Introduction to recognition

Source: Charley Harper
Outline

• Overview of recognition tasks
• A statistical learning approach
• “Classic” or “shallow” recognition pipeline
  • “Bag of features” representation
  • Classifiers: nearest neighbor, linear, SVM
• After that: neural networks, “deep” recognition pipeline
Common recognition tasks

Adapted from Fei-Fei Li
Image classification and tagging

- outdoor
- mountains
- city
- Asia
- Lhasa
- ...

Adapted from Fei-Fei Li
Object detection

- find pedestrians

Adapted from Fei-Fei Li
Activity recognition

- walking
- shopping
- rolling a cart
- sitting
- talking
- ...

Adapted from Fei-Fei Li
Semantic segmentation

Adapted from Fei-Fei Li
Semantic segmentation

Adapted from Fei-Fei Li
Detection, semantic segmentation, instance segmentation

Image source
This is a busy street in an Asian city. Mountains and a large palace or fortress loom in the background. In the foreground, we see colorful souvenir stalls and people walking around and shopping. One person in the lower left is pushing an empty cart, and a couple of people in the middle are sitting, possibly posing for a photograph.
Image classification
The statistical learning framework

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

\[
f(\text{apple}) = \text{“apple”}
\]
\[
f(\text{tomato}) = \text{“tomato”}
\]
\[
f(\text{cow}) = \text{“cow”}
\]
The statistical 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 \( y = f(x) \).
“Classic” recognition pipeline

- Hand-crafted feature representation
- Off-the-shelf trainable classifier
“Classic” representation: Bag of features
Motivation 1: Part-based models

Motivation 2: Texture models

Motivation 3: Bags of words

Motivation 3: Bags of words

Motivation 3: Bags of words

- Orderless document representation: frequencies of words from a dictionary

Salton & McGill (1983)
Motivation 3: Bags of words

• Orderless document representation: frequencies of words from a dictionary  
  Salton & McGill (1983)
Bag of features: Outline

1. Extract local features
2. Learn “visual vocabulary”
3. Quantize local features using visual vocabulary
4. Represent images by frequencies of “visual words”
1. Local feature extraction

- Sample patches and extract descriptors
2. Learning the visual vocabulary

Extracted descriptors from the training set
2. Learning the visual vocabulary
2. Learning the visual vocabulary

Visual vocabulary

Clustering

Slide credit: Josef Sivic
Recall: K-means clustering

- Want to minimize sum of squared Euclidean distances between features \( x_i \) and their nearest cluster centers \( m_k \)

\[
D(X, M) = \sum_{\text{cluster } k} \sum_{\text{point } i \text{ in cluster } k} (x_i - m_k)^2
\]

**Algorithm:**
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each feature to the nearest center
  - Recompute each cluster center as the mean of all features assigned to it
Recall: Visual vocabularies

Source: B. Leibe
Bag of features: Outline

1. Extract local features
2. Learn “visual vocabulary”
3. Quantize local features using visual vocabulary
4. Represent images by frequencies of “visual words”
Spatial pyramids

Lazebnik, Schmid & Ponce (CVPR 2006)
Spatial pyramids

Lazebnik, Schmid & Ponce (CVPR 2006)
Spatial pyramids

Lazebnik, Schmid & Ponce (CVPR 2006)
Spatial pyramids

- Scene classification results

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (vocabulary size: 16)</th>
<th>Strong features (vocabulary size: 200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0 (1 × 1)</td>
<td>45.3 ± 0.5</td>
<td>56.2 ± 0.6</td>
</tr>
<tr>
<td>1 (2 × 2)</td>
<td>53.6 ± 0.3</td>
<td>64.7 ± 0.7</td>
</tr>
<tr>
<td>2 (4 × 4)</td>
<td>61.7 ± 0.6</td>
<td>66.8 ± 0.6</td>
</tr>
<tr>
<td>3 (8 × 8)</td>
<td>63.3 ± 0.8</td>
<td></td>
</tr>
</tbody>
</table>
Spatial pyramids

- Caltech101 classification results

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (16)</th>
<th>Strong features (200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0</td>
<td>15.5 ±0.9</td>
<td>41.2 ±1.2</td>
</tr>
<tr>
<td>1</td>
<td>31.4 ±1.2</td>
<td>55.9 ±0.9</td>
</tr>
<tr>
<td>2</td>
<td>47.2 ±1.1</td>
<td>63.6 ±0.9</td>
</tr>
<tr>
<td>3</td>
<td>52.2 ±0.8</td>
<td>60.3 ±0.9</td>
</tr>
</tbody>
</table>
“Classic” recognition pipeline

- Hand-crafted feature representation
- Off-the-shelf trainable classifier
Classifiers: Nearest neighbor

\[ f(x) = \text{label of the training example nearest to } x \]

All we need is a distance or similarity function for our inputs
No training required!
Functions for comparing histograms

- **L1 distance:**
  \[ D(h_1, h_2) = \sum_{i=1}^{N} |h_1(i) - h_2(i)| \]

- **\(\chi^2\) distance:**
  \[ D(h_1, h_2) = \sum_{i=1}^{N} \frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)} \]

- **Quadratic distance (cross-bin distance):**
  \[ D(h_1, h_2) = \sum_{i,j} A_{ij} (h_1(i) - h_2(j))^2 \]

- **Histogram intersection (similarity function):**
  \[ I(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i)) \]
K-nearest neighbor classifier

- For a new point, find the k closest points from training data
- Vote for class label with labels of the k points

$k = 5$
K-nearest neighbor classifier

Which classifier is more robust to outliers?

Credit: Andrej Karpathy, http://cs231n.github.io/classification/
K-nearest neighbor classifier

Left: Example images from the CIFAR-10 dataset. Right: first column shows a few test images and next to each we show the top 10 nearest neighbors in the training set according to pixel-wise difference.

Credit: Andrej Karpathy, http://cs231n.github.io/classification/
Linear classifiers

Find a *linear function* to separate the classes:

\[ f(x) = \text{sgn}(w \cdot x + b) \]
Visualizing linear classifiers

Nearest neighbor vs. linear classifiers

**NN pros:**
- Simple to implement
- Decision boundaries not necessarily linear
- Works for any number of classes
- *Nonparametric* method

**NN cons:**
- Need good distance function
- Slow at test time

**Linear pros:**
- Low-dimensional *parametric* representation
- Very fast at test time

**Linear cons:**
- Works for two classes
- How to train the linear function?
- What if data is not linearly separable?
Linear classifiers

- When the data is linearly separable, there may be more than one separator (hyperplane)
Support vector machines

• Find hyperplane that maximizes the margin between the positive and negative examples

\[ \text{x}_i \text{ positive (} y_i = 1 \text{): } \text{x}_i \cdot \text{w} + b \geq 1 \]
\[ \text{x}_i \text{ negative (} y_i = -1 \text{): } \text{x}_i \cdot \text{w} + b \leq -1 \]

For support vectors, \[ \text{x}_i \cdot \text{w} + b = \pm 1 \]

Distance between point and hyperplane:
\[ \frac{|\text{x}_i \cdot \text{w} + b|}{\|\text{w}\|} \]

Therefore, the margin is \[ 2 / \|\text{w}\| \]

Finding the maximum margin hyperplane

1. Maximize margin \( \frac{2}{\|w\|} \)

2. Correctly classify all training data:

   \( x_i \) positive \((y_i = 1)\): \( x_i \cdot w + b \geq 1\)

   \( x_i \) negative \((y_i = -1)\): \( x_i \cdot w + b \leq -1\)

**Quadratic optimization problem:**

\[
\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i (w \cdot x_i + b) \geq 1
\]

SVM parameter learning

- **Separable data:**
  \[
  \begin{align*}
  \min_{\mathbf{w}, b} & \quad \frac{1}{2} \| \mathbf{w} \|^2 \\
  \text{subject to} & \quad y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1
  \end{align*}
  \]

  - Maximize margin
  - Classify training data correctly

- **Non-separable data:**
  \[
  \min_{\mathbf{w}, b} \quad \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i (\mathbf{w} \cdot \mathbf{x}_i + b))
  \]

  - Maximize margin
  - Minimize classification mistakes
SVM parameter learning

\[
\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))
\]

Demo: [http://cs.stanford.edu/people/karpathy/svmjs/demo](http://cs.stanford.edu/people/karpathy/svmjs/demo)
Nonlinear SVMs

- **General idea:** the original input space can always be mapped to some higher-dimensional feature space where the training set is separable.

\[ \Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]

*Image source*
Nonlinear SVMs

- Linearly separable dataset in 1D:

  ![Linearly separable dataset in 1D](image)

- Non-separable dataset in 1D:

  ![Non-separable dataset in 1D](image)

- We can map the data to a higher-dimensional space:

  ![Higher-dimensional space](image)

Slide credit: Andrew Moore
The kernel trick

• **General idea:** the original input space can always be mapped to some higher-dimensional feature space where the training set is separable

• **The kernel trick:** instead of explicitly computing the lifting transformation $\varphi(x)$, define a kernel function $K$ such that

$$K(x, y) = \varphi(x) \cdot \varphi(y)$$

(to be valid, the kernel function must satisfy **Mercer’s condition**
The kernel trick

- Linear SVM decision function:

\[ \mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \]

The kernel trick

- Linear SVM decision function:

\[ w \cdot x + b = \sum_i \alpha_i y_i x_i \cdot x + b \]

- Kernel SVM decision function:

\[ \sum_i \alpha_i y_i \varphi(x_i) \cdot \varphi(x) + b = \sum_i \alpha_i y_i K(x_i, x) + b \]

- This gives a nonlinear decision boundary in the original feature space

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998
Polynomial kernel: \( K(x, y) = (c + x \cdot y)^d \)
Gaussian kernel

- Also known as the radial basis function (RBF) kernel:

\[
K(x, y) = \exp\left(-\frac{1}{\sigma^2} \|x - y\|^2\right)
\]
Gaussian kernel

SV’s
Kernels for bags of features

- **Histogram intersection:**
  \[ K(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i)) \]

- **Square root (Bhattacharyya kernel):**
  \[ K(h_1, h_2) = \sum_{i=1}^{N} \sqrt{h_1(i)h_2(i)} \]

- **Generalized Gaussian kernel:**
  \[ K(h_1, h_2) = \exp \left( -\frac{1}{A} D(h_1, h_2)^2 \right) \]

  - \( D \) can be L1 distance, Euclidean distance, \( \chi^2 \) distance, etc.
SVMs: Pros and cons

**Pros**
- Kernel-based framework is very powerful, flexible
- Training is convex optimization, globally optimal solution can be found
- Amenable to theoretical analysis
- SVMs work very well in practice, even with very small training sample sizes

**Cons**
- No “direct” multi-class SVM, must combine two-class SVMs (e.g., with one-vs-others)
- Computation, memory (esp. for nonlinear SVMs)