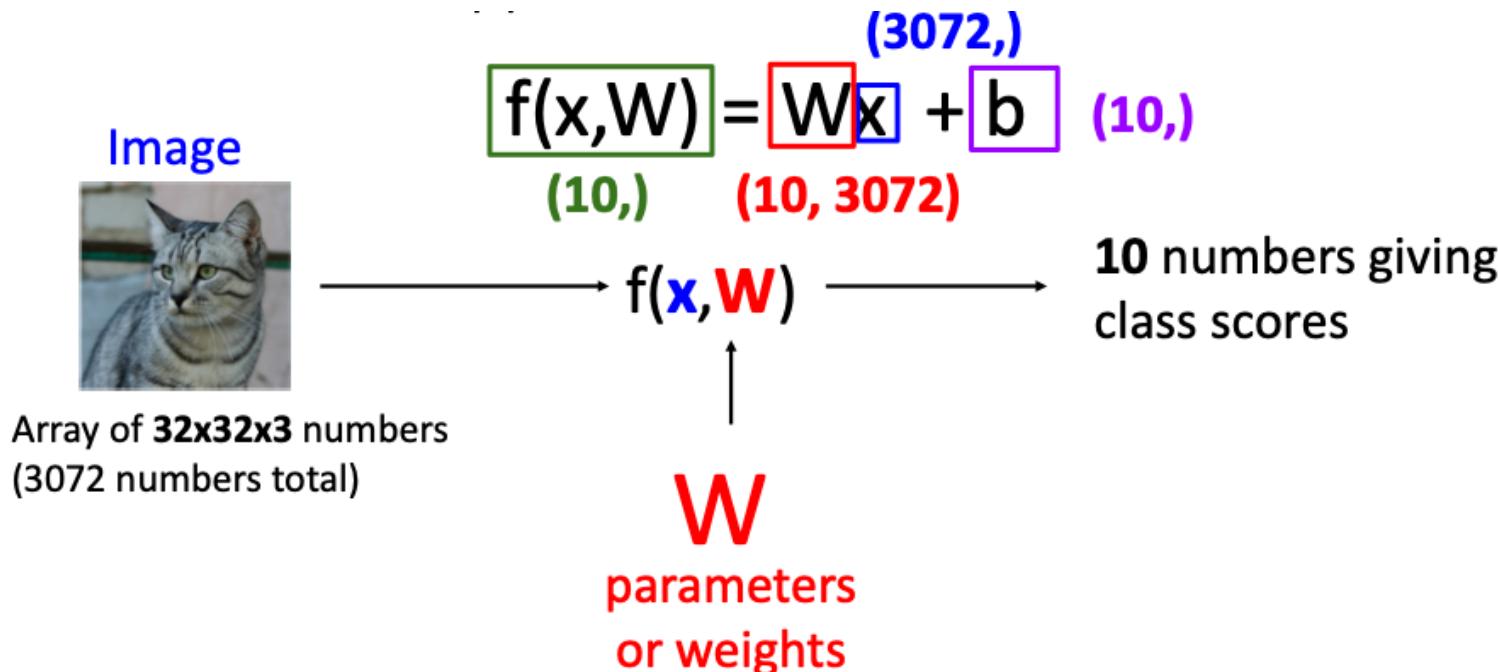
# Parametric Approach: Linear Classifier



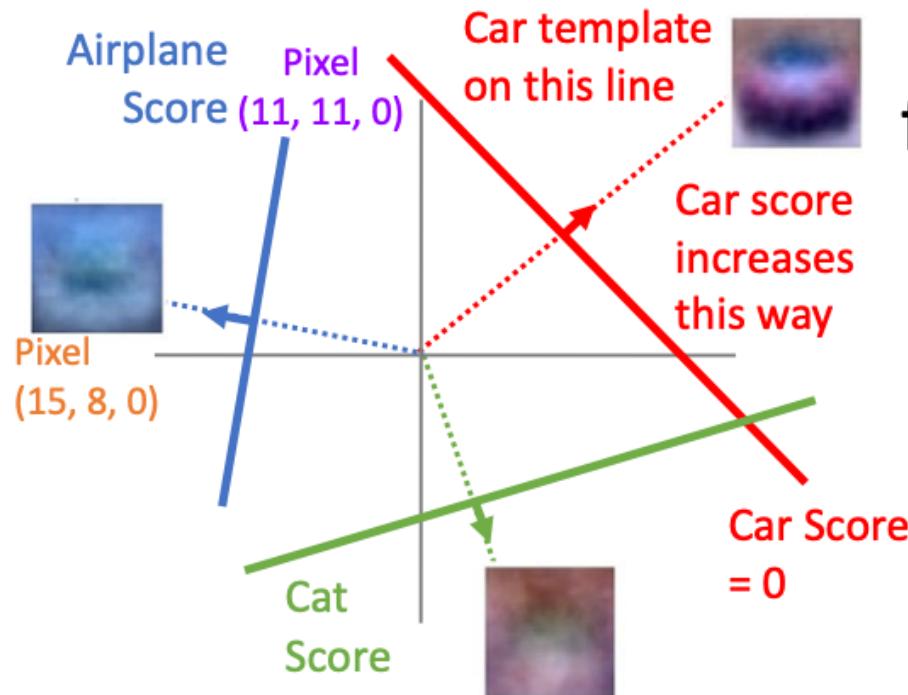
# Parametric Approach: Linear Classifier



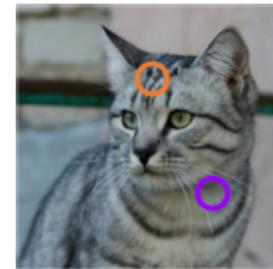
# Parametric Approach: Linear Classifier



# Parametric Approach: Linear Classifier

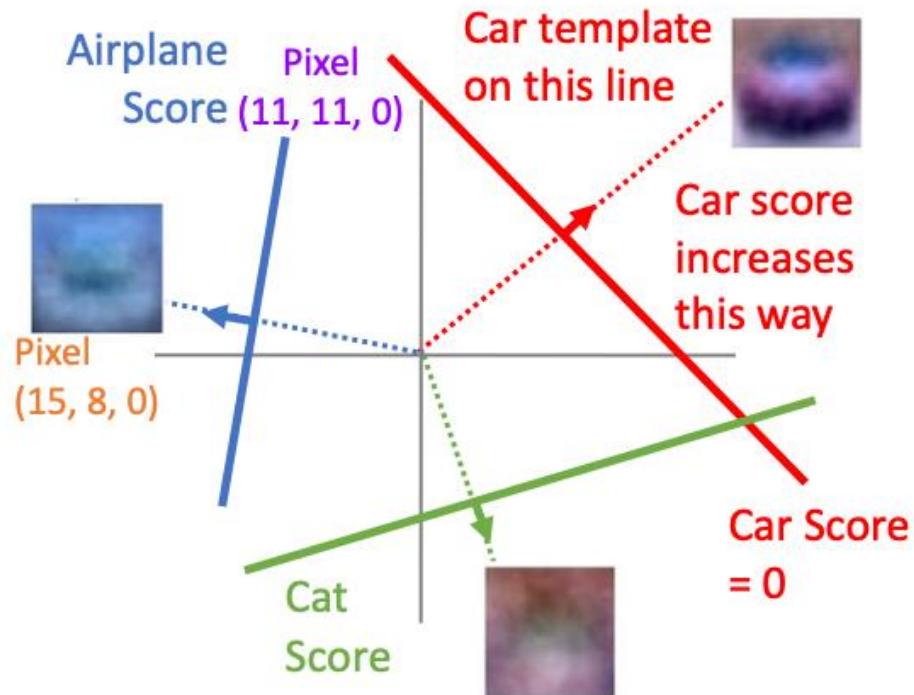


$$f(x, W) = Wx + b$$

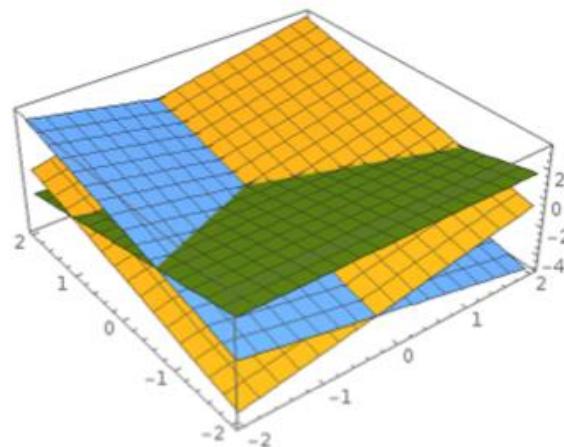


Array of **32x32x3** numbers  
(3072 numbers total)

# Parametric Approach: Linear Classifier



Hyperplanes carving up a high-dimensional space



Plot created using [Wolfram Cloud](#)

# Today's Agenda

---

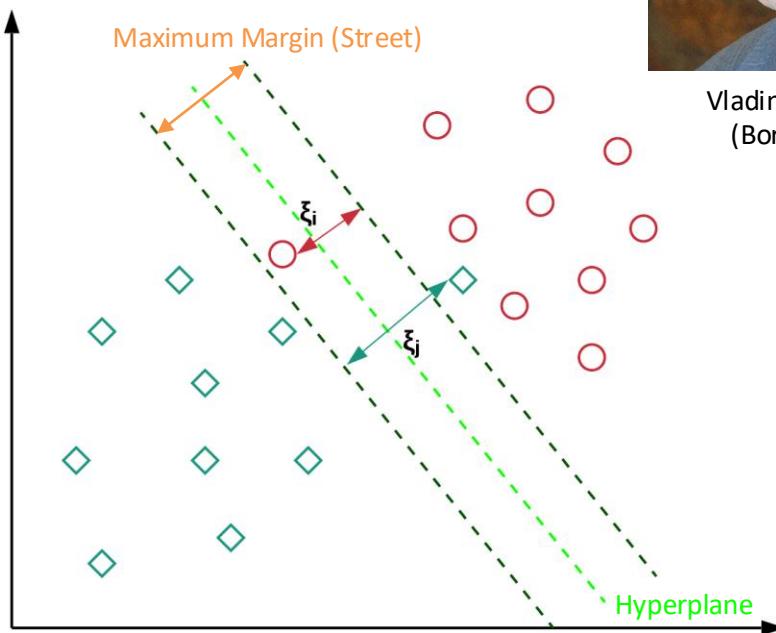
## Image Classification

- Problem Setting and Challenges
- Bag of words
- Nearest Neighbors
- Hyperparameters and Training
- Linear Classifier
- **SVMs**
- Classification Metrics and Performance Evaluation

# Support Vector Machines (SVMs)

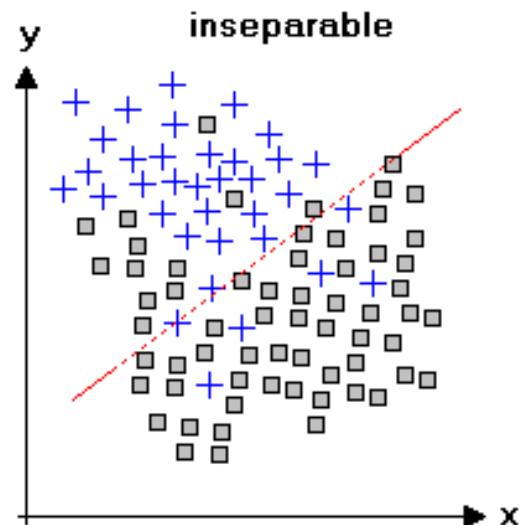
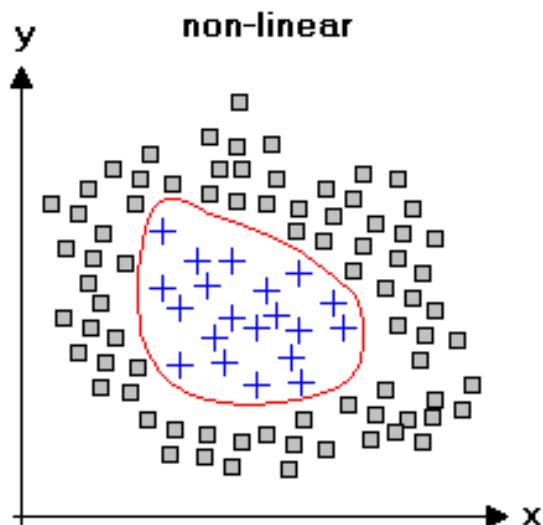
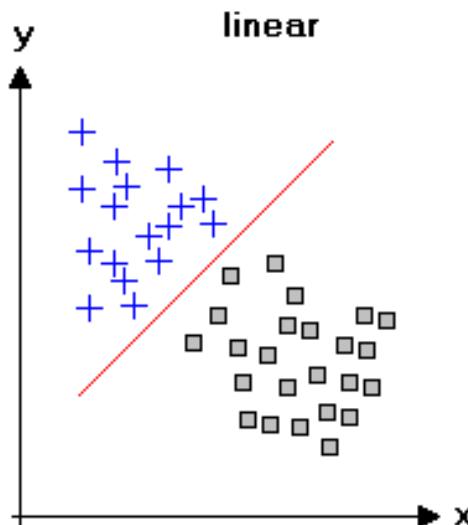
- Goal: Find separating hyperplane (1D: point, 2D: line, 3D: plane, etc.)
- Maximizes margin (street) between classes
- Support vectors are data points which are closest to the decision boundary
- Only support vectors define the decision boundary (no other data points)
- Support vectors are found via optimization if linear separation exists (**Hard Margin SVM**)
- **Soft Margin SVMs** allow for outliers by additionally minimizing slack variables  $\xi_i$  that shift separation boundary
- Extension to non-linear decision boundaries via kernel trick (i.e. apply a non-linear transformation before SVM fitting).

**SVMs work very well and are quite robust to noisy data!**



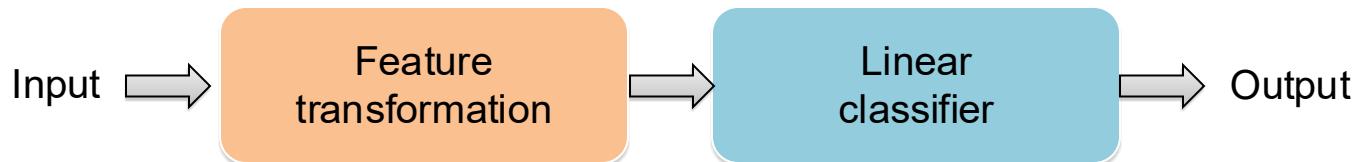
Vladimir Vapnik  
(Born 1936)

# Feature Spaces - Separability

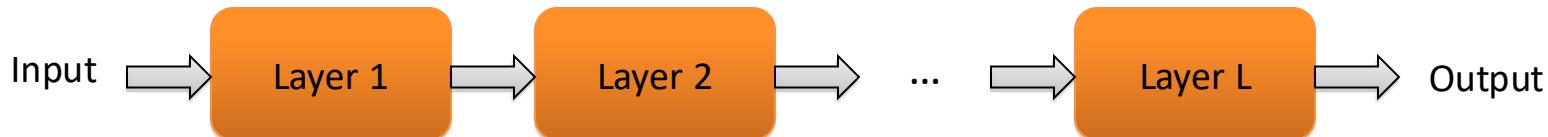


# From linear to nonlinear classifiers

- To achieve good accuracy on challenging problems, we need to be able to train *nonlinear* models
- Two strategies for making nonlinear predictors out of linear ones:
  - **“Shallow” approach:** nonlinear feature transformation followed by linear classifier
  - **“Deep” approach:** stack multiple layers of linear predictors (interspersed with nonlinearities)

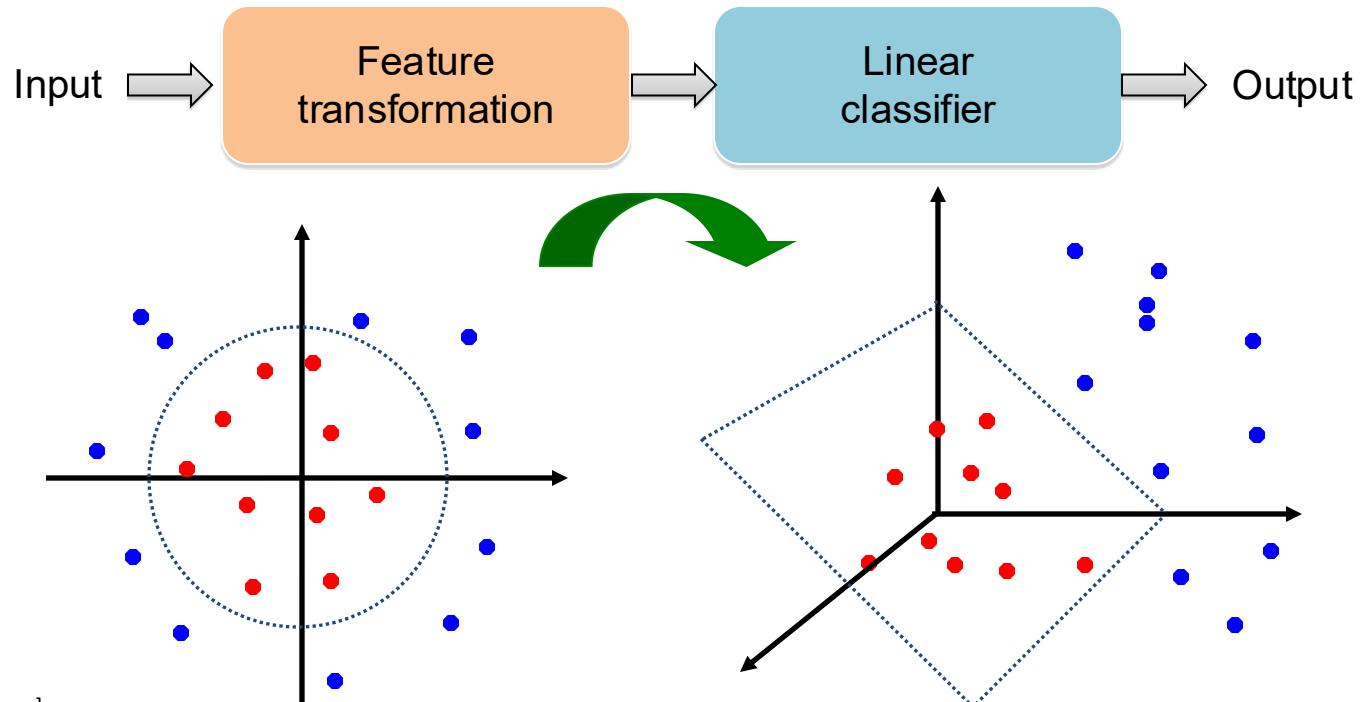


– **“Deep” approach:** stack multiple layers of linear predictors (interspersed with nonlinearities)



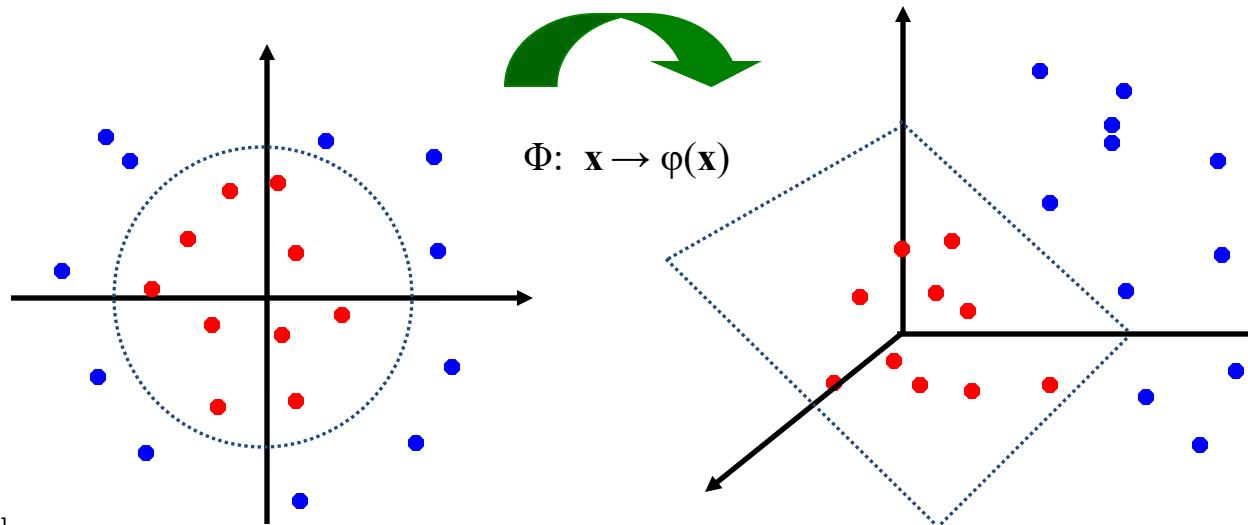
# Shallow approach: Feature Transform

Idea: Lift the problem to another space where the data becomes linearly separable.  
(typically with higher dimension)



# Nonlinear SVMs

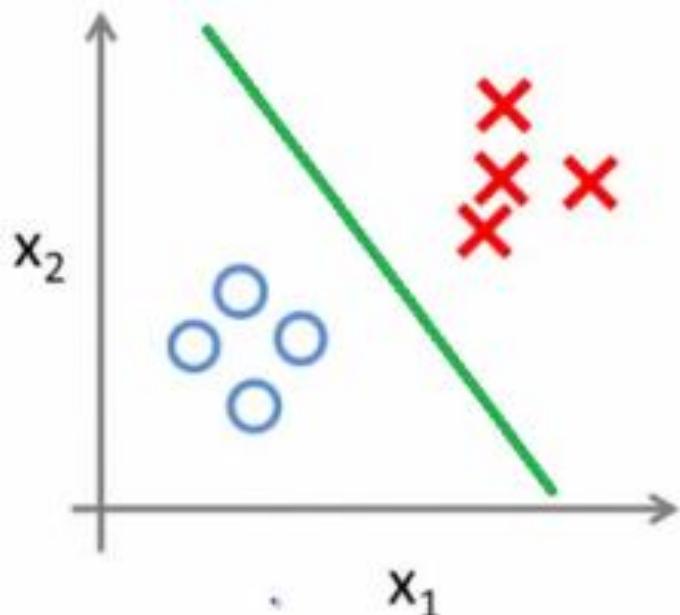
- General idea: map the original feature space to a higher-dimensional one where the training data is (hopefully) separable
  - Because of the special properties of SVM optimization, this can be done without explicitly performing the lifting transformation



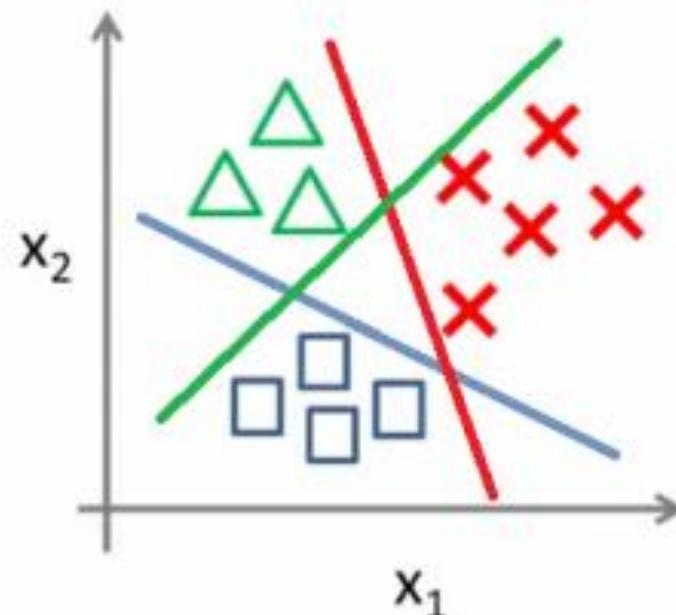
[Image credit: Andrew Moore]

# Multi-class Classification

Binary classification:



Multi-class classification:



- Simple multi-class extension: series of binary classification - one class against all other.

# Today's Agenda

---

## Image Classification

- Problem Setting and Challenges
- Bag of words
- Nearest Neighbors
- Hyperparameters and Training
- Linear Classifier
- SVMs
- **Classification Metrics and Performance Evaluation**

# Classification - Performance Evaluation

- Accuracy:

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FP} + \text{TP} + \text{FN}}$$

- Precision and Recall:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F1 score: geometric mean of precision and recall

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

# Precision and Recall

How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

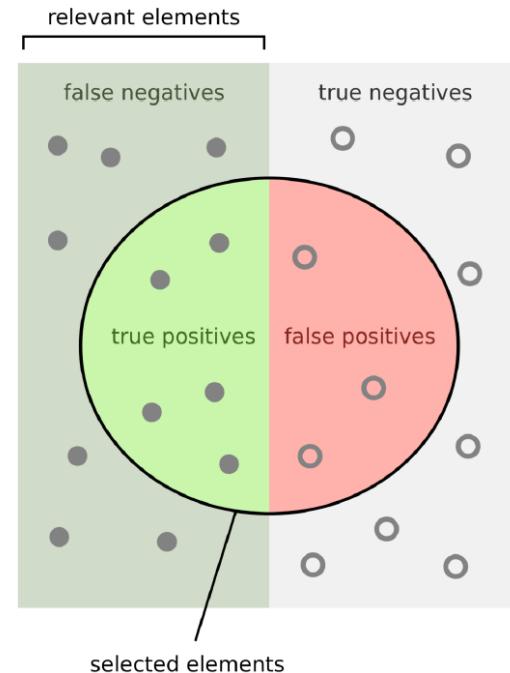


How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$



Combine P+R into: F1-Measure



# Accuracy

- Accuracy:

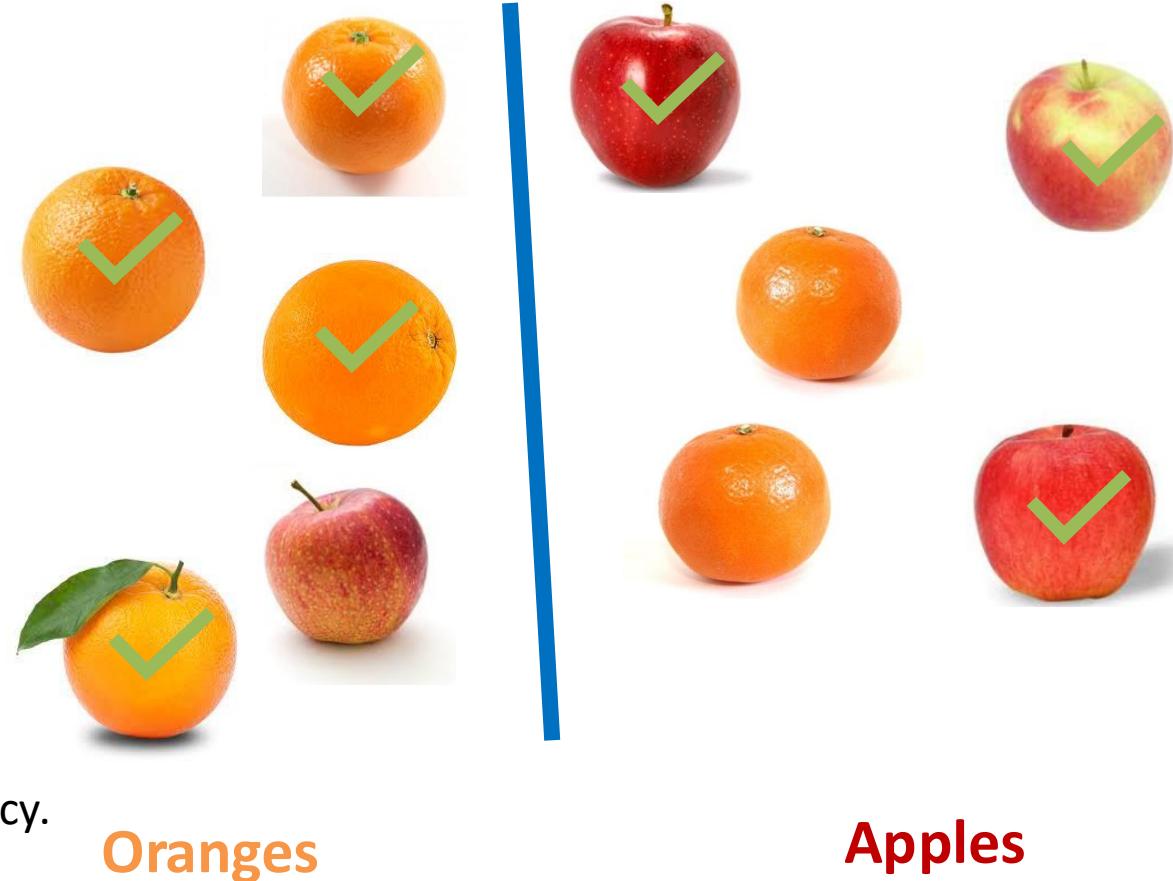
$$= \frac{\text{total correct}}{\text{total observations}}$$

$$= \frac{7}{10} = 70\%$$

- Problem with imbalanced classes, e.g.:

990 oranges & 10 apples

Model that predicts always  
oranges then has 99 % accuracy.

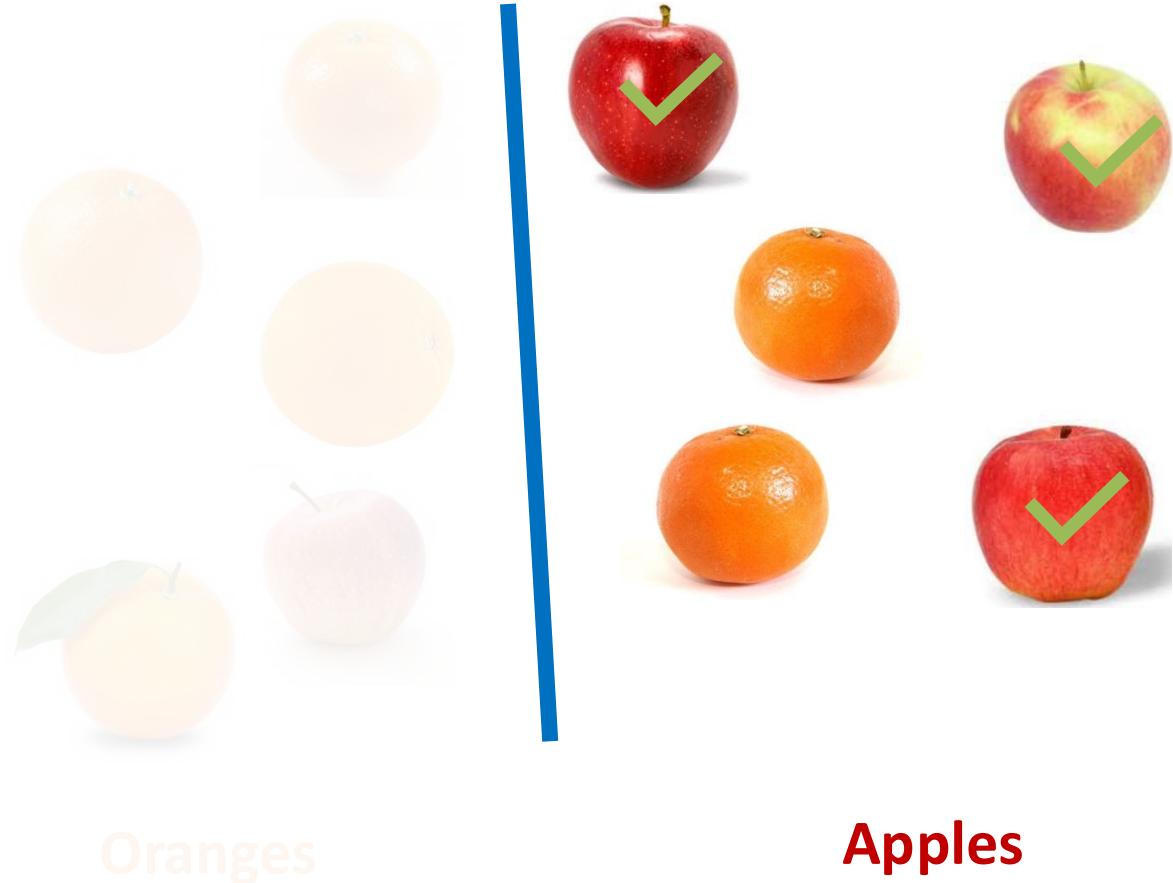


# Precision

- Precision for apples

$$= \frac{\text{total apples correct}}{\text{total apple side observations}}$$

$$= \frac{3}{5} = 60\%$$

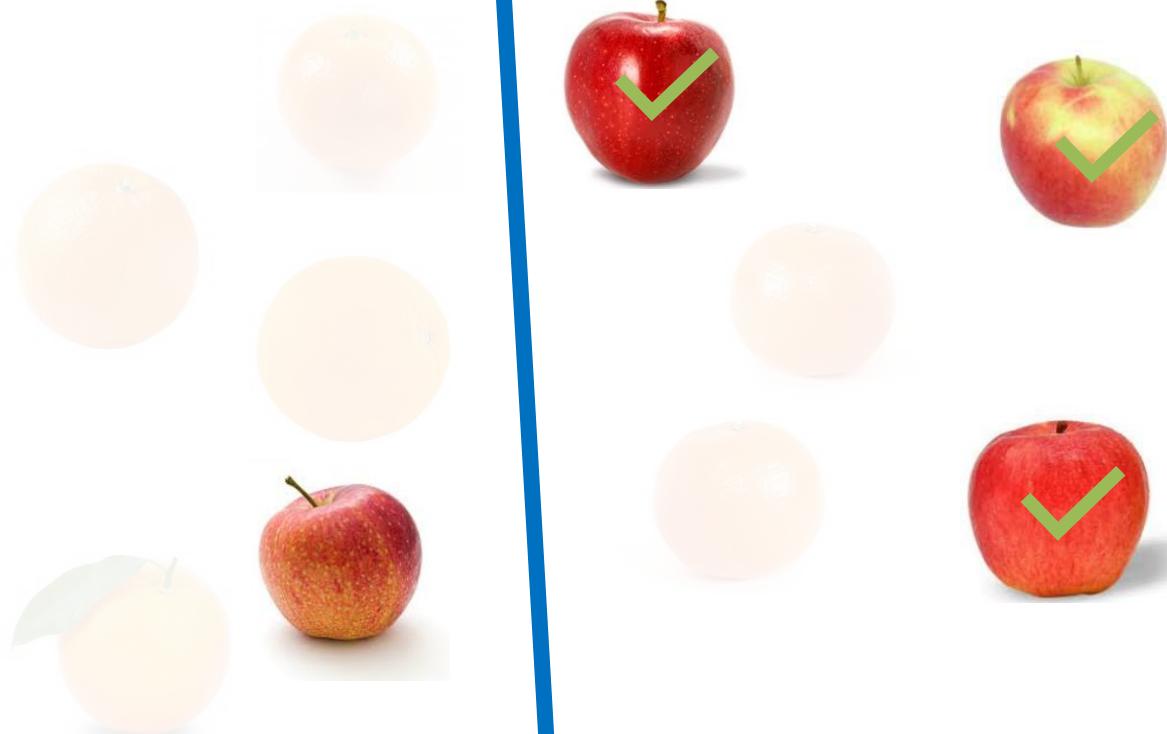


# Recall

- Recall for apples

$$= \frac{\text{total apples correct}}{\text{all actual apples}}$$

$$= \frac{3}{4} = 75\%$$



Oranges

Apples

# Change Decision Boundary

- Precision for apples

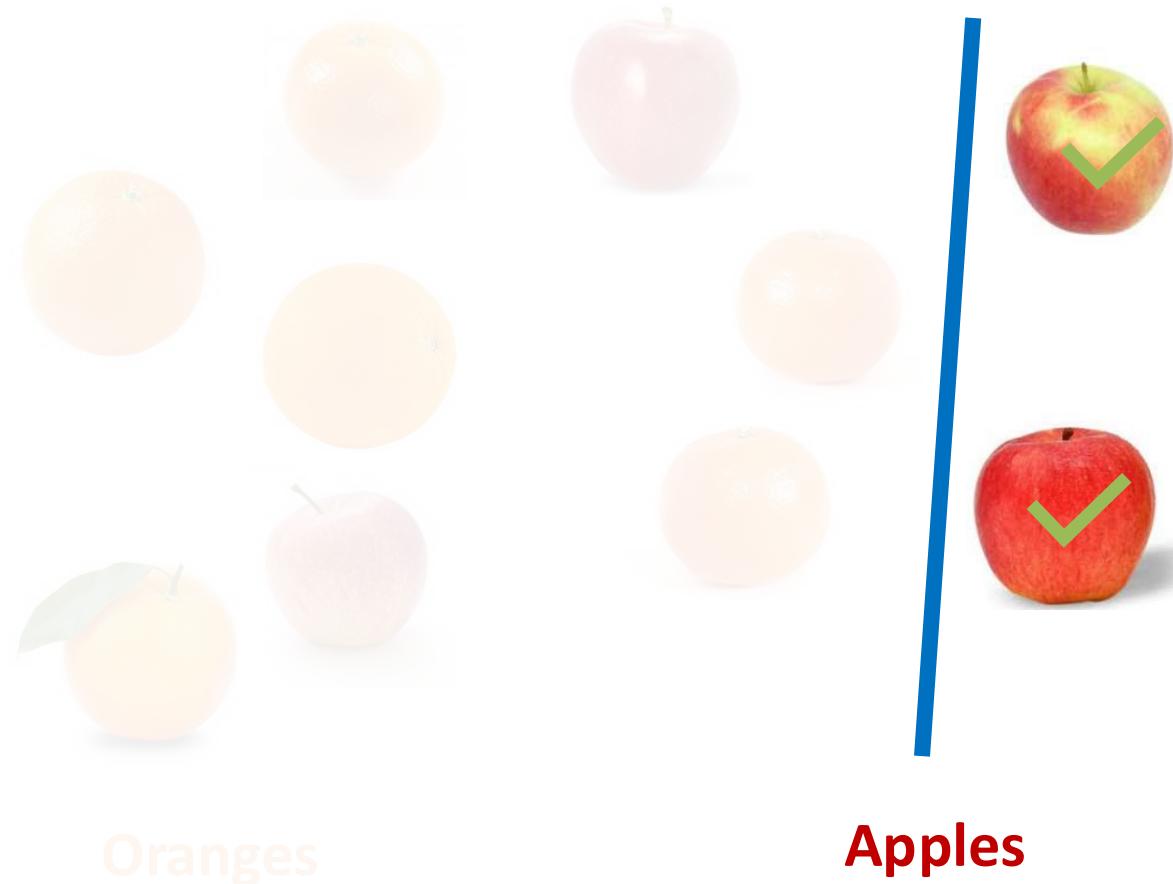
$$= \frac{\text{total apples correct}}{\text{total apple side observations}}$$

$$= \frac{2}{2} = 100\%$$

- Recall for apples

$$= \frac{\text{total apples correct}}{\text{all actual apples}}$$

$$= \frac{2}{4} = 50\%$$



# Change Decision Boundary

- Precision for apples

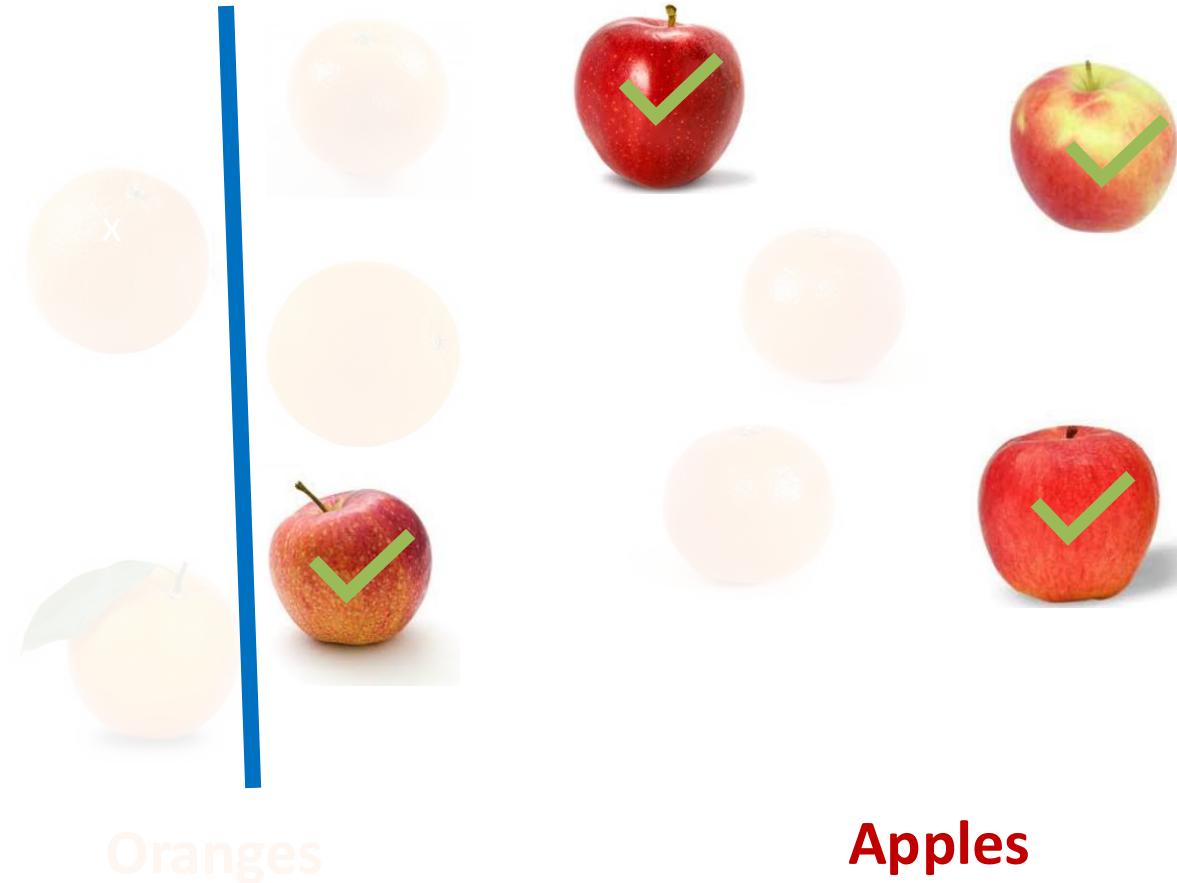
$$= \frac{\text{total apples correct}}{\text{total apple side observations}}$$

$$= \frac{4}{8} = 50\%$$

- Recall for apples

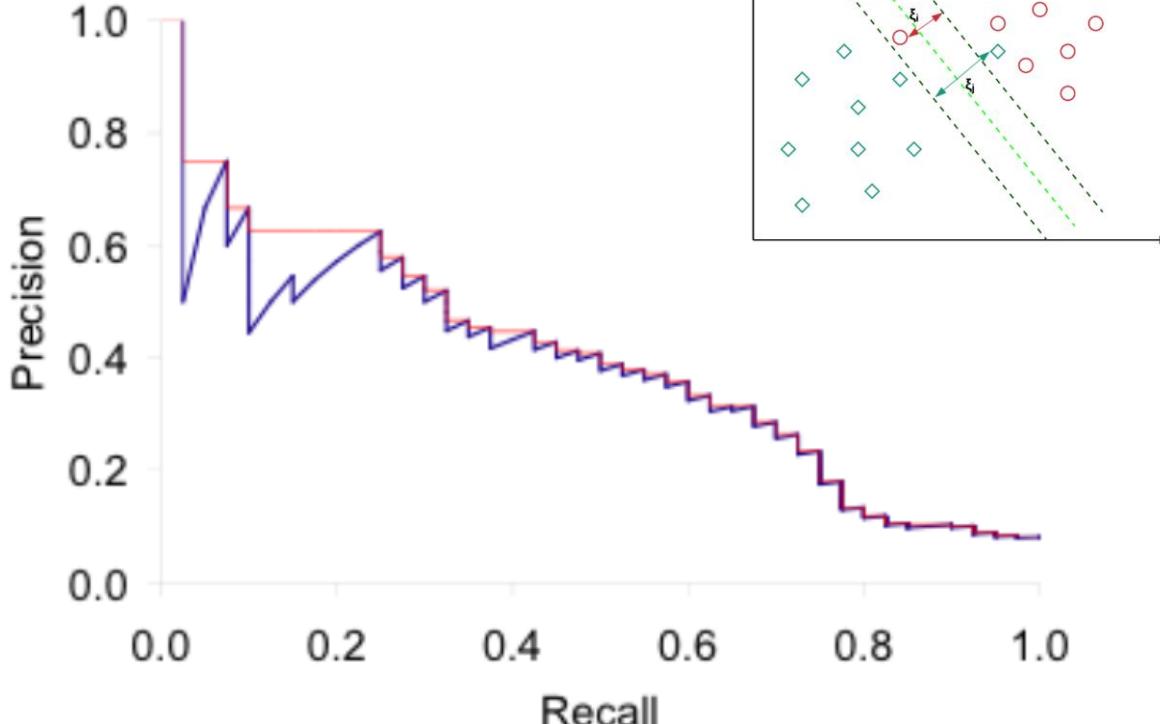
$$= \frac{\text{total apples correct}}{\text{all actual apples}}$$

$$= \frac{4}{4} = 100\%$$



# Precision-Recall Curve

- Trade-off Precision and Recall



# More Resources

Papers with code <https://paperswithcode.com/datasets?task=image-classification>

177 dataset results for **Image Classification**

Search for datasets

Best match

**Filter by Modality**

Images	139
Texts	13
Medical	11
Biomedical	4
Graphs	4
Hyperspectral images	3
Videos	3
Audio	2

**Filter by Task (clear)**

**Image Classification**  x

Question Answering	292
Semantic Segmentation	221
Object Detection	210
Speech Recognition	180
Language Modelling	130
Text Classification	102

**CIFAR-10**  
 The CIFAR-10 dataset (Canadian Institute for Advanced Research, 10 classes) is a subset of the Tiny Images dataset and consists of 60000 32x32 color images. The images are labelled with...  
 9,361 PAPERS • 58 BENCHMARKS

---

**ImageNet**  
 The ImageNet dataset contains 14,197,122 annotated images according to the WordNet hierarchy. Since 2010 the dataset is used in the ImageNet Large Scale Visual Recognition Challenge...  
 8,932 PAPERS • 86 BENCHMARKS

---

**MNIST**  
 The MNIST database (Modified National Institute of Standards and Technology database) is a large collection of handwritten digits. It has a training set of 60,000 examples, and a test set of...  
 5,506 PAPERS • 43 BENCHMARKS

---

**CIFAR-100**  
 The CIFAR-100 dataset (Canadian Institute for Advanced Research, 100 classes) is a subset of the Tiny Images dataset and consists of 60000 32x32 color images. The 100 classes in the...  
 4,612 PAPERS • 34 BENCHMARKS

---

**SVHN (Street View House Numbers)**  
 Street View House Numbers (SVHN) is a digit classification benchmark dataset that contains 600,000 32x32 RGB images of printed digits (from 0 to 9) cropped from pictures of house nu...  
 2,254 PAPERS • 10 BENCHMARKS

---

**CelebA (CelebFaces Attributes Dataset)**  
 CelebFaces Attributes dataset contains 202,599 face images of the size 178×218 from 10,177 celebrities, each annotated with 40 binary labels indicating facial attributes like hair color, gen...  
 2,186 PAPERS • 15 BENCHMARKS

---

**Fashion-MNIST**  
 Fashion-MNIST is a dataset comprising of 28×28 grayscale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and th...  
 1,925 PAPERS • 12 BENCHMARKS

# More Resources

Google Vision API <https://www.youtube.com/watch?v=9FMVZkxfrfA>

The screenshot shows a Jupyter Notebook interface with the title "Google Vision API". The notebook has a "Steps:" section containing the following numbered instructions:

1. Visit <https://console.cloud.google.com/>
2. Create a new project if none exist (otherwise you can use an existing one)
3. Click "Go to APIs Overview" -> Click "Enable APIs and Services" -> Enable the "Cloud Vision API"
4. Click the "Credentials" tab on the left. Click "+ CREATE CREDENTIALS" at the top and choose "Service Account". Give the service account a name and click "Create"
5. Click on the newly created service account, ensure it is enabled, and click "ADD KEY" -> "Create new key". Pick "JSON" and download the json file and store it in the current working directory.
6. Run the cells in this notebook.

In [4]:

```
pip install google-cloud-vision
```

Requirement already satisfied: google-cloud-vision in c:\users\ritvik\anaconda3\lib\site-packages (1.0.0)  
Requirement already satisfied: google-api-core[grpc]<2.0.0dev,>=1.14.0 in c:\users\ritvik\anaconda3\lib\site-packages (from google-cloud-vision) (1.22.0)

---

# Questions?

# Disclaimer

---

Many of the slides used here are obtained from online resources (including many open lecture materials) without appropriate acknowledgement. They are used here for the sole purpose of classroom teaching. All the credit and all the copyrights belong to the original authors. You should not copy it, redistribute it, put it online, or use it for any other purposes than for this course.