Lecture 16: Introduction to Object Recognition and the Nearest Neighbor Approach

Professor Fei-Fei Li
Stanford Vision Lab
What we will learn today?

- Introduction
- 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

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
Applications of computer vision

Computational photography

Assistive technologies

Surveillance

Security

Assistive driving
Categorization vs Single instance recognition

Does this image contain the Chicago Macy building’s?
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 are capable 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
Challenges: scale
Challenges: deformation
Challenges: occlusion

Magritte, 1957
Challenges: background clutter
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 \( y = f(x) \).
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

\[ Dist(X, Y) = \sqrt{\sum_{i=1}^{D} (X_i - Y_i)^2} \]
1-nearest neighbor

Distance measure - Euclidean

\[ Dist(X,Y) = \sqrt{\sum_{i=1}^{D} (X_i - Y_i)^2} \]
3-nearest neighbor

Distance measure - Euclidean

$$Dist(X, Y) = \sqrt{\sum_{i=1}^{D} (X_i - Y_i)^2}$$
5-nearest neighbor

Distance measure - Euclidean

\[ Dist(X, Y) = \sqrt{\sum_{i=1}^{D} (X_i - Y_i)^2} \]
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
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!
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)

\[
\begin{array}{cccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\quad \text{vs} \quad
\begin{array}{cccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array}
\]

\[
d = 1.4142
\]

Fei-Fei Li

Lecture 16
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

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

\[d = 1.4142\] vs \[d = 1.4142\]
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

• 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 \((0.001)^{1/d}\)
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?

Slide credit: L. Lazebnik
Generalization

- **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
No Free Lunch Theorem

© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com

Slide credit: D. Hoiem
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
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

Slide credit: D. Hoiem
How to reduce variance?

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

\[ f(\text{apple}) = \text{“apple”} \]
\[ f(\text{tomato}) = \text{“tomato”} \]
\[ f(\text{cow}) = \text{“cow”} \]

Dataset: ETH-80, by B. Leibe

Slide credit: L. Lazebnik
A simple pipeline

Training

Training Images

Training Images

Testing

Test Image

Image Features

Training

Training Labels

Learned Classifier

Image Features

Learned Classifier

Prediction

Dataset: ETH-80, by B. Leibe

Slide credit: D. Hoiem, L. Lazebnik
A simple pipeline

Training

Training Images

Image Features

Training Labels

Training

Learned Classifier

Testing

Test Image

Image Features

Learned Classifier

Prediction

Dataset: ETH-80, by B. Leibe
Slide credit: D. Hoiem, L. Lazebnik
Image features

Input image

Color: Quantize RGB values

<table>
<thead>
<tr>
<th>Invariance?</th>
</tr>
</thead>
<tbody>
<tr>
<td>? Translation</td>
</tr>
<tr>
<td>? Scale</td>
</tr>
<tr>
<td>? Rotation</td>
</tr>
<tr>
<td>? Occlusion</td>
</tr>
</tbody>
</table>
Image features

Input image

Color: Quantize RGB values

Invariance?

Translation

Scale

Rotation

Occlusion
Image features

Input image

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

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

**Input image**

<table>
<thead>
<tr>
<th>Color: Quantize RGB values</th>
<th>Invariance?</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Color bar" /></td>
<td>☑ Translation</td>
</tr>
<tr>
<td></td>
<td>☑ Scale</td>
</tr>
<tr>
<td></td>
<td>☑ Rotation</td>
</tr>
<tr>
<td></td>
<td>☛ Occlusion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Global shape: PCA space</th>
<th>Invariance?</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="PCA space" /></td>
<td>☛ Translation</td>
</tr>
<tr>
<td></td>
<td>❔ Scale</td>
</tr>
<tr>
<td></td>
<td>☑ Rotation</td>
</tr>
<tr>
<td></td>
<td>☛ 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
- Occlusion
Image features

Input image

Color: Quantize RGB values

Global shape: PCA space

Local shape: shape context

Invariance?
Translation
Scale
Rotation
Occlusion

Invariance?
Translation
Scale
Rotation
Occlusion

Invariance?
Translation
Scale
Rotation
Occlusion

Invariance?
Translation
Scale
Rotation
Occlusion
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

? Occlusion

**Texture:** Filter banks

Invariance?

? Translation

? Scale

? Rotation

? Occlusion
### 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
  - 😐 Occlusion

**Texture:** Filter banks

- **Invariance?**
  - 😐 Translation
  - 😐 Scale
  - 😐 Rotation
  - 😐 Occlusion
A simple pipeline

Training

- Training Images
- Training Labels
- Image Features
- Training
- Learned Classifier

Testing

- Test Image
- Image Features
- Learned Classifier
- Prediction

Dataset: ETH-80, by B. Leibe
Slide credit: D. Hoiem, L. Lazebnik

Fei-Fei Li
Lecture 16
Classifiers: Nearest neighbor

Training examples from class 1

Training examples from class 2

Slide credit: L. Lazebnik
A simple pipeline

Training

Training Images

Training Images

Image Features

Training

Training Labels

Learned Classifier

Testing

Test Image

Image Features

Learned Classifier

Prediction

Dataset: ETH-80, by B. Leibe  Slide credit: D. Hoiem, L. Lazebnik
Classifiers: Nearest neighbor

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

Slide credit: L. Lazebnik
## Results

<table>
<thead>
<tr>
<th></th>
<th>Color</th>
<th>$D_x$</th>
<th>$D_y$</th>
<th>Mag-Lap</th>
<th>PCA Masks</th>
<th>PCA Gray</th>
<th>Cont. Greedy</th>
<th>Cont. DynProg</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>57.56%</td>
<td>85.37%</td>
<td></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>76.59%</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>75.12%</td>
<td>99.76%</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>72.20%</td>
<td>62.44%</td>
<td>84.63%</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>77.80%</td>
<td>66.34%</td>
<td>81.95%</td>
<td>82.93%</td>
<td>82.93%</td>
<td>67.84%</td>
</tr>
<tr>
<td>horse</td>
<td>32.68%</td>
<td>58.78%</td>
<td>70.98%</td>
<td>96.10%</td>
<td>84.63%</td>
<td>84.63%</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>99.76%</td>
<td>99.76%</td>
<td>99.02%</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>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>86.40%</td>
<td>80.87%</td>
</tr>
</tbody>
</table>

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

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 $D_x D_y$</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 $D_x D_y$</td>
</tr>
</tbody>
</table>
What we have learned today?

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