中山大學
SUN YAT-SEN UNIVERSITY

Lecture 12.
Visual Recognition

Pattern Recognition and Computer Vision

Guanbin Li,
School of Computer Science and Engineering, Sun Yat-Sen University

扫码签到

What we will learn today?

- Introduction to object recognition
- K-nearest neighbor algorithm
- Naïve Bayes Classification
- A simple Object Recognition pipeline

What do we mean by learning based vision or ‘semantic vision’?

What are the different visual recognition tasks?

Classification

- Does this image contain a building? [yes/no]

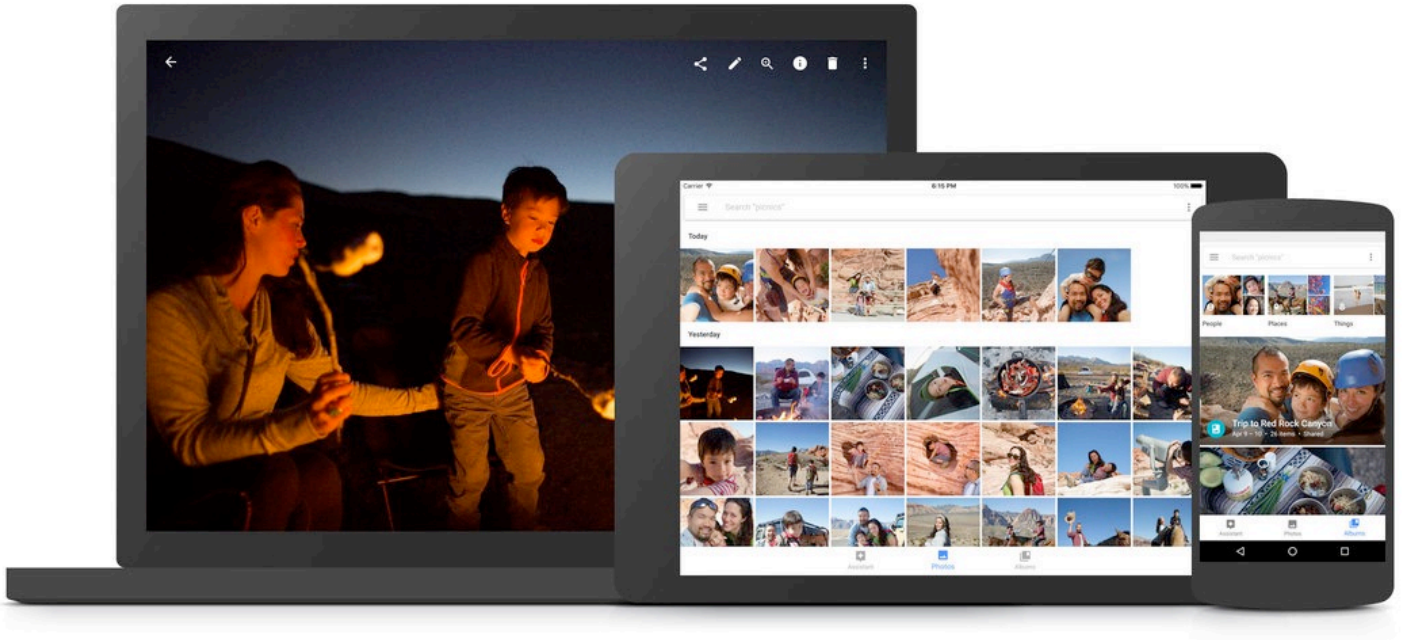
Classification

- Is this a beach?

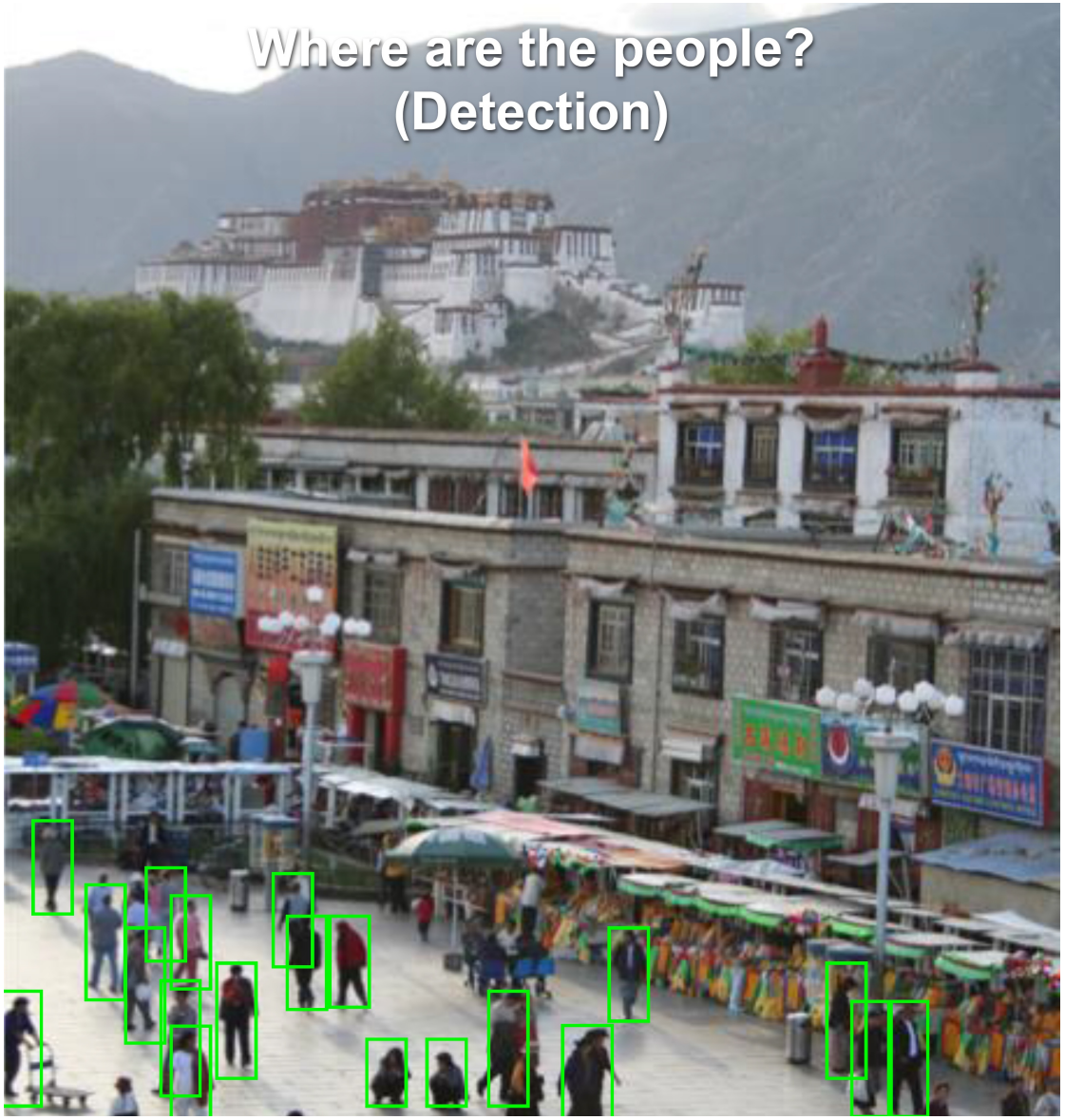
Image search

Image search

- Organizing photo collections



Detection



Detection

Where are the faces? (Detection)

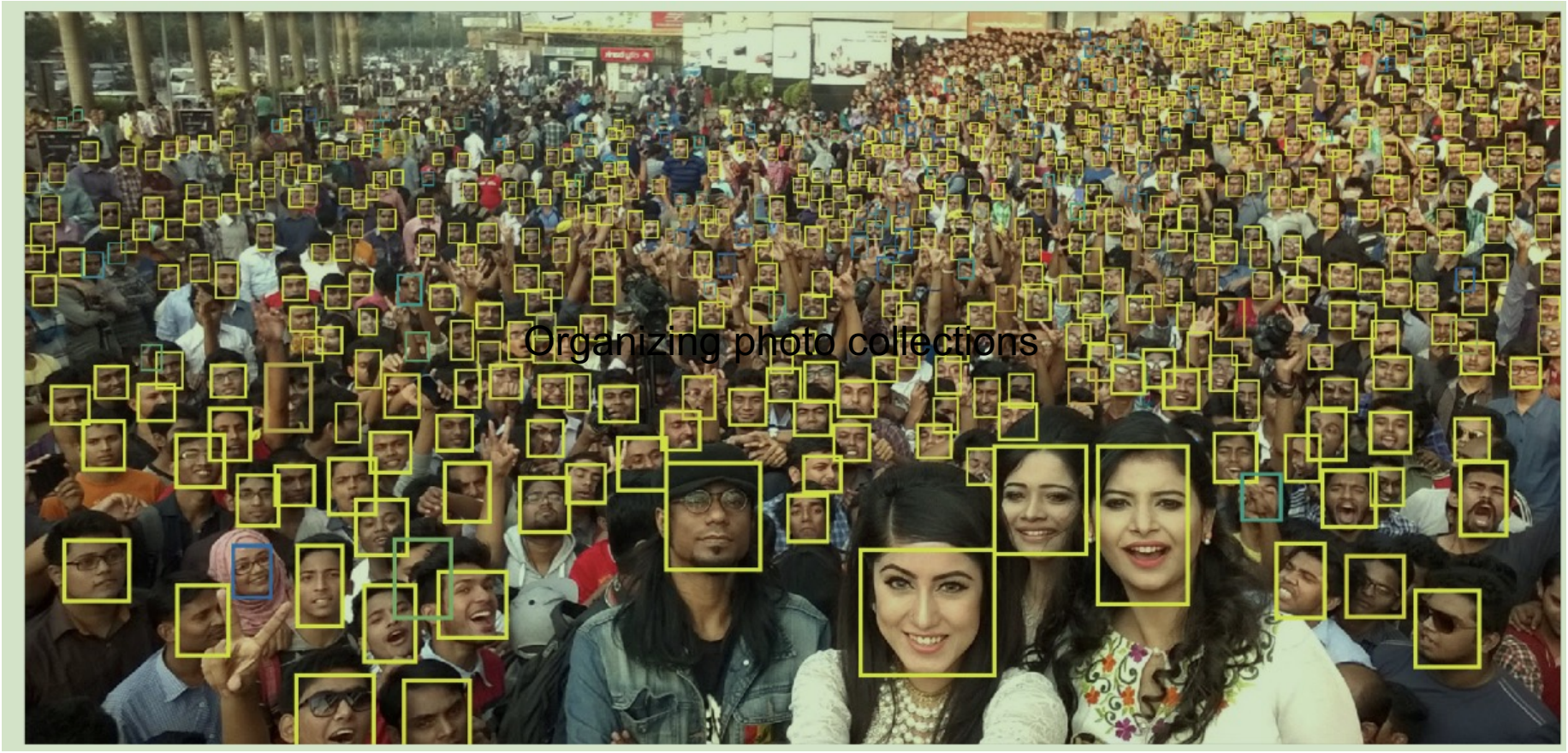
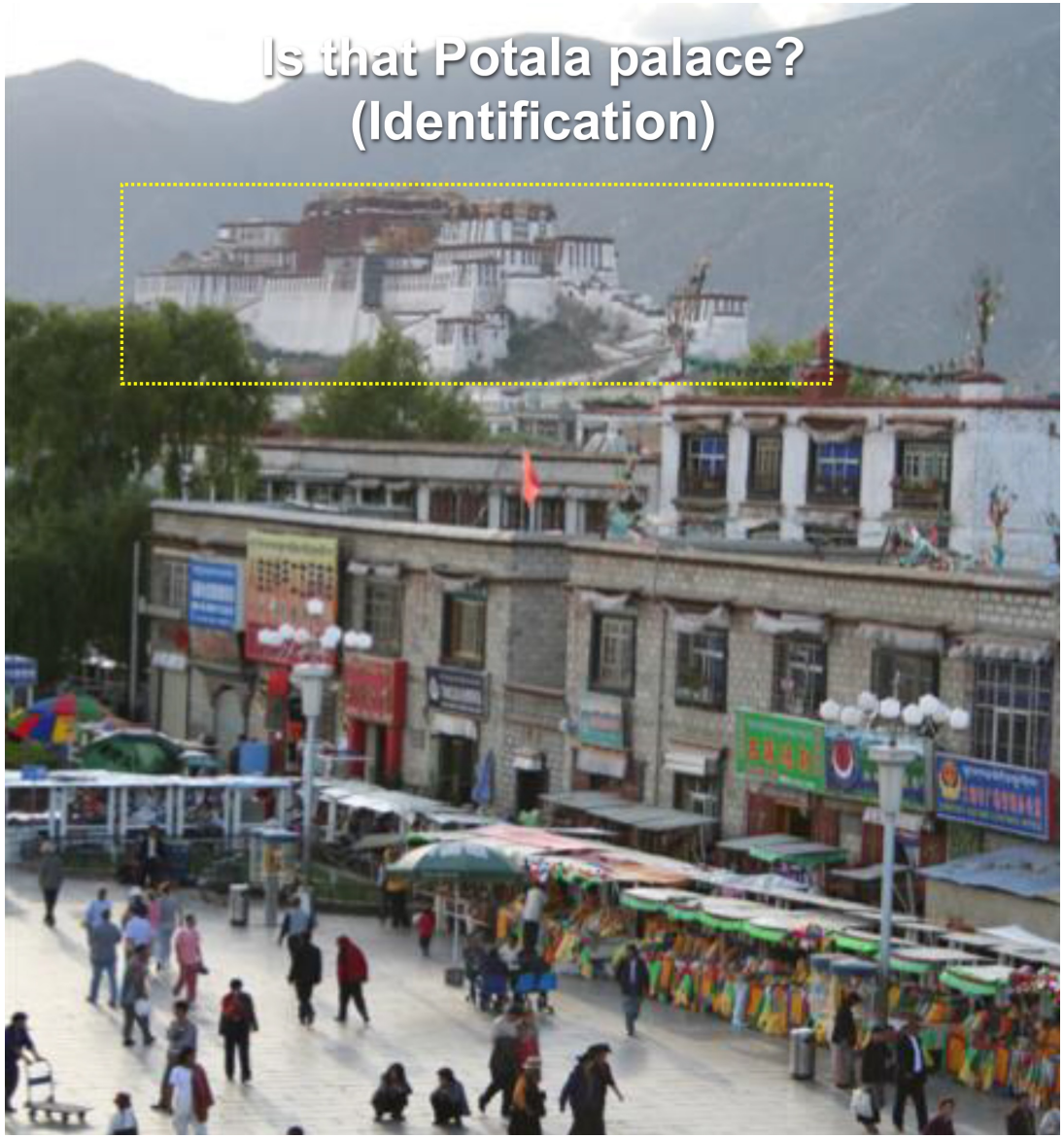
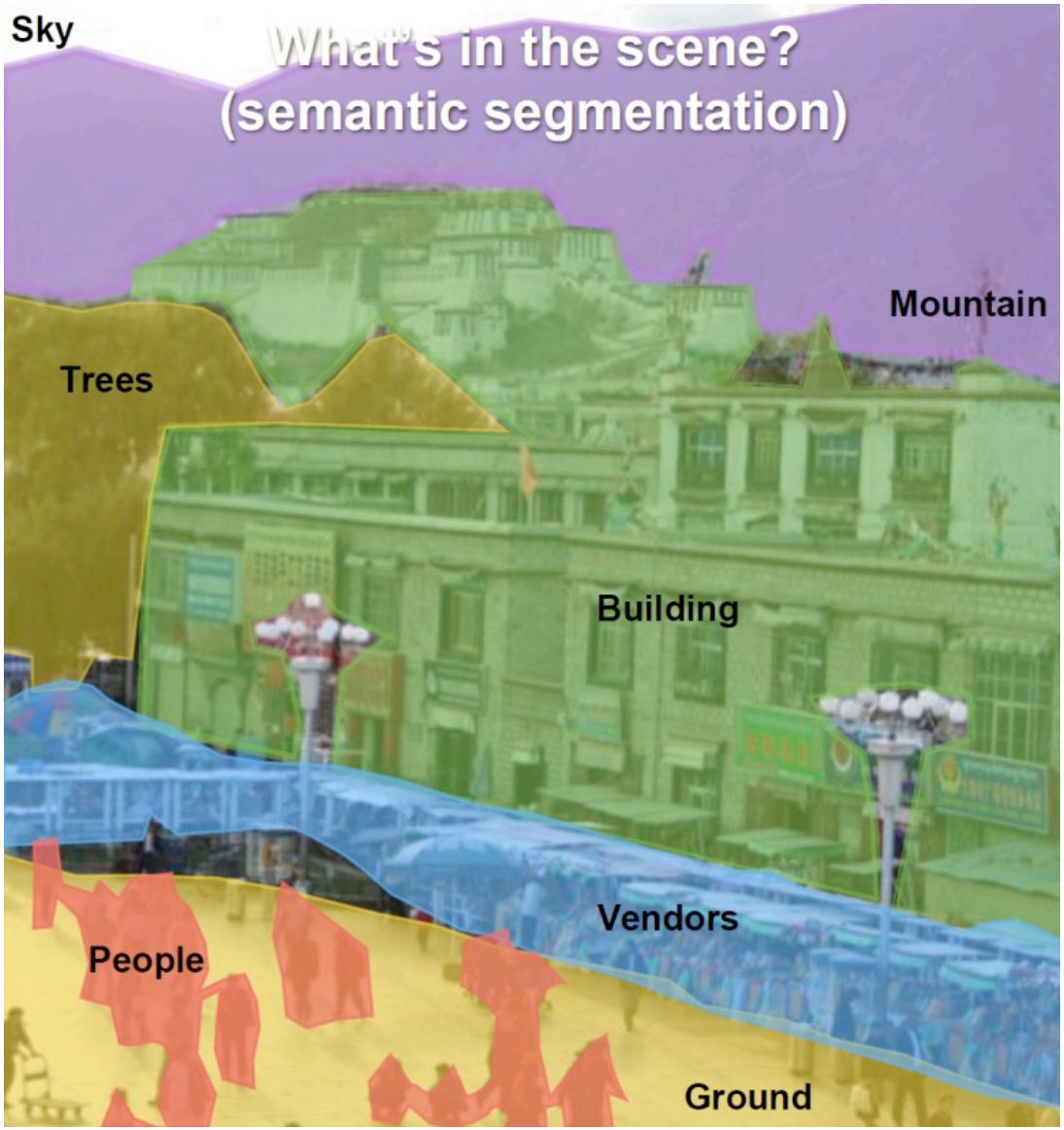


Image Identification

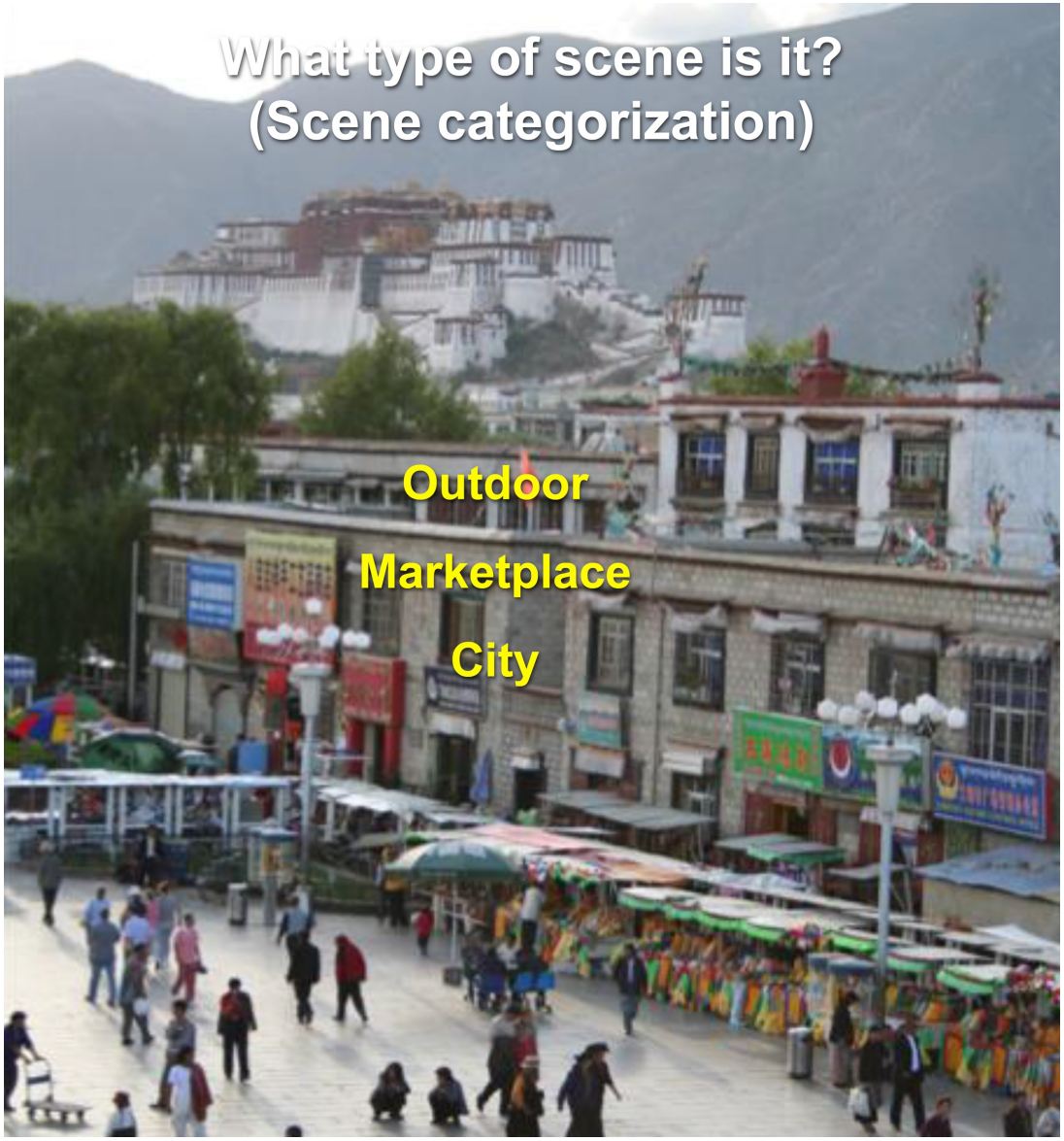


Semantic Segmentation



Object Categorization

Scene Classification



Activity or Event recognition

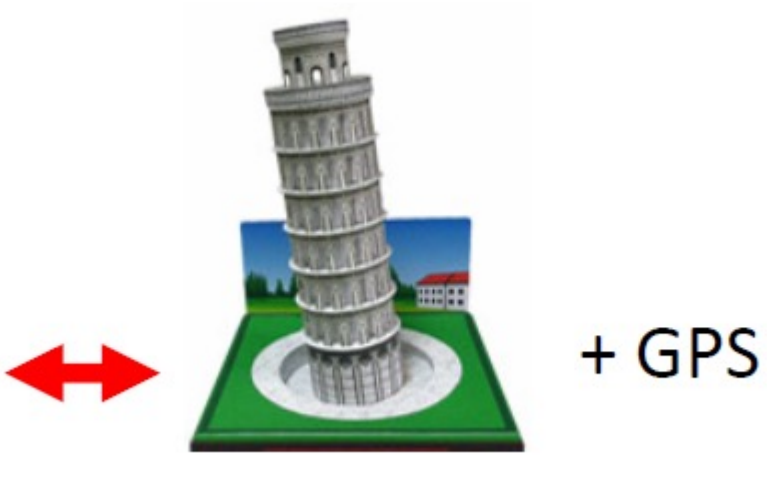
- What are these people doing?

Categorization vs Single instance recognition

- Where is the SKITTLES?

Applications of computer vision

- Recognizing landmarks in mobile platforms



Applications of computer vision

Autonomous Vehicle

Applications of computer vision

Autonomous Checkout

Visual Recognition

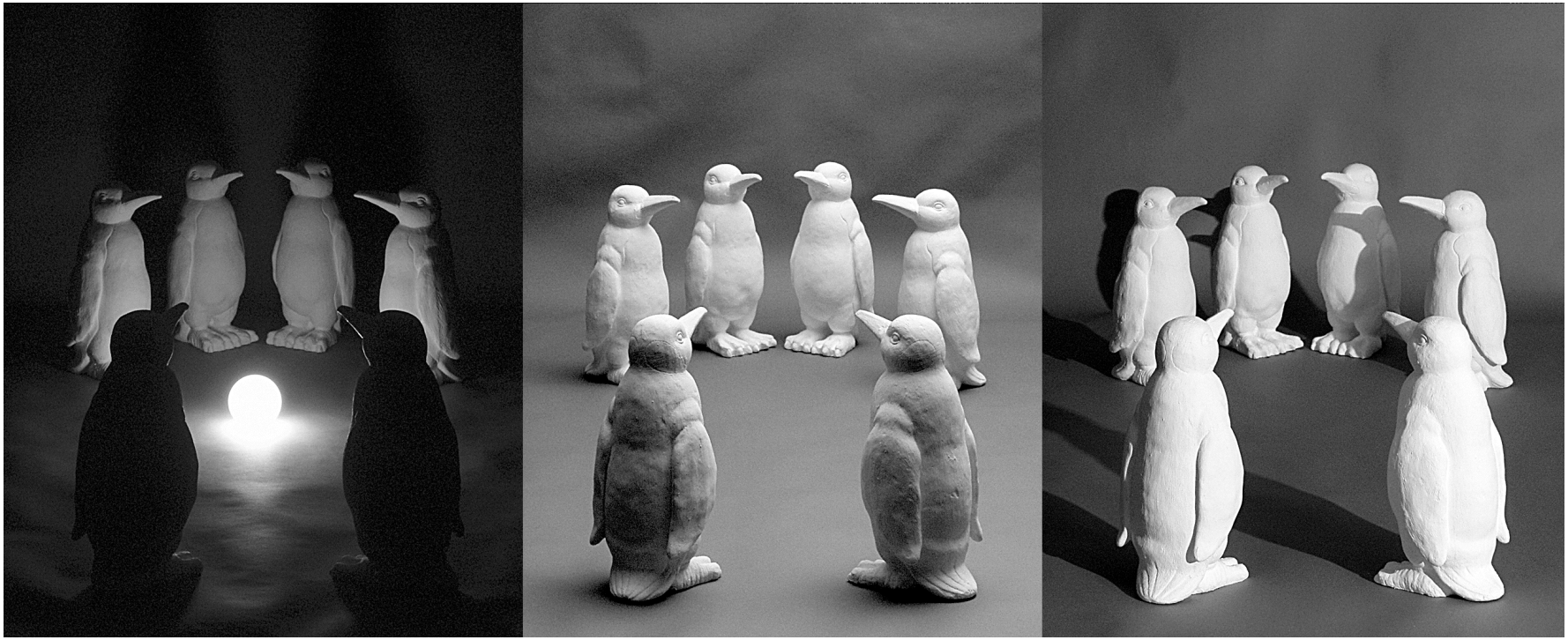
- Design algorithms that have the capability to:
  - – Classify images or videos
  - – Detect and localize objects
  - – Estimate semantic and geometrical attributes
  - – Classify human activities and events

Why is this challenging?

How many object categories are there?

Challenges: viewpoint variation

Challenges: illumination

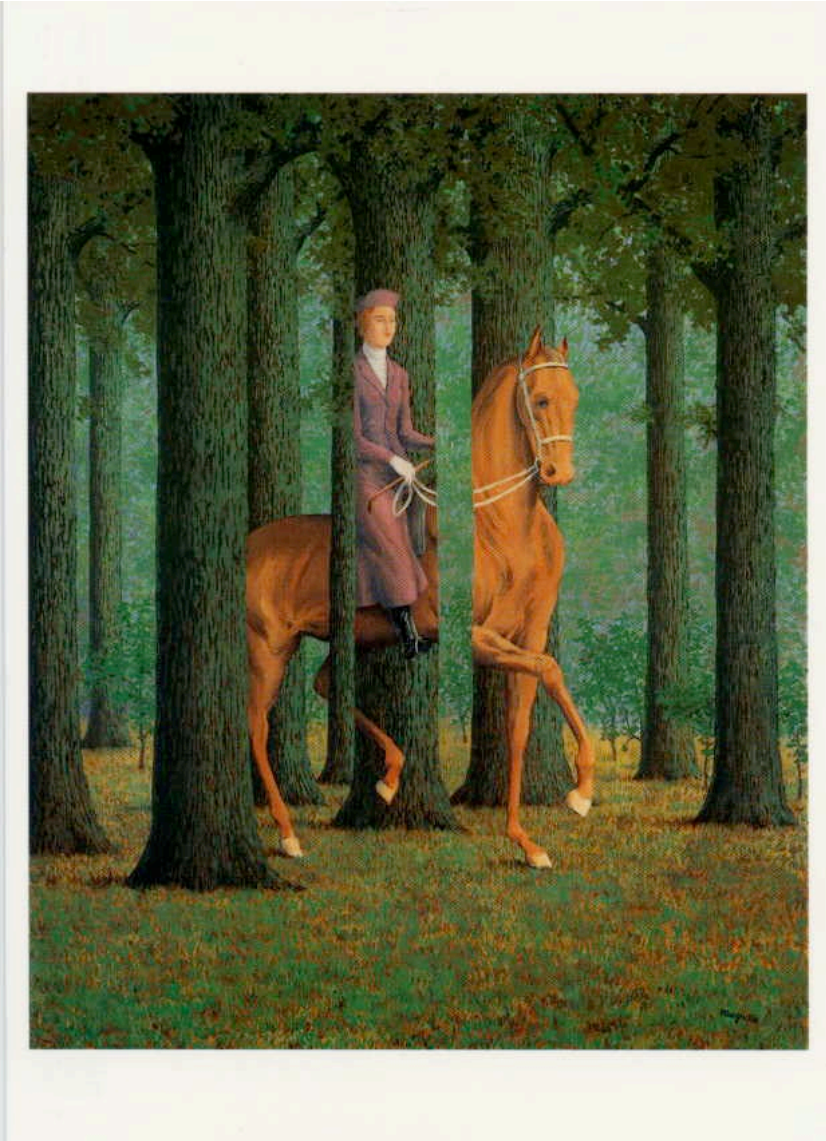


Challenges: scale

Challenges: deformation

Challenges: deformation

Challenges: occlusion



Challenges: background clutter

Challenges: background clutter

Challenges: intra-class variation

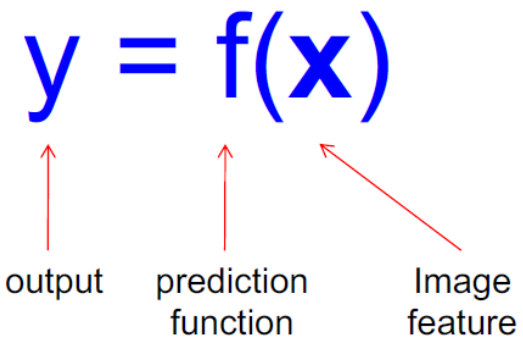





What we will learn today?

- Introduction to object recognition
- K-nearest neighbor algorithm
- Naïve Bayes Classification
- A simple Object Recognition pipeline

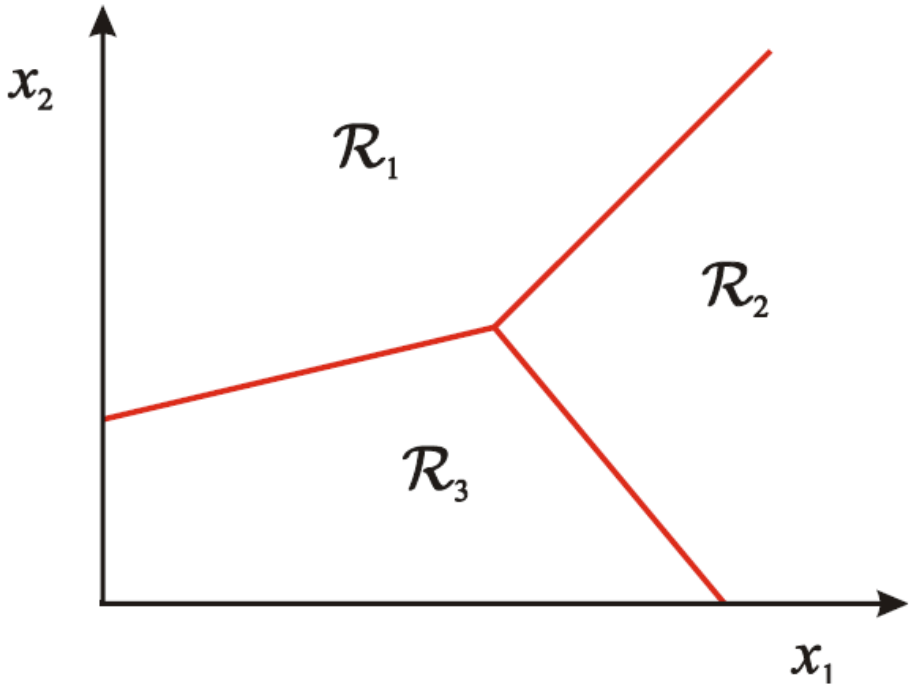
The machine learning framework



- Training: given a training set of labeled examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- Testing: apply f to a never before seen test example x and output the predicted value $y = f(x)$

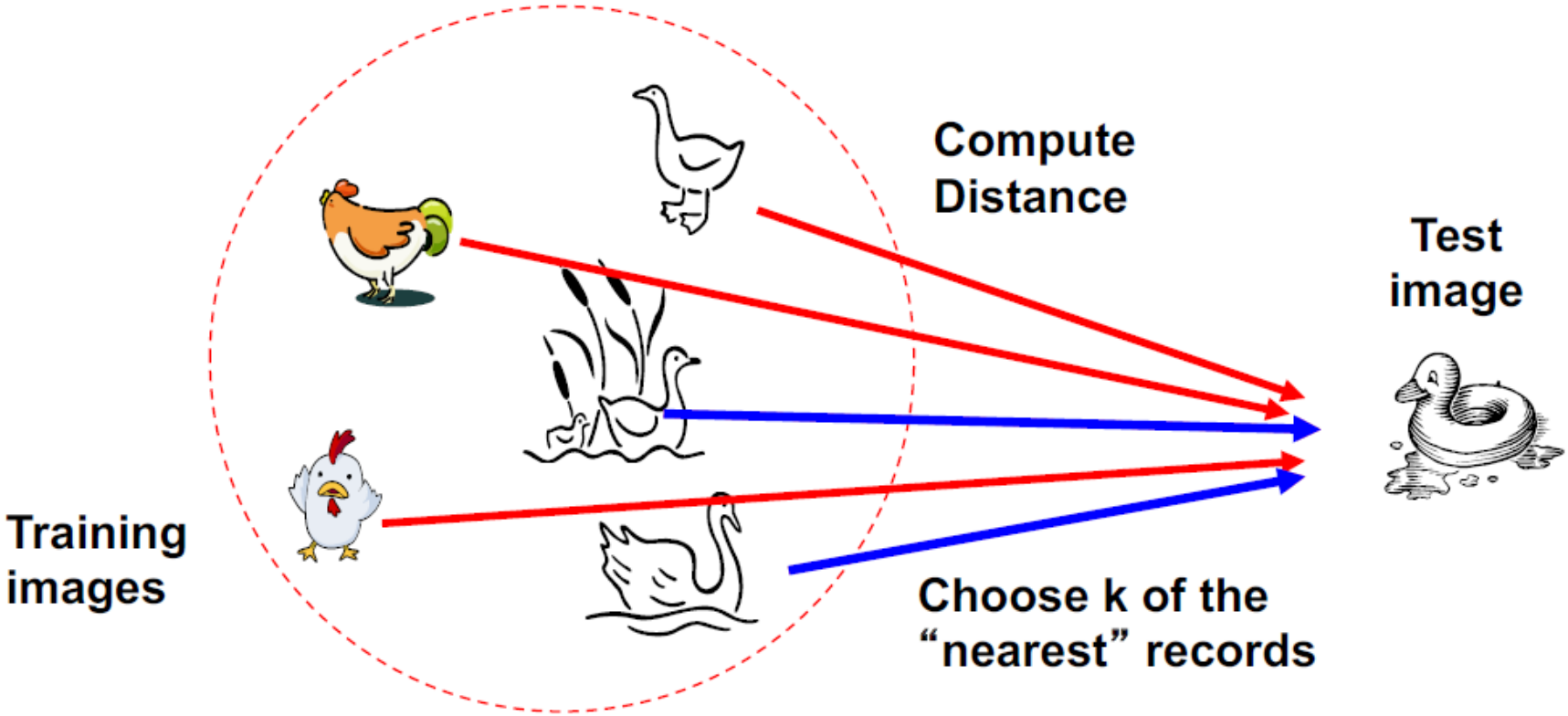
Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into decision regions separated by decision boundaries



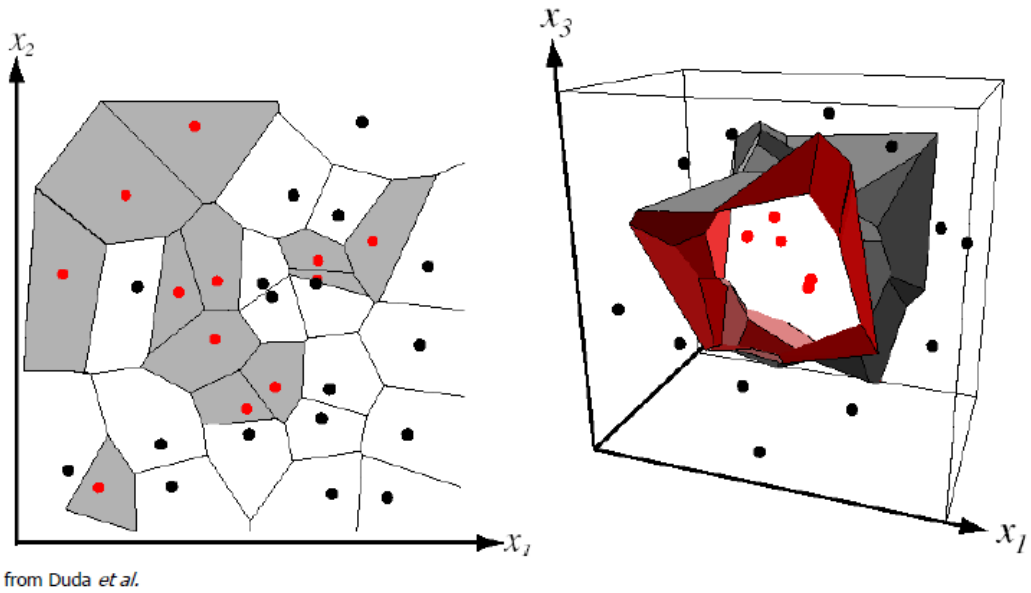
Nearest Neighbor Classifier

- Assign label of nearest training data point to each test data point



Nearest Neighbor Classifier

- Assign label of nearest training data point to each test data point
- Allows for complex decision boundaries that are shaped around each data point in the training set.



from Duda *et al.*

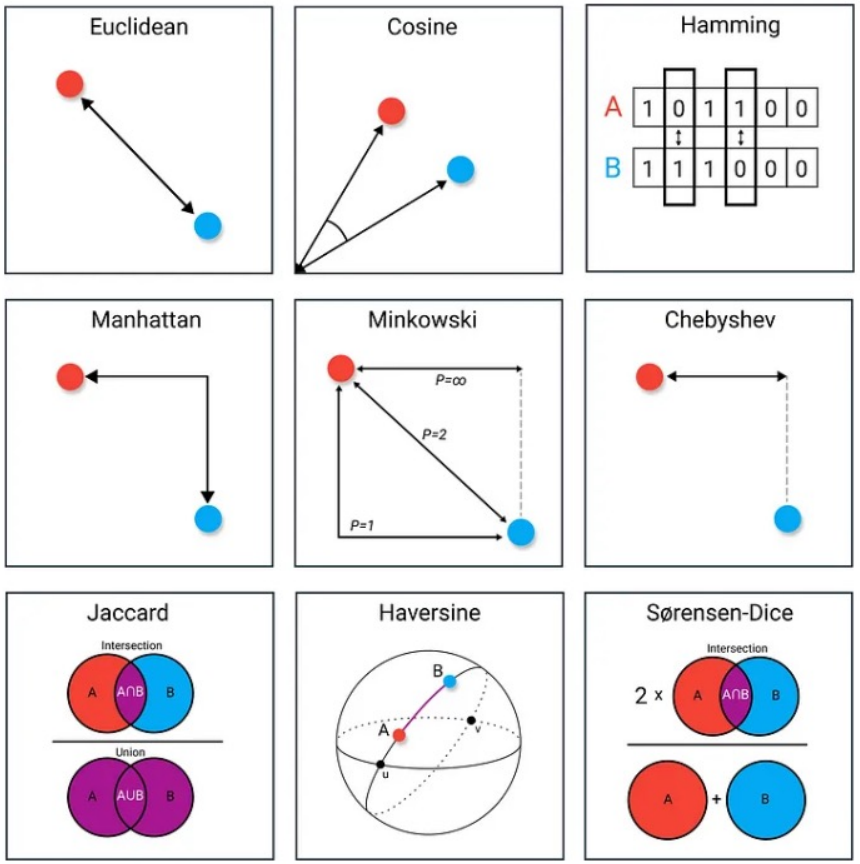
partitioning of feature space
for two-category 2D and 3D data

K-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X_i^n - X_i^m)^2}$$

Where X^n and X^m are the n-th and m-th data points

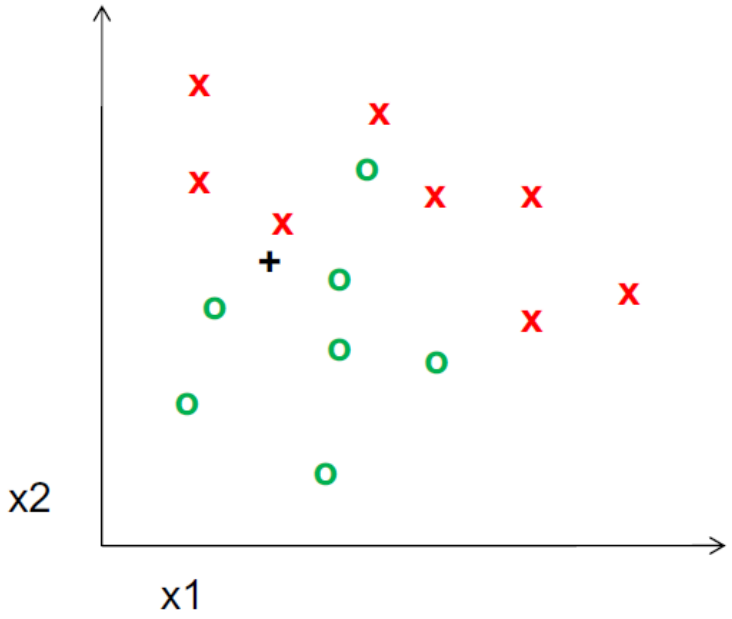


K-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X_i^n - X_i^m)^2}$$

Where X^n and X^m are the n-th and m-th data points

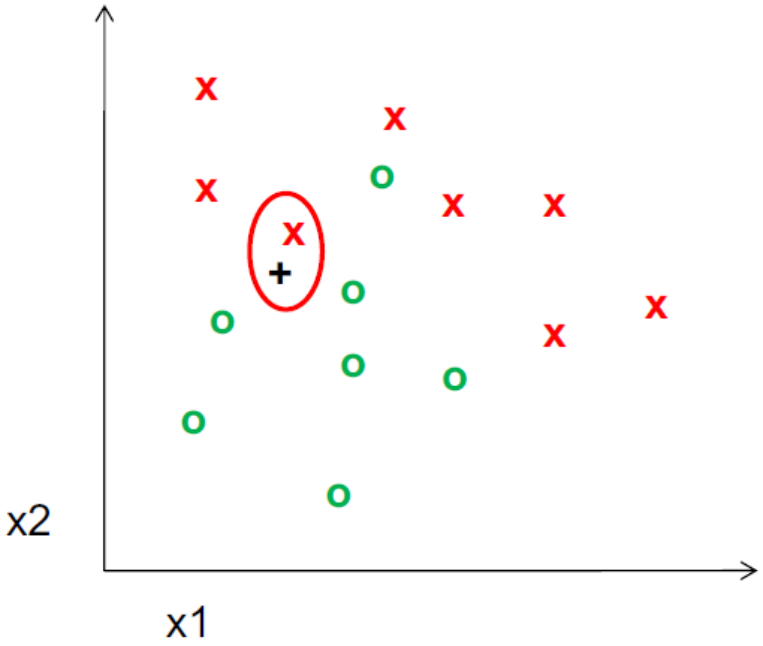


1-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X_i^n - X_i^m)^2}$$

Where X^n and X^m are the n-th and m-th data points

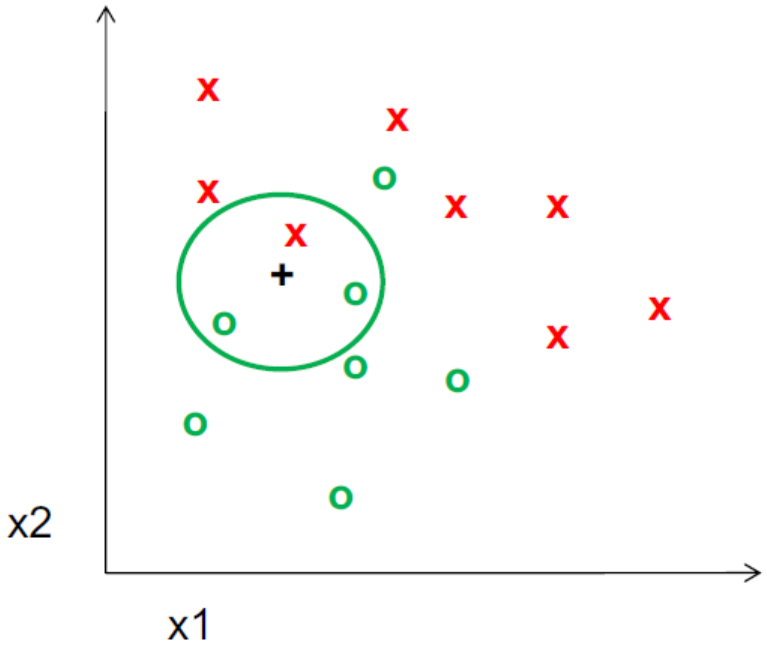


3-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X_i^n - X_i^m)^2}$$

Where X^n and X^m are the n-th and m-th data points

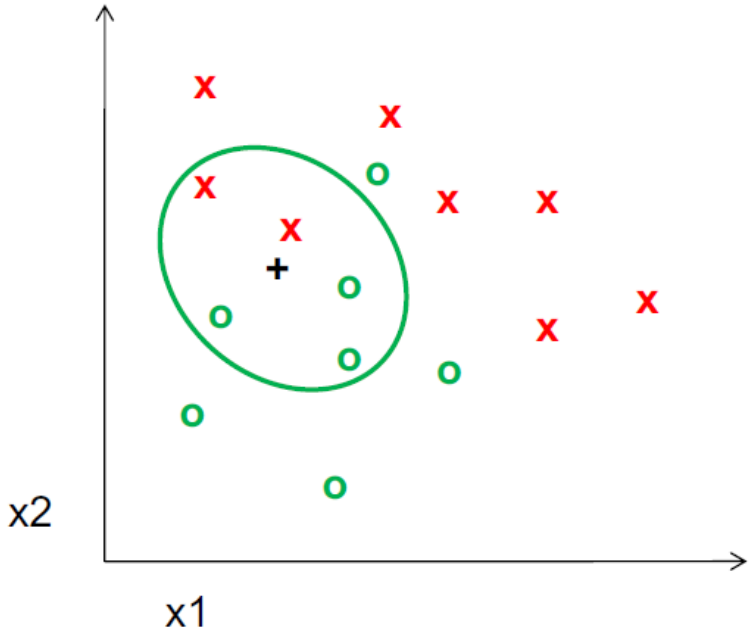


5-nearest neighbor

Distance measure - Euclidean

$$Dist(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X_i^n - X_i^m)^2}$$

Where X^n and X^m are the n-th and m-th data points



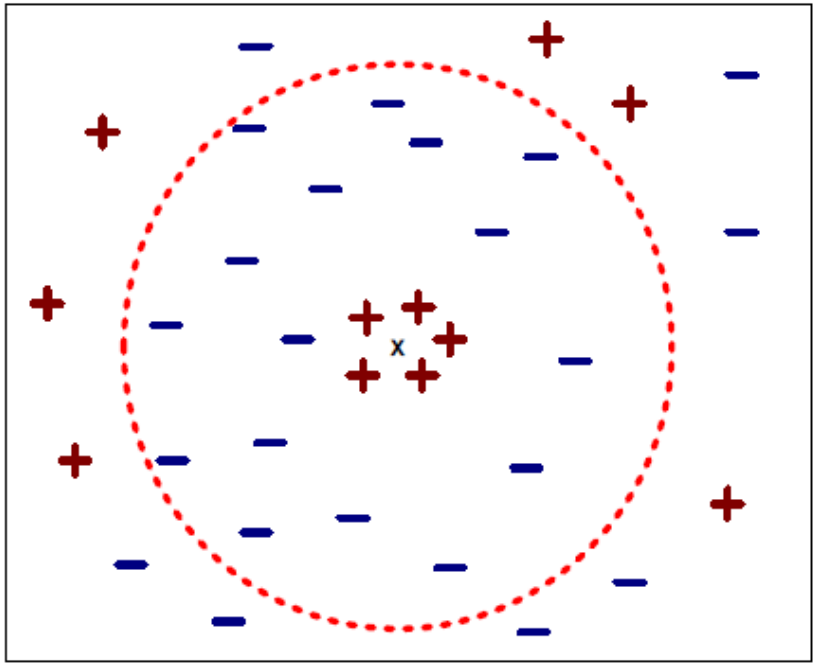
K-NN: a very useful algorithm

- Simple, a good one to try first
- Very flexible decision boundaries
- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error (out of scope for this class).

https://cedar.buffalo.edu/~srihari/CSE555/Chap4.Part3.pdf

K-NN: issues to keep in mind

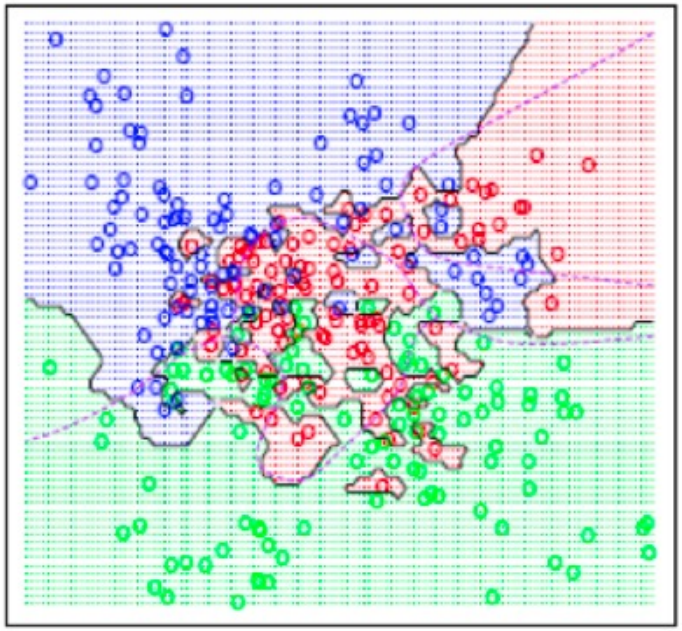
- Choosing the value of k:
  - – If too small, sensitive to noise points
  - – If too large, neighborhood may include points from other classes



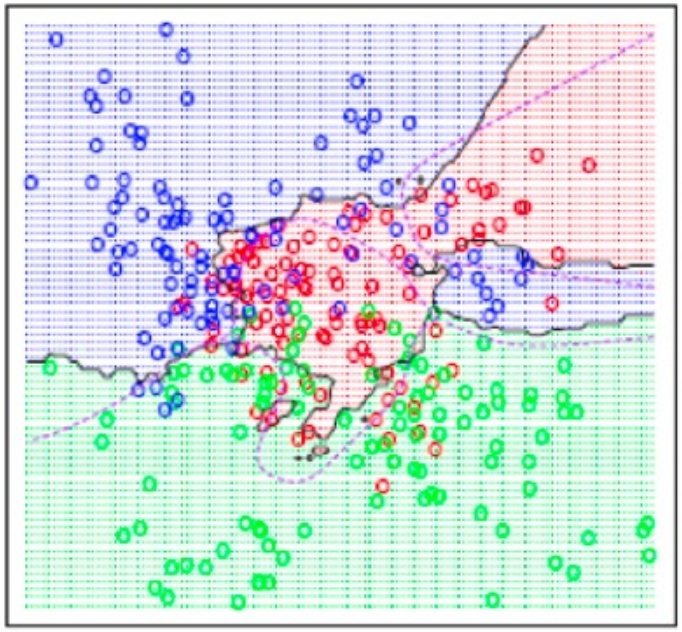
K-NN: issues to keep in mind

- Choosing the value of k:
  - – If too small, sensitive to noise points
  - – If too large, neighborhood may include points from other classes

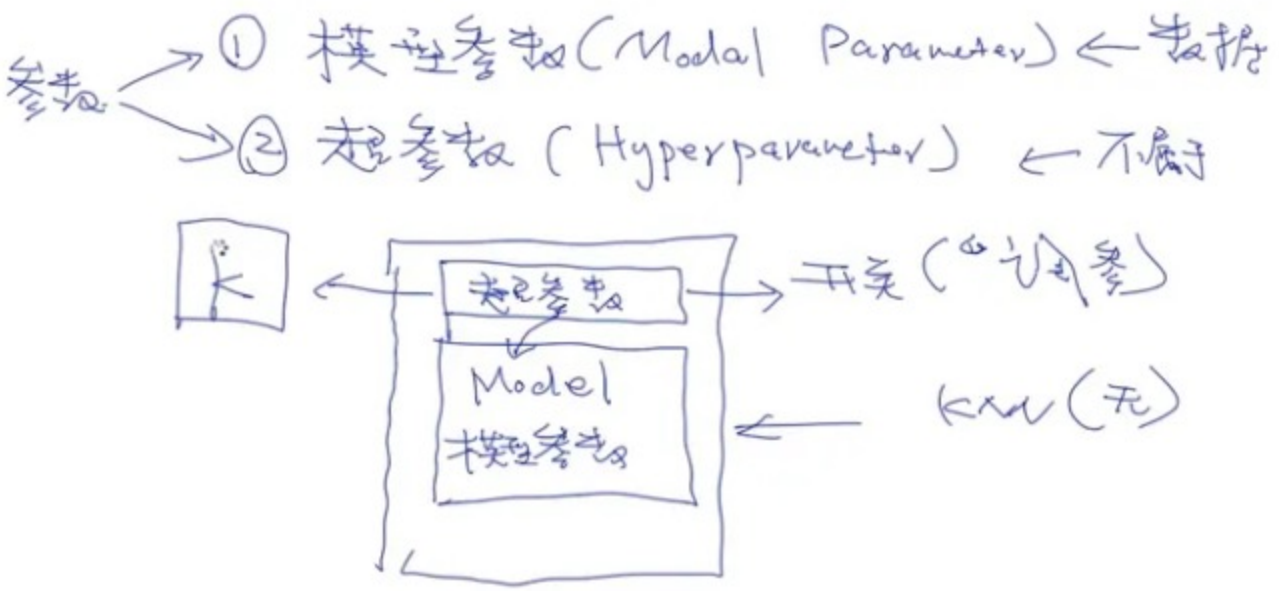
K=1



K=15



K-NN: issues to keep in mind



K-NN：Hyperparameters

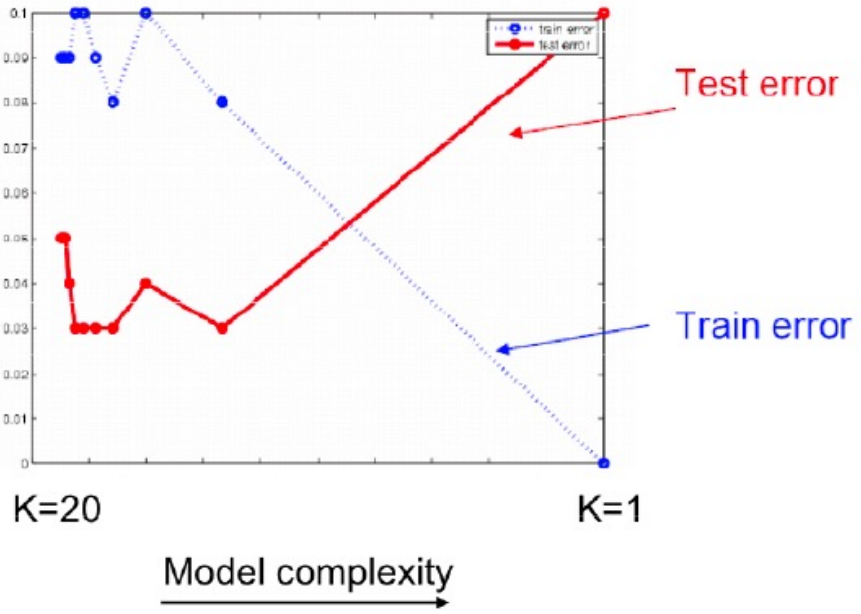
- What is the best distance to use?
- What is the best value of k to use?

- i.e., how do we set the hyperparameters?

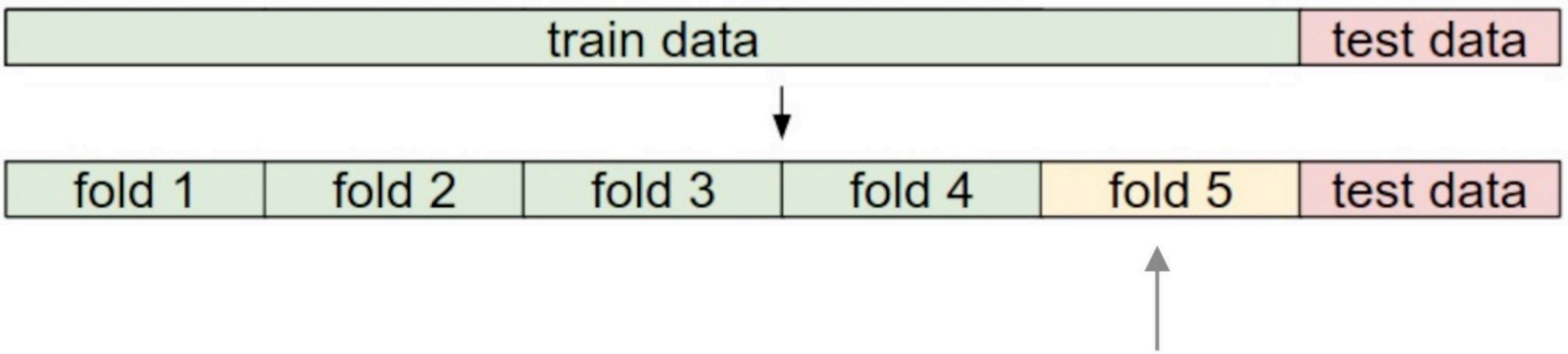
- Very problem-dependent
- Must try them all and see what works best

K-NN: issues to keep in mind

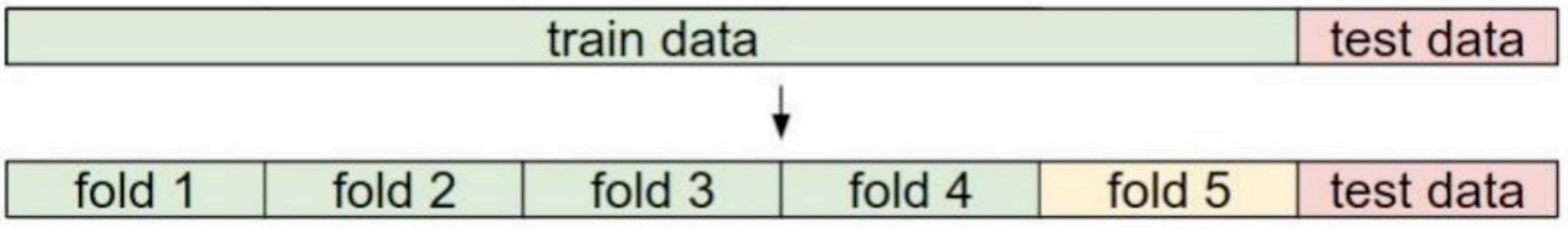
- Choosing the value of k:
  - – If too small, sensitive to noise points
  - – If too large, neighborhood may include points from other classes
  - – Solution: cross validate!



Cross validation



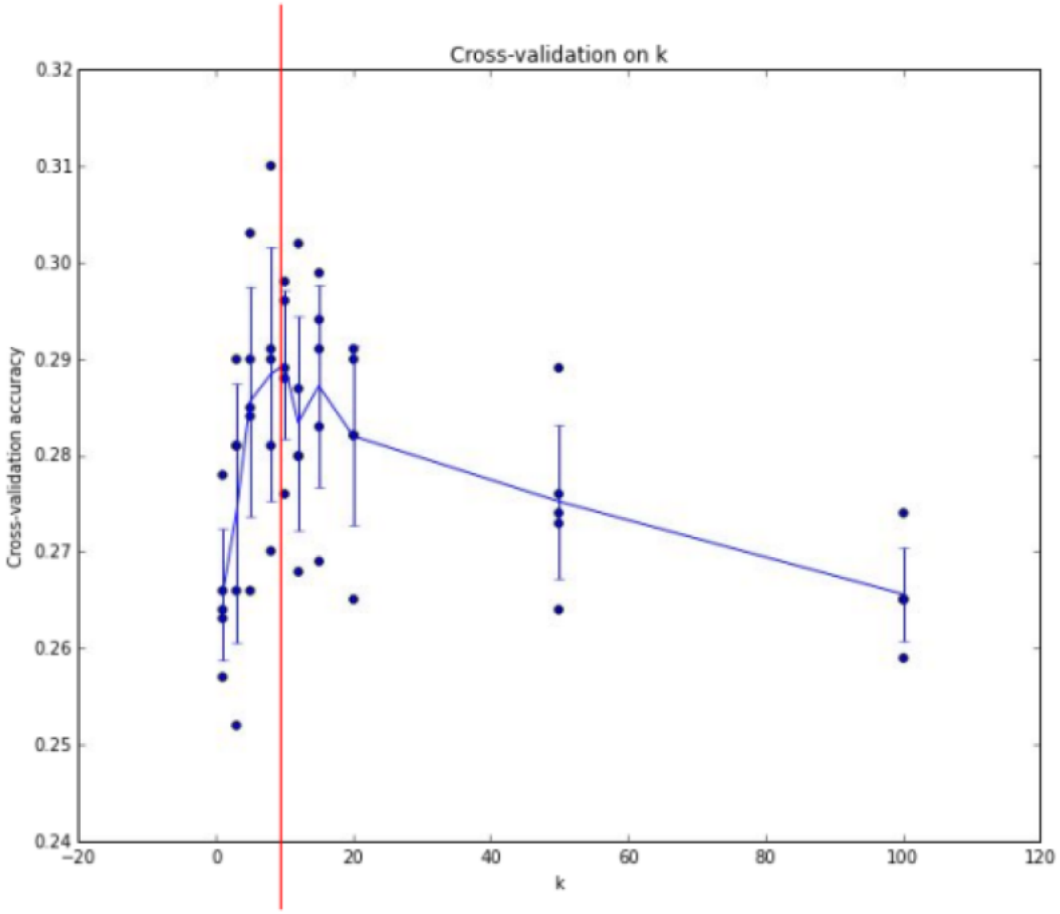
Cross validation



Cross-validation

cycle through the choice of which fold is the validation fold, average results.

Cross validation



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 ~= 7 works best
for this data)

K-NN: Pros and Cons

Pros

- simple yet effective

Cons

- search is expensive (can be sped-up)
- storage requirements
- difficulties with high-dimensional data

K-NN: Complexity and Storage

- N training images, M test images

Do not consider the feature dimension for each image

- Training: ?
- Testing: ?

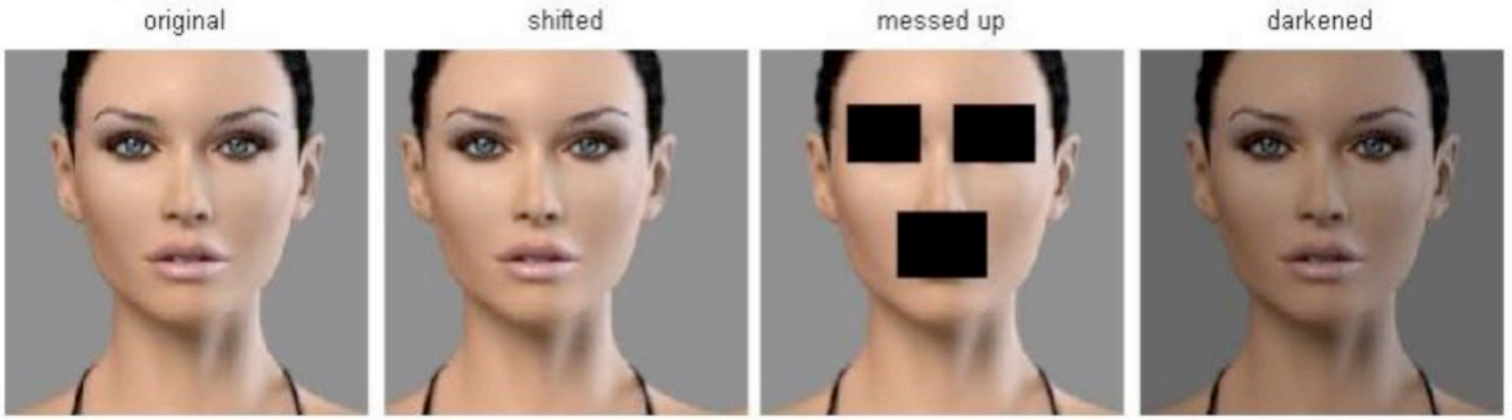
K-NN: Complexity and Storage

- N training images, M test images
- Training: O(1)
- Testing: O(MN)
- Hmm…
  - Normally need the opposite
  - Slow training (ok), fast testing (necessary)

K-NN in image recognition

k-Nearest Neighbor on images **never used.**

- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive

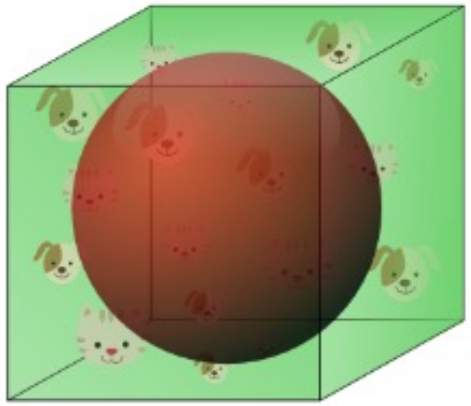


(all 3 images have same L2 distance to the one on the left)

K-NN: issues to keep in mind

- Choosing the value of k:
  - – If too small, sensitive to noise points
  - – If too large, neighborhood may include points from other classes
  - – Solution: cross validate!
- Complexity and Storage:
  - – Solution: Representative samples! KD tree (only for low dimension)!
- Can produce counter-intuitive results (using Euclidean measure)
  - – Solution: normalize the vectors to unit length
- Curse of Dimensionality
  - – Solution: no good one

K-NN: issues to keep in mind



- Curse of Dimensionality
  - – Solution: no good one

Many classifiers to choose from

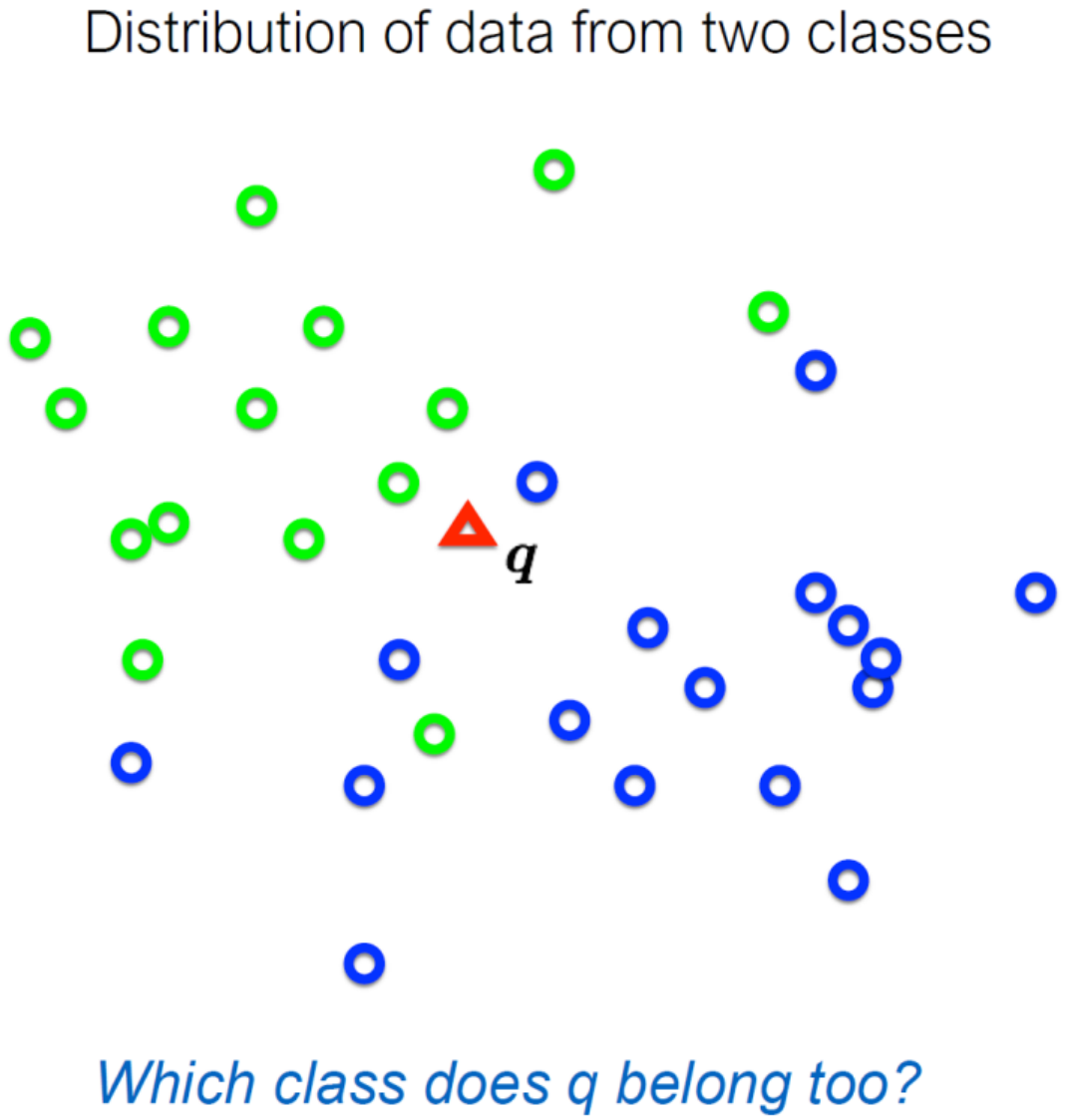
- K-nearest neighbor
- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- RBMs
- Etc.

Which is the best one?

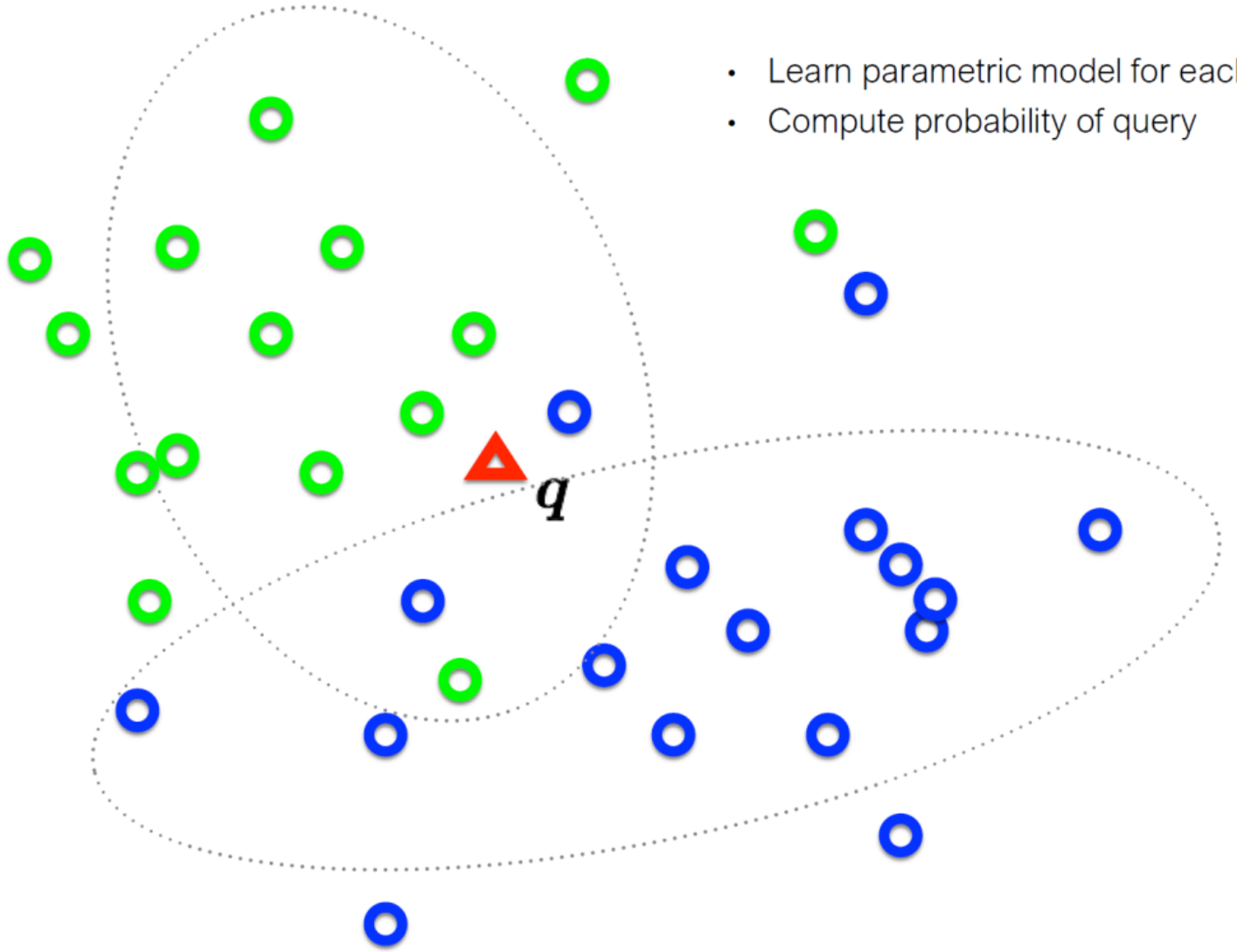
What we will learn today?

- Introduction to object recognition
- K-nearest neighbor algorithm
- Naïve Bayes Classification （朴素贝叶斯分类）
- A simple Object Recognition pipeline

Naïve Bayes


Distribution of data from two classes
q
Which class does q belong too?

Naïve Bayes


• Learn parametric model for each class
• Compute probability of query
q

Naïve Bayes

This is called the posterior.

the probability of a class z given the observed features X

$$p(z|X)$$

For classification, z is a discrete random variable
(e.g., car, person, building)

X is a set of observed features
(e.g., features from a single image)

(it's a function that returns a single probability value)

Naïve Bayes

This is called the posterior:

the probability of a class z given the observed features X

$$p(z|x_1, \ldots, x_N)$$

For classification, z is a discrete random variable
(e.g., car, person, building)

Each x is an observed feature
(e.g., visual words)

(it's a function that returns a single probability value)

Naïve Bayes

Recall:

The posterior can be decomposed according to **Bayes' Rule**

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

posterior; likelihood; prior

In our context…

$$p(\boldsymbol{z}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \frac{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)}$$

Naïve Bayes

The naive Bayes' classifier is solving this optimization

$$\hat{z} = \arg\max_{z \in \boldsymbol{z}} p(z|\boldsymbol{X})$$

MAP (maximum a posteriori) estimate

$$\hat{z} = \arg\max_{z \in \boldsymbol{z}} \frac{p(\boldsymbol{X}|z)p(z)}{p(\boldsymbol{X})}$$

Bayes' Rule

$$\hat{z} = \arg\max_{z \in \boldsymbol{z}} p(\boldsymbol{X}|z)p(z)$$

Remove constants

To optimize this…we need to compute this

Compute the likelihood…

Naïve Bayes

A naive Bayes' classifier assumes all features are ***conditionally independent***

$$
\begin{aligned}
p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N | \boldsymbol{z}) &= p(\boldsymbol{x}_1 | \boldsymbol{z}) p(\boldsymbol{x}_2, \ldots, \boldsymbol{x}_N | \boldsymbol{z}) \\
&= p(\boldsymbol{x}_1 | \boldsymbol{z}) p(\boldsymbol{x}_2 | \boldsymbol{z}) p(\boldsymbol{x}_3, \ldots, \boldsymbol{x}_N | \boldsymbol{z}) \\
&= p(\boldsymbol{x}_1 | \boldsymbol{z}) p(\boldsymbol{x}_2 | \boldsymbol{z}) \cdots p(\boldsymbol{x}_N | \boldsymbol{z})
\end{aligned}
$$

Recall:

$$p(x, y) = p(x|y)p(y) \qquad p(x, y) = p(x)p(y)$$

Naïve Bayes

To compute the MAP estimate

Given (1) a set of known parameters

$$p(\boldsymbol{z}) \quad p(\boldsymbol{x}|\boldsymbol{z})$$

(2) observations

$$\{x_1, x_2, \ldots, x_N\}$$

Compute which z has the largest probability

$$\hat{z} = \arg\max_{z \in \boldsymbol{z}} p(z) \prod_n p(x_n|z)$$

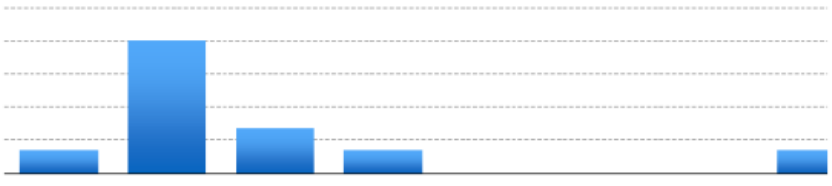
Naïve Bayes

The Newspa

Sunday, December 22, 2013

DARPA Selects Carnegie Me

The Tartan Rescue Team from Carnegie Mellon University's National Robotics Engineering Center ranked third among teams competing in the Defense Advanced Research Projects Agency (DARPA) Robotics Challenge Trials this weekend in Homestead, Fla., and was selected by the agency as one of eight teams eligible for DARPA funding to prepare for next December's finals. The team's four-kmbed CMU Highly Intelligent Mobile Platform, or CHIMP, robot scored 18 out of a possible 32 points during the two-day trials. It demonstrated its ability to perform such tasks as removing debris, cutting a hole through a wall and closing a series of valves.



count	1	6	2	1	0	0	0	1
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
p(x|z)	0.09	0.55	0.18	0.09	0.0	0.0	0.0	0.09

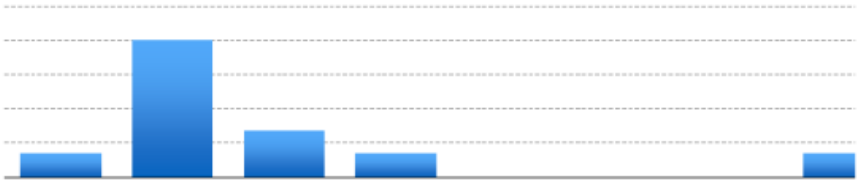
$$p(X|z) = \prod_v p(x_v|z)^{c(w_v)}$$
$$= (0.09)^1(0.55)^6 \cdots (0.09)^1$$

Numbers get really small so use log probabilities

$$\log p(X|z = \text{'grandchallenge'}) = -2.42 - 3.68 - 3.43 - 2.42 - 0.07 - 0.07 - 0.07 - 2.42 = -14.58$$

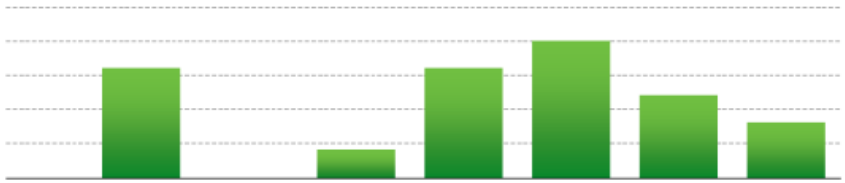
$$\log p(X|z = \text{'softrobot'}) = -7.63 - 9.37 - 15.18 - 2.97 - 0.02 - 0.01 - 0.02 - 2.27 = -37.48$$

Naïve Bayes



count	1	6	2	1	0	0	0	1
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
p(x|z)	0.09	0.55	0.18	0.09	0.0	0.0	0.0	0.09

log $p(X|z=$'grandchallenge'$) =$ **-14.58**
log $p(X|z=$'soft robot'$) =$ -37.48



count	0	4	0	1	4	5	3	2
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
p(x|z)	0.0	0.21	0.0	0.05	0.21	0.26	0.16	0.11

log $p(X|z=$'grandchallenge'$) =$ -94.06
log $p(X|z=$'soft robot'$) =$ **-32.41**

Naïve Bayes

We have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Problem: Players will play if the weather is sunny. Is this statement correct?

We can solve it using the above-discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here P(Sunny | Yes) * P(Yes) is in the numerator, and P (Sunny) is in the denominator.

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naïve Bayes

Consider the problem of playing golf. The dataset is represented as below.

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The variable y is the class variable(play golf), which represents if it is suitable to play golf or not given the conditions. Variable X represent the parameters /features.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Naïve Bayes

Let us test it on a new set of features (let us call it today):

```
today = (Sunny, Hot, Normal, False)
```

So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(Yes|today) \propto \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx 0.0141$$

and

$$P(No|today) \propto \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.0068$$

Now, since

$$P(Yes|today) + P(No|today) = 1$$

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that golf would be played is 'Yes'.

Many classifiers to choose from

- K-nearest neighbor
- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- RBMs
- Etc.

Which is the best one?

Generalization

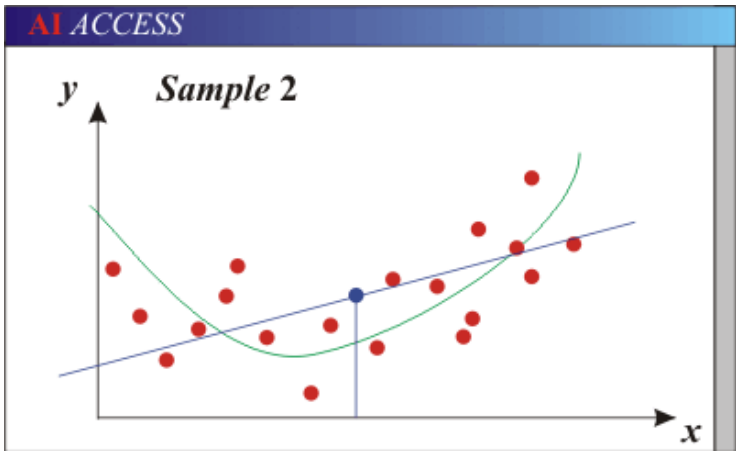
- How well does a learned model generalize from the data it was trained on to a new test set?

Training set (labels known)

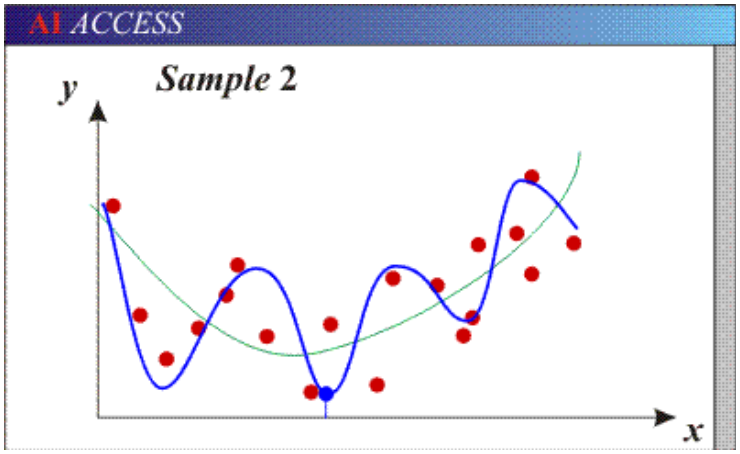
Test set (labels unknown)

Bias-Variance Trade-off

- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).



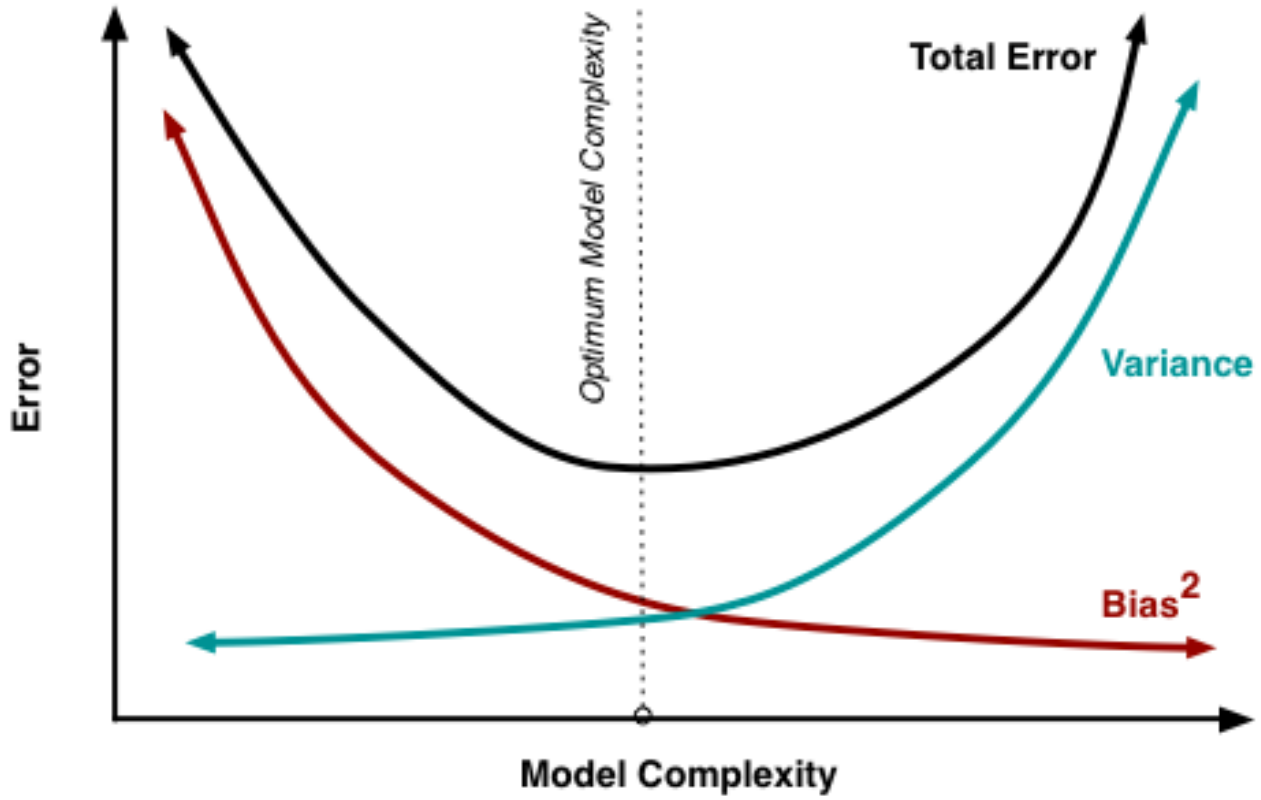
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).



Bias versus variance

- Components of generalization error
  - – **Bias**: how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - – **Variance**: how much models estimated from different training sets differ from each other
- **Underfitting**: model is too “simple” to represent all the relevant class characteristics
  - – High bias and low variance
  - – High training error and high test error
- **Overfitting**: model is too “complex” and fits irrelevant characteristics (noise) in the data
  - – Low bias and high variance
  - – Low training error and high test error

Bias versus variance trade off



No Free Lunch Theorem

- In a supervised learning setting, we can't tell which classifier will have best generalization

Remember…

- No classifier is inherently better than any other: you need to make assumptions to generalize
- Three kinds of error
  - – Inherent: unavoidable
  - – Bias: due to over-simplifications
  - – Variance: due to inability to perfectly estimate parameters from limited data

How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data

How do you reduce bias?

Remarks about applying ML to object recognition

- There are machine learning algorithms to choose from
- Know your data:
  - – How much supervision do you have?
  - – How many training examples can you afford?
  - – How noisy?
- Know your goal (i.e. task):
  - – Affects your choices of representation
  - – Affects your choices of learning algorithms
  - – Affects your choices of evaluation metrics
- Understand the math behind each machine learning algorithm under consideration!

What we will learn today?

- Introduction to object recognition
- K-nearest neighbor algorithm
- Naïve Bayes Classification
- A simple Object Recognition pipeline

Object recognition: a classification framework

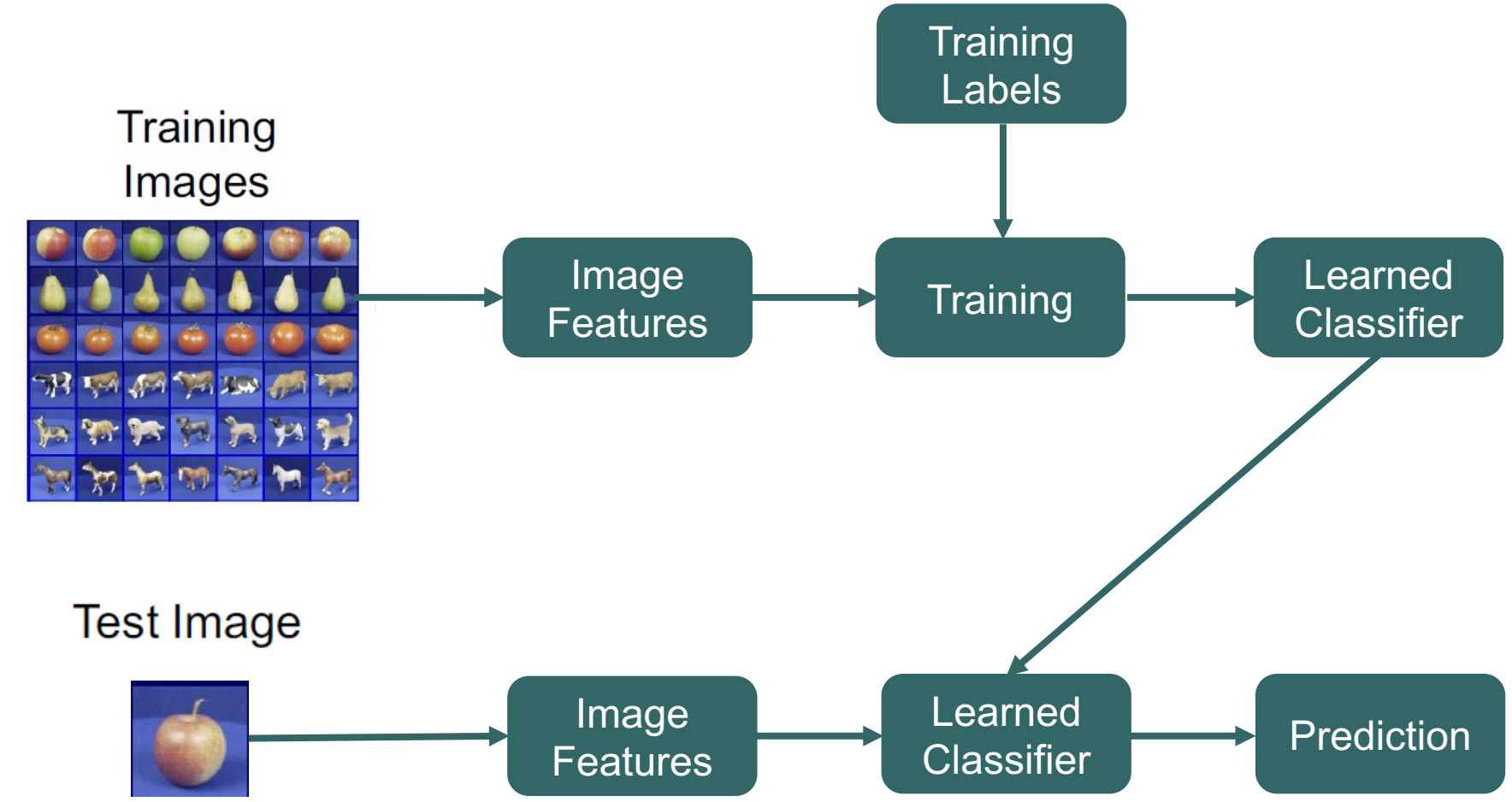
- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

A simple pipeline - Training



A simple pipeline - Training

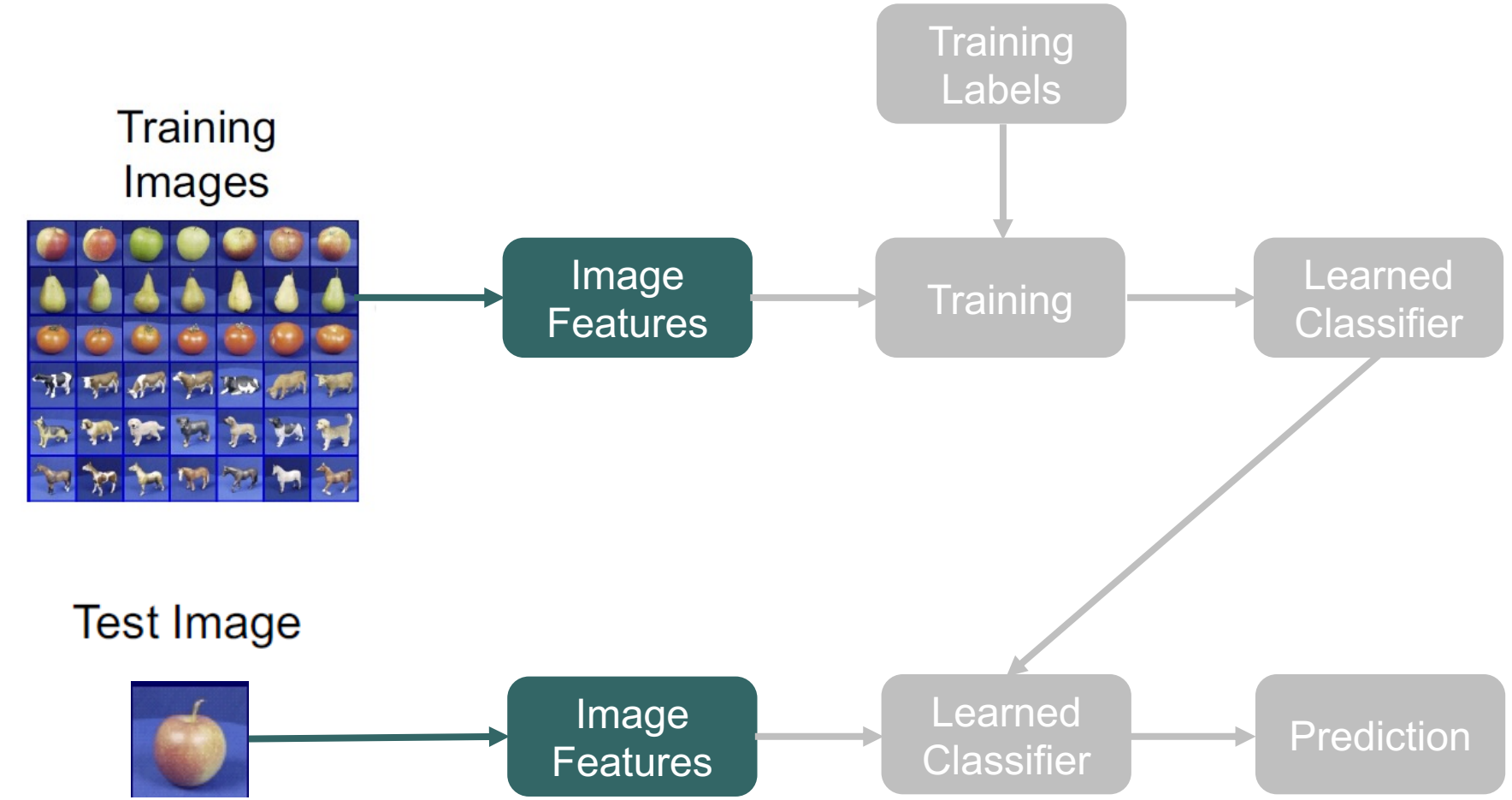
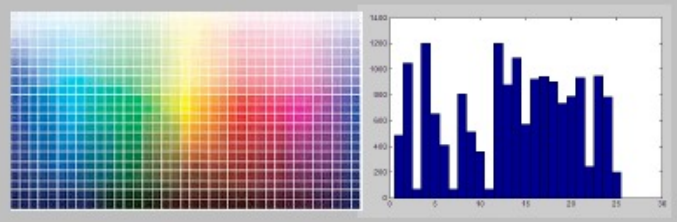


Image features

Input image

Color: Quantize RGB values

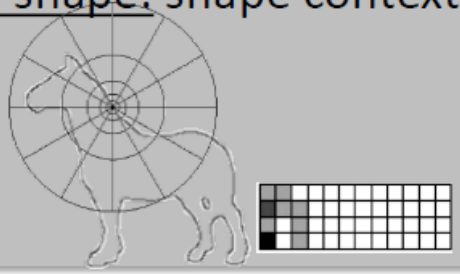


Invariance?
- ✓ Translation
- ✓ Scale
- ✓ Rotation
- ✗ Occlusion

Global shape: PCA space

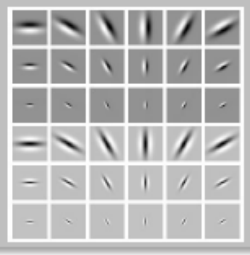
Invariance?
- ✓ Translation
- ? Scale
- ✓ Rotation (in-planar)
- ✗ Occlusion

Local shape: shape context



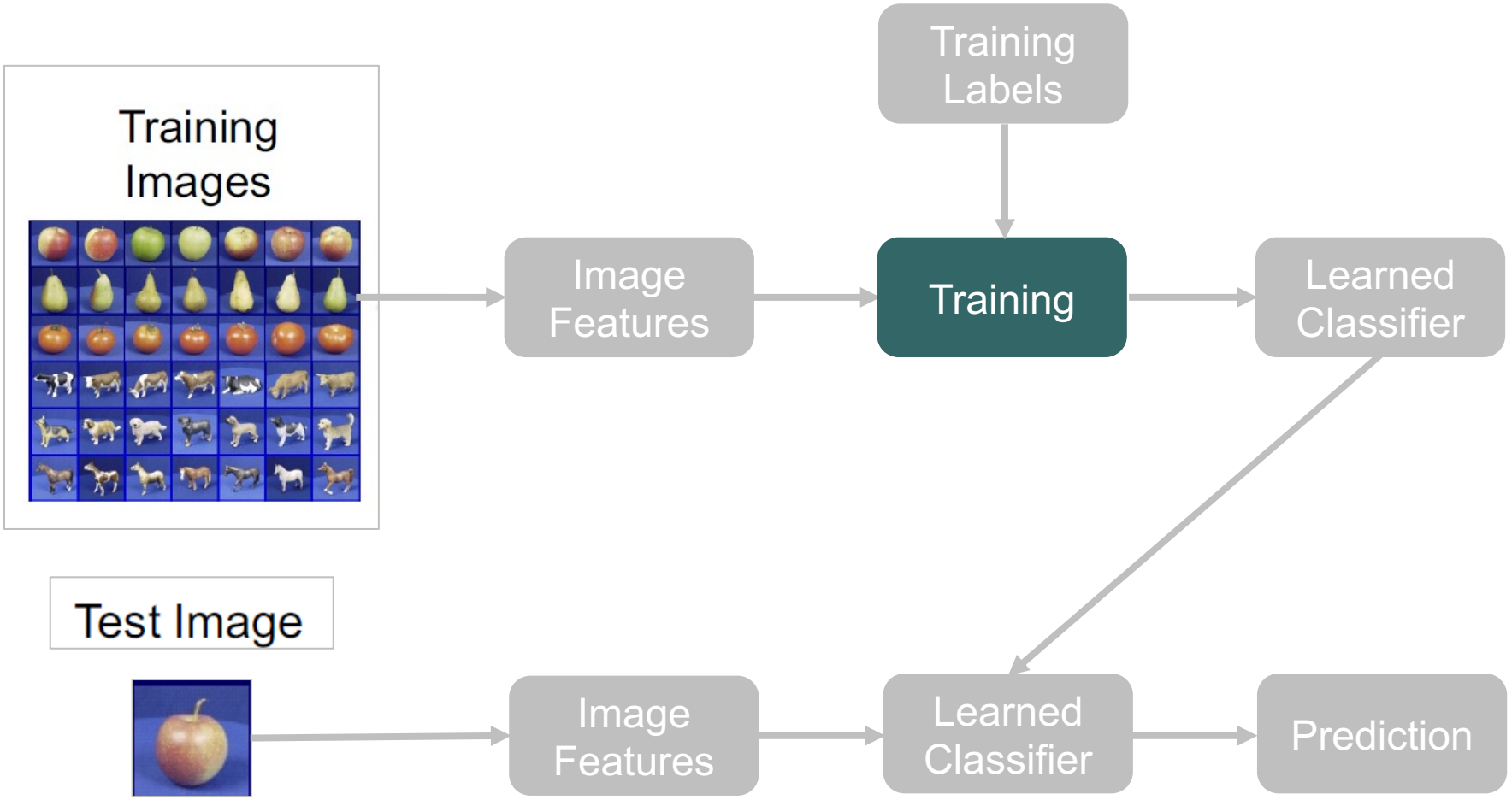
Invariance?
- ✓ Translation
- ✓ Scale
- ? Rotation (in-planar)
- ✗ Occlusion

Texture: Filter banks

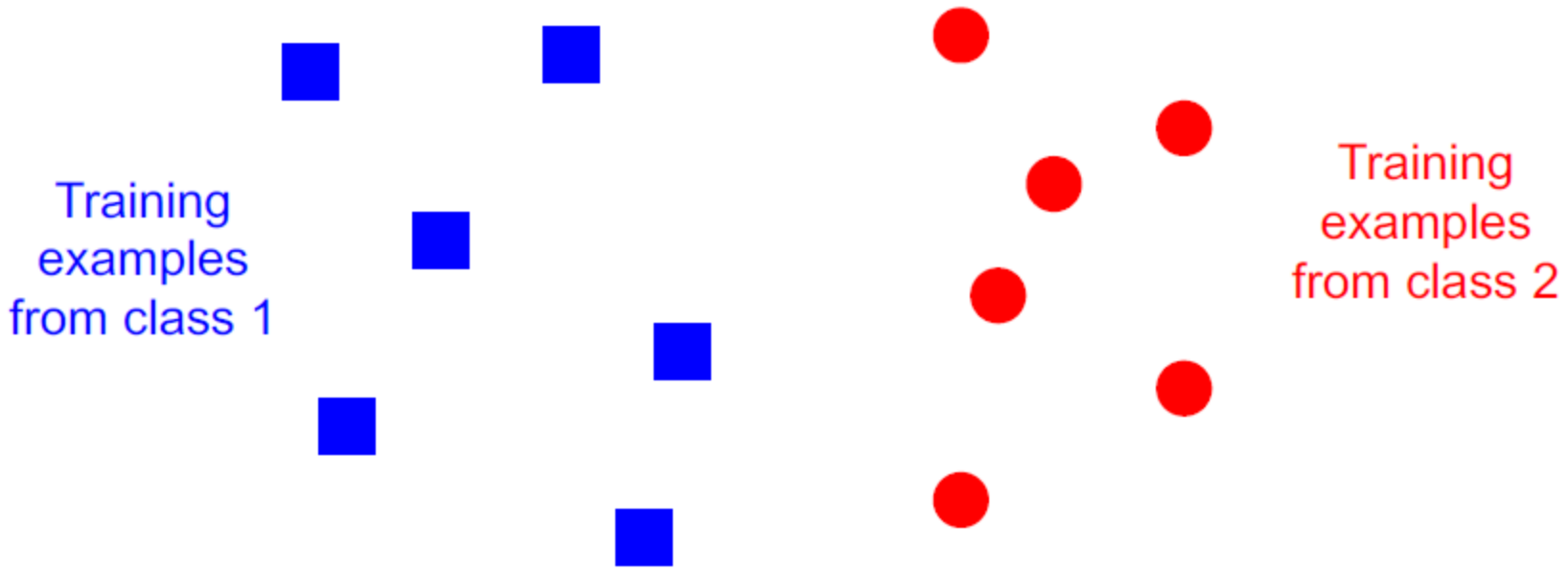


Invariance?
- ✓ Translation
- ? Scale
- ? Rotation (in-planar)
- ✗ Occlusion

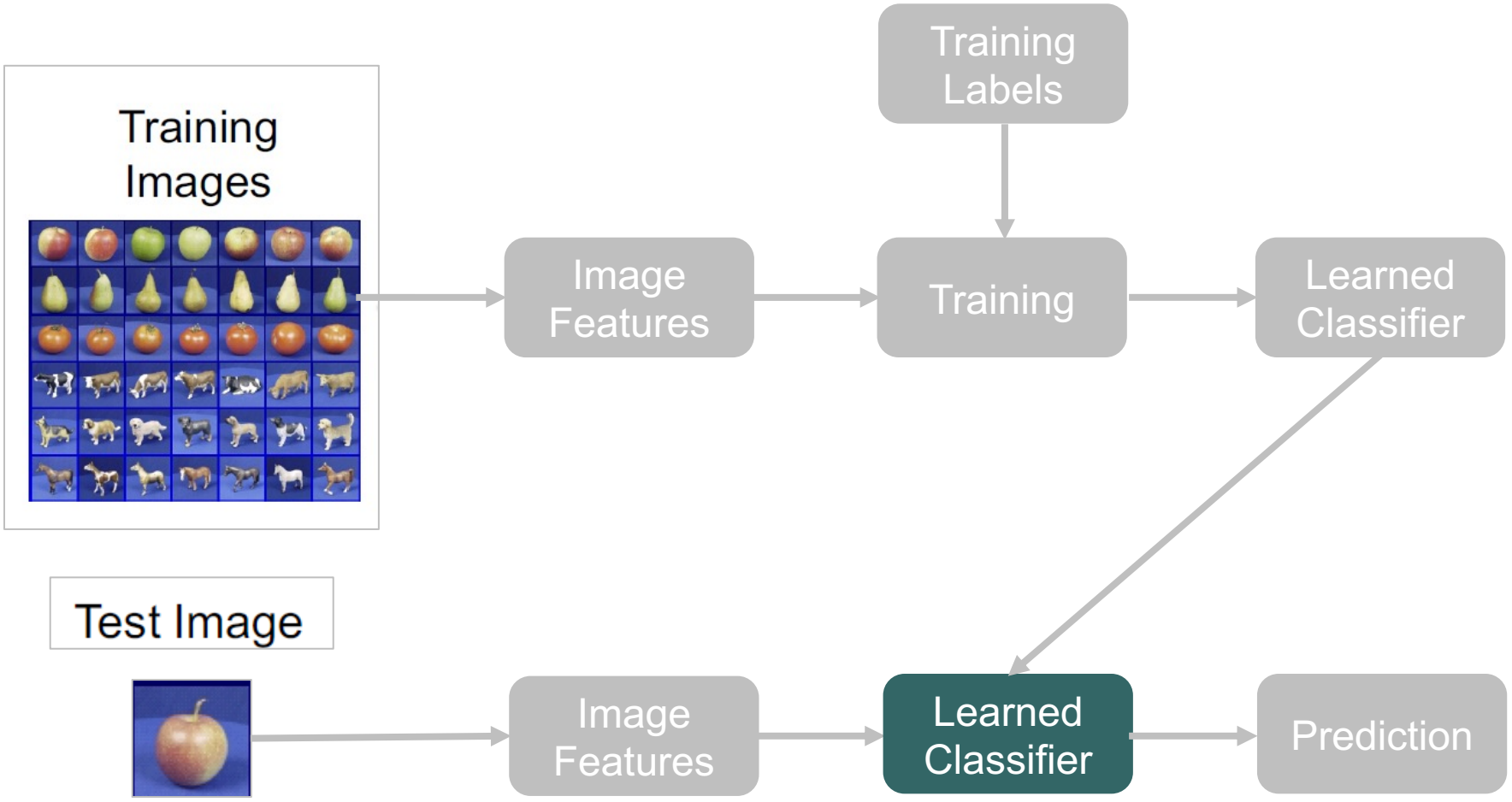
A simple pipeline - Training



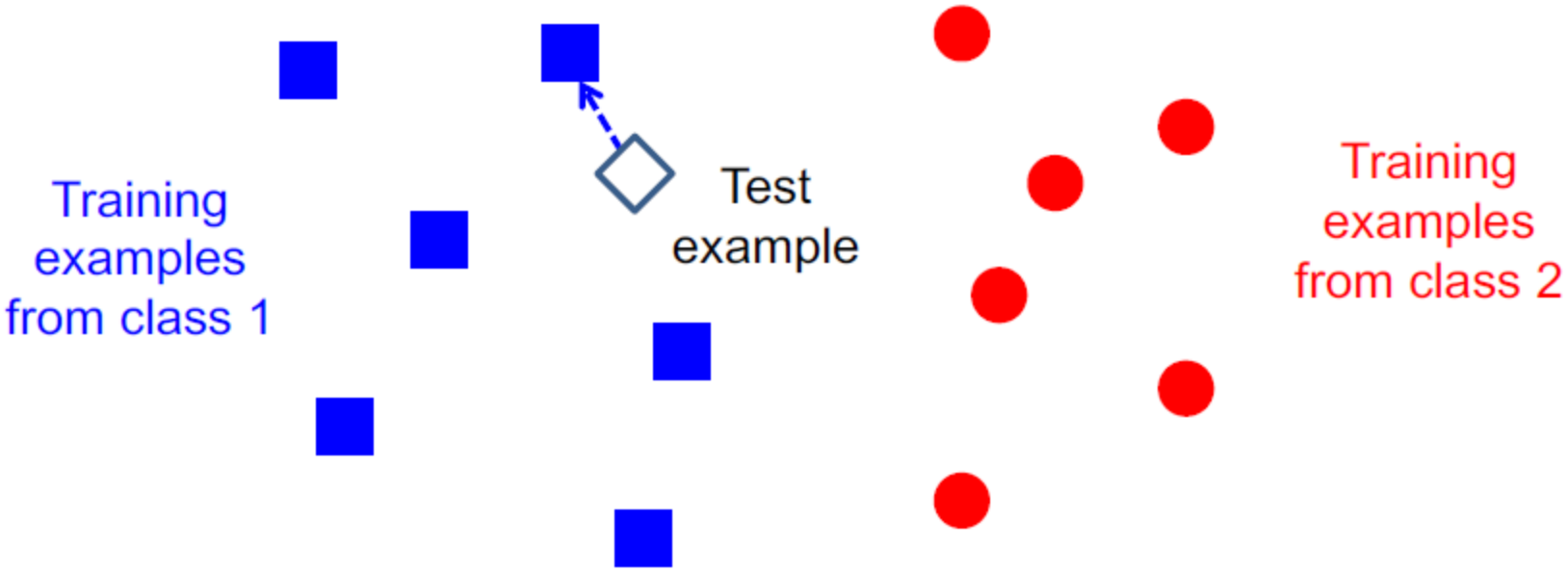
Classifiers: Nearest neighbor



A simple pipeline - Training



Classifiers: Nearest neighbor



Experiment results

	Color	$D_x D_y$	Mag-Lap	PCA Masks	PCA Gray	Cont. Greedy	Cont. DynProg	Avg.
apple	57.56%	**85.37%**	80.24%	78.78%	**88.29%**	77.07%	76.34%	77.66%
pear	66.10%	90.00%	85.37%	**99.51%**	**99.76%**	90.73%	91.71%	89.03%
tomato	**98.54%**	94.63%	**97.07%**	67.80%	76.59%	70.73%	70.24%	82.23%
cow	86.59%	82.68%	**94.39%**	75.12%	62.44%	86.83%	86.34%	82.06%
dog	34.63%	62.44%	74.39%	72.20%	66.34%	**81.95%**	**82.93%**	67.84%
horse	32.68%	58.78%	70.98%	77.80%	77.32%	**84.63%**	**84.63%**	69.55%
cup	79.76%	66.10%	77.80%	**96.10%**	**96.10%**	**99.76%**	**99.02%**	87.81%
car	62.93%	**98.29%**	77.56%	**100.0%**	**97.07%**	**99.51%**	**100.0%**	90.77%
total	64.85%	79.79%	82.23%	83.41%	82.99%	86.40%	86.40%	80.87%

What we have learned today?

- Introduction to recognition tasks
- Statistical learning approach and K-nearest neighbor algorithm
- Naïve Bayes Classification
- Classic / Shallow Pipeline of Object Recognition