RECOGNITION

Thanks to Svetlana Lazebnik and Andrew Zisserman for the use of some slides.
How many categories?
Variability makes recognition hard

Camera position
Illumination
Shape parameters

Within-class variations?
Variations within the same class
History

1960s – early 1990s: geometry
1990s: appearance
Mid-1990s: sliding window
Late 1990s: local features
Early 2000s: parts-and-shape models
Mid-2000s: bags of features
Present trends: data-driven methods, context
2D objects
Eigenfaces (Turk & Pentland, 1991)

<table>
<thead>
<tr>
<th>Experimental Condition</th>
<th>Correct/Unknown Recognition Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forced classification</td>
<td>96/0, 85/0, 64/0</td>
</tr>
<tr>
<td>Forced 100% accuracy</td>
<td>100/19, 100/39, 100/60</td>
</tr>
<tr>
<td>Forced 20% unknown rate</td>
<td>100/20, 94/20, 74/20</td>
</tr>
</tbody>
</table>
Local features

Parts-and-shape models

Model:

- Object as a set of parts
- Relative locations between parts
- Appearance of part

Figure from [Fischler & Elschlager 73]
Constellation models

Objects as texture

• All of these are treated as being the same - no segmentation (background, foreground)
Global scene descriptors

- The “gist” of a scene: Oliva & Torralba (2001)

http://people.csail.mit.edu/torralba/code/spatialenvelope/
Data-driven methods

J. Hays and A. Efros, Scene Completion using Millions of Photographs, SIGGRAPH 2007
Data-driven methods

J. Tighe and S. Lazebnik, ECCV 2010
Overview

• Basic recognition tasks
• A statistical learning approach
• Traditional or “shallow” recognition pipeline
  • Bags of features
  • Classifiers
• Currently best approaches: neural networks and “deep” recognition pipeline
RECOGNITION AS AN APPLICATION OF MACHINE LEARNING
Image classification

- outdoor/indoor
- city/forest/factory/etc.
Image tagging

• street
• people
• building
• mountain
• …
Object detection

- find pedestrians
Activity recognition

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

[Image: A busy street scene with a list of activities written on it.]
Image parsing

- sky
- mountain
- building
- tree
- banner
- building
- street lamp
- market
- people
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:

\[
\begin{align*}
  f(\text{apple}) &= \text{“apple”} \\
  f(\text{tomato}) &= \text{“tomato”} \\
  f(\text{cow}) &= \text{“cow”}
\end{align*}
\]
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 \( y = f(x) \).
Traditional recognition pipeline

- Features are not learned
- Trainable classifier is often generic (e.g. SVM)
Bags of features
Traditional features: Bags-of-features

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

1. Local feature extraction

- Sample patches and extract descriptors
Keypoints

patches surrounding keypoints
2. Learning the visual vocabulary

Extracted descriptors from the training set

patches surrounding keypoints

Slide credit: Josef Sivic except for the image patches
2. Learning the visual vocabulary

Slide credit: Josef Sivic
2. Learning the visual vocabulary

Visual vocabulary

Clustering

Slide credit: Josef Sivic
Review: 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
Example visual vocabulary

Source: B. Leibe

Appearance codebook
Bag-of-features steps

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

Image Pixels → Hand-designed feature extraction → Trainable classifier → Object Class
Classifiers: Nearest neighbor

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

All we need is a distance function for our inputs
No training required!
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

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?
Support vector machines

• When the data is linearly separable, there may be more than one separator (hyperplane)

Which separator is best?
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

For support vectors:

\[ x_i \cdot w + b = \pm 1 \]

Distance between point and hyperplane:

\[ \frac{|x_i \cdot w + b|}{||w||} \]

Therefore, the margin is

\[ 2 / ||w|| \]

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

Maximize margin

Classify training data correctly

• Non-separable data:

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

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.

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

Input Space  Feature Space
Nonlinear SVMs

• Linearly separable dataset in 1D:

• Non-separable dataset in 1D:

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

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

• Kernel SVM decision function:

\[ \sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \]

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

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 histograms

• Histogram intersection:

\[ K(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i)) \]

• Square root (Bhattacharyya kernel):

\[ K(h_1, h_2) = \sum_{i=1}^{N} \sqrt{h_1(i)h_2(i)} \]
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)
Generalization

• Generalization refers to the ability to correctly classify never before seen examples
• Can be controlled by turning “knobs” that affect the complexity of the model
Diagnosing generalization ability

- **Training error**: how does the model perform on the data on which it was trained?
- **Test error**: how does it perform on never before seen data?

![Graph showing underfitting and overfitting](source: D. Hoiem)

Underfitting

Overfitting

Training error

Test error

Error

Model complexity

Low

High
Underfitting and overfitting

- **Underfitting**: training and test error are both *high*
  - Model does an equally poor job on the training and the test set
  - Either the training procedure is ineffective or the model is too “simple” to represent the data

- **Overfitting**: Training error is *low* but test error is *high*
  - Model fits irrelevant characteristics (noise) in the training data
  - Model is too complex or amount of training data is insufficient
Effect of training set size

- Many training examples
- Few training examples
- Model complexity: High

Test Error

Source: D. Hoiem
Validation

- Split the data into **training**, **validation**, and **test** subsets
- Use training set to **optimize model parameters**
- Use validation test to **choose the best model**
- Use test set only to **evaluate performance**
Summary

The different steps

Manually gathered training images → Visual words → Learn a visual category model → Evaluate classifier/detector → Test images