Lecture: Object Recognition

Juan Carlos Niebles and Ranjay Krishna
Stanford Vision and Learning Lab
What we will learn today?

• Introduction to object recognition
• K-nearest neighbor algorithm
• A simple Object Recognition pipeline
What are the different visual recognition tasks?
Classification:
Does this image contain a building? [yes/no]

Yes!
Classification:
Is this an beach?
Image search
Image Search

Organizing photo collections
Detection:
Does this image contain a car? [where?]
Detection:
Which object does this image contain? [where?]
Detection:
Accurate localization (segmentation)
Detection: Estimating object semantic & geometric attributes

Object: Building, 45º pose, 8-10 meters away
It has bricks

Object: Person, back; 1-2 meters away

Object: Police car, side view, 4-5 m away
Categorization vs Single instance recognition

Does this image contain the Chicago Macy’s building?
Categorization vs Single instance recognition

Where is the crunchy nut?
Applications of computer vision

• Recognizing landmarks in mobile platforms
Activity or Event recognition

What are these people doing?
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?

~10,000 to 30,000
Challenges: viewpoint variation

Michelangelo 1475-1564
Challenges: illumination

image credit: J. Koenderink
Challenges: scale
Challenges: deformation
Challenges: occlusion

Magritte, 1957
Art Segway - Magritte
Challenges: background clutter

Kilmeny Niland. 1995
Challenges: intra-class variation
What we will learn today?

• Introduction
• K-nearest neighbor algorithm
• A simple Object Recognition pipeline
The machine learning framework

\[ y = f(x) \]

- **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 \( y = f(x) \)

Slide credit: L. Lazebnik
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

Source: N. Goyal
Nearest Neighbor Classifier

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

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

Source: D. Lowe
K-nearest neighbor

Distance measure - Euclidean

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

Where $X^n$ and $X^m$ are the n-th and m-th data points
1-nearest neighbor

Distance measure - Euclidean

\[ \text{Dist}(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X^n_i - X^m_i)^2} \]

Where \( X^n \) and \( X^m \) are the n-th and m-th data points
Distance measure - Euclidean

\[ \text{Dist}(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X^n_i - X^m_i)^2} \]

Where \(X^n\) and \(X^m\) are the n-th and m-th data points.
5-nearest neighbor

Distance measure - Euclidean

\[ Dist(X^n, X^m) = \sqrt{\sum_{i=1}^{D} (X^n_i - X^m_i)^2} \]

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).
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-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
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!

• Can produce counter-intuitive results (using Euclidean measure)
Euclidean measure

\[
\begin{align*}
\begin{array}{ccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\end{align*}
\]

vs

\[
\begin{align*}
\begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
\end{align*}
\]

d = 1.4142
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!

• Can produce counter-intuitive results (using Euclidean measure)
  – **Solution**: normalize the vectors to unit length
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!

• Can produce counter-intuitive results (using Euclidean measure)
  – Solution: normalize the vectors to unit length

• Curse of Dimensionality
Curse of dimensionality

- Assume 5000 points uniformly distributed in the unit hypercube and we want to apply 5-NN. Suppose our query point is at the origin.
  - In 1-dimension, we must go a distance of $5/5000=0.001$ on the average to capture 5 nearest neighbors.
  - In 2 dimensions, we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume.
  - In $d$ dimensions, we must go $(0.001)^{1/d}$
Curse of dimensionality

- Assume 5000 points uniformly distributed in the unit hypercube and we want to apply 5-NN. Suppose our query point is at the origin.
  - In 1-dimension, we must go a distance of 5/5000 = 0.001 on the average to capture 5 nearest neighbors.
  - In 2 dimensions, we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume.
  - In d dimensions, we must go $(0.001)^{1/d}$.
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!

• Can produce counter-intuitive results (using Euclidean measure)
  – **Solution**: normalize the vectors to unit length

• 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?

Slide credit: D. Hoiem
Generalization

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

Slide credit: D. Hoiem
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

Slide credit: L. Lazebnik
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?
Last remarks about applying machine learning methods 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
- K-nearest neighbor algorithm
- 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:

\[
f( ) = \text{“apple”}
\]
\[
f( ) = \text{“tomato”}
\]
\[
f( ) = \text{“cow”}
\]
A simple pipeline - Training

Training Images

Image Features
A simple pipeline - Training

Training Images

Image Features

Training

Training Labels
A simple pipeline - Training

Training Images

Training Labels

Image Features

Training

Learned Classifier
A simple pipeline - Training

Training Images

Training Labels

Image Features → Training → Learned Classifier

Test Image

Image Features
A simple pipeline - Training

Training Images

Training Labels

Image Features

Training

Learned Classifier

Test Image

Image Features

Learned Classifier

Prediction
A simple pipeline - Training

1. Training Images
2. Training Labels
3. Training
4. Learned Classifier
5. Learned Classifier
6. Prediction
7. Test Image
Image features

Input image

Color: Quantize RGB values

Invariance?
- Translation
- Scale
- Rotation
- Occlusion
Image features

Input image

Color: Quantize RGB values

Invariance?

Translation

Scale

Rotation (in-planar)

Occlusion
Image features

Input image

**Color:** Quantize RGB values

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

**Global shape:** PCA space

- Invariance?
  - Translation
  - Scale
  - Rotation (in-planar)
  - Occlusion
### Image features

**Input image**

<table>
<thead>
<tr>
<th><strong>Color</strong>: Quantize RGB values</th>
<th><strong>Invariance?</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Color histogram" /></td>
<td>☹️ Translation</td>
</tr>
<tr>
<td></td>
<td>☹️ Scale</td>
</tr>
<tr>
<td></td>
<td>☺️ Rotation (in-planar)</td>
</tr>
<tr>
<td></td>
<td>☹️ Occlusion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Global shape</strong>: PCA space</th>
<th><strong>Invariance?</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Global shape" /></td>
<td>☹️ Translation</td>
</tr>
<tr>
<td></td>
<td>☹️ Scale</td>
</tr>
<tr>
<td></td>
<td>☺️ Rotation (in-planar)</td>
</tr>
<tr>
<td></td>
<td>☹️ Occlusion</td>
</tr>
</tbody>
</table>
# Image features

## Color: Quantize RGB values

<table>
<thead>
<tr>
<th>Invariance?</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="neutral" alt="Translation" /></td>
</tr>
<tr>
<td><img src="neutral" alt="Scale" /></td>
</tr>
<tr>
<td><img src="happy" alt="Rotation" /></td>
</tr>
<tr>
<td><img src="sad" alt="Occlusion" /></td>
</tr>
</tbody>
</table>

### Global shape: PCA space

![No Invariance](sad)

<table>
<thead>
<tr>
<th>Invariance?</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="neutral" alt="Translation" /></td>
</tr>
<tr>
<td><img src="neutral" alt="Scale" /></td>
</tr>
<tr>
<td><img src="happy" alt="Rotation" /></td>
</tr>
<tr>
<td><img src="sad" alt="Occlusion" /></td>
</tr>
</tbody>
</table>

### Local shape: shape context

<table>
<thead>
<tr>
<th>Invariance?</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="neutral" alt="Translation" /></td>
</tr>
<tr>
<td><img src="neutral" alt="Scale" /></td>
</tr>
<tr>
<td><img src="happy" alt="Rotation" /> (in-planar)</td>
</tr>
<tr>
<td><img src="sad" alt="Occlusion" /></td>
</tr>
</tbody>
</table>
Image features

Input image

**Color:** Quantize RGB values

- Invariance?
  - Translation
  - Scale
  - Rotation
  - Occlusion

**Global shape:** PCA space

- Invariance?
  - Translation
  - Scale
  - Rotation
  - Occlusion

**Local shape:** shape context

- Invariance?
  - Translation
  - Scale
  - Rotation (in-planar)
  - Occlusion
Image features

**Color:** Quantize RGB values

Invariance?
- Translation
- Scale
- Rotation
- Occlusion

**Global shape:** PCA space

Invariance?
- Translation
- Scale
- Rotation
- Occlusion

**Local shape:** shape context

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

**Texture:** Filter banks

Invariance?
- Translation
- Scale
- Rotation (in-planar)
- Occlusion
Image features

Input image

<table>
<thead>
<tr>
<th>Color: Quantize RGB values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invariance?</td>
</tr>
<tr>
<td>Translation</td>
</tr>
<tr>
<td>Scale</td>
</tr>
<tr>
<td>Rotation</td>
</tr>
<tr>
<td>Occlusion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Global shape: PCA space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invariance?</td>
</tr>
<tr>
<td>Translation</td>
</tr>
<tr>
<td>Scale</td>
</tr>
<tr>
<td>Rotation</td>
</tr>
<tr>
<td>Occlusion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Local shape: shape context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invariance?</td>
</tr>
<tr>
<td>Translation</td>
</tr>
<tr>
<td>Scale</td>
</tr>
<tr>
<td>Rotation (in-planar)</td>
</tr>
<tr>
<td>Occlusion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Texture: Filter banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invariance?</td>
</tr>
<tr>
<td>Translation</td>
</tr>
<tr>
<td>Scale</td>
</tr>
<tr>
<td>Rotation (in-planar)</td>
</tr>
<tr>
<td>Occlusion</td>
</tr>
</tbody>
</table>
A simple pipeline - Training

Training Images

Image Features

Training

Labels

Learned Classifier

Test Image

Image Features

Learned Classifier

Prediction
Classifiers: Nearest neighbor

Training examples from class 1

Training examples from class 2

Slide credit: L. Lazebnik
A simple pipeline - Training

Training Images

Test Image

Training Labels

Image Features

Training

Learned Classifier

Learned Classifier

Prediction

Image Features
Classifiers: Nearest neighbor

- Training examples from class 1
- Test example
- Training examples from class 2

Slide credit: L. Lazebnik
## Results

Dataset: ETH-80, by B. Leibe, 2003

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>57.56%</td>
<td>85.37%</td>
<td>80.24%</td>
<td>78.78%</td>
<td>88.29%</td>
<td>77.07%</td>
<td>76.34%</td>
<td>77.66%</td>
</tr>
<tr>
<td>pear</td>
<td>66.10%</td>
<td>90.00%</td>
<td>85.37%</td>
<td>99.51%</td>
<td>99.76%</td>
<td>90.73%</td>
<td>91.71%</td>
<td>89.03%</td>
</tr>
<tr>
<td>tomato</td>
<td>98.54%</td>
<td>94.63%</td>
<td>97.07%</td>
<td>67.80%</td>
<td>76.59%</td>
<td>70.73%</td>
<td>70.24%</td>
<td>82.23%</td>
</tr>
<tr>
<td>cow</td>
<td>86.59%</td>
<td>82.68%</td>
<td>94.39%</td>
<td>75.12%</td>
<td>62.44%</td>
<td>86.83%</td>
<td>86.34%</td>
<td>82.06%</td>
</tr>
<tr>
<td>dog</td>
<td>34.63%</td>
<td>62.44%</td>
<td>74.39%</td>
<td>72.20%</td>
<td>66.34%</td>
<td>81.95%</td>
<td>82.93%</td>
<td>87.84%</td>
</tr>
<tr>
<td>horse</td>
<td>32.68%</td>
<td>58.78%</td>
<td>70.98%</td>
<td>77.80%</td>
<td>77.32%</td>
<td>84.63%</td>
<td>84.63%</td>
<td>69.55%</td>
</tr>
<tr>
<td>cup</td>
<td>79.76%</td>
<td>66.10%</td>
<td>77.80%</td>
<td>96.10%</td>
<td>96.10%</td>
<td>99.76%</td>
<td>99.02%</td>
<td>87.81%</td>
</tr>
<tr>
<td>car</td>
<td>62.93%</td>
<td>98.29%</td>
<td>77.56%</td>
<td>100.0%</td>
<td>97.07%</td>
<td>99.51%</td>
<td>100.0%</td>
<td>90.77%</td>
</tr>
<tr>
<td>total</td>
<td>64.85%</td>
<td>79.79%</td>
<td>82.23%</td>
<td>83.41%</td>
<td>82.99%</td>
<td>86.40%</td>
<td>86.40%</td>
<td>80.87%</td>
</tr>
</tbody>
</table>

Dataset: ETH-80, by B. Leibe, 2003
<table>
<thead>
<tr>
<th>Category</th>
<th>Primary feature(s)</th>
<th>Secondary feature(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>PCA Gray</td>
<td>Texture $D_xD_y$</td>
</tr>
<tr>
<td>pear</td>
<td>PCA Gray / Masks</td>
<td></td>
</tr>
<tr>
<td>tomato</td>
<td>Color</td>
<td>Texture Mag-Lap</td>
</tr>
<tr>
<td>cow</td>
<td>Texture Mag-Lap</td>
<td>Contour / Color</td>
</tr>
<tr>
<td>dog</td>
<td>Contour</td>
<td></td>
</tr>
<tr>
<td>horse</td>
<td>Contour</td>
<td></td>
</tr>
<tr>
<td>cup</td>
<td>Contour</td>
<td>PCA Gray / Masks</td>
</tr>
<tr>
<td>car</td>
<td>PCA Masks / Contour</td>
<td>Texture $D_xD_y$</td>
</tr>
</tbody>
</table>

Dataset: ETH-80, by B. Leibe, 2003
What we have learned today?

- Introduction
- K-nearest neighbor algorithm
- A simple Object Recognition pipeline