Outline

- **Overview**: recognition tasks
- **Statistical learning approach**
- **Classic / Shallow Pipeline**
  - “Bag of features” representation
  - Classifiers: nearest neighbor, linear, SVM
- **Deep Pipeline**
  - Neural Networks
Common Recognition Tasks

Adapted from Fei-Fei Li
Image Classification and Tagging

What is this an image of?

- outdoor
- mountains
- city
- Asia
- Lhasa

Adapted from Fei-Fei Li
Object Detection

Localize!

Adapted from Fei-Fei Li
Activity Recognition

What are they doing?

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

Label Every Pixel
Semantic Segmentation

Label Every Pixel

Adapted from Fei-Fei Li
Detection, semantic and 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
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 labeled *training set* 
\[ \{(x_1, y_1), \ldots, (x_N, y_N)\} \]

Learn the prediction function \( f \), by minimizing prediction error on *training set*

**Testing**
Given unlabeled *test instance* 
\( x \)

Predict the output label \( y \) as 
\[ y = f(x) \]
Steps

Training

- Training Images
  - Training Images (images of objects like apples and dogs)

Testing

- Test Image
  - Test Image (an apple)

1. Image Features
2. Training
3. Learned model
4. Learned model
5. Prediction
6. “apple”
"Classic" recognition pipeline

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

Example 2: Texture models

Example 3: Bags of words

Example 3: Bags of words

Orderless document representation: frequencies of words from a dictionary

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

Example 3: Bags of words

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

Slide credit: Josef Sivic
2. Learning the visual vocabulary

Slide credit: Josef Sivic
2. Learning the visual vocabulary

Visual vocabulary

Clustering

Slide credit: Josef Sivic
K-means clustering

**Goal:** 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
Visual vocabularies

Source: B. Leibe
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>Weak features (vocabulary size: 16)</th>
<th>Strong features (vocabulary size: 200)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>Single-level</td>
</tr>
<tr>
<td>0 (1 × 1)</td>
<td>45.3 ±0.5</td>
</tr>
<tr>
<td>1 (2 × 2)</td>
<td>53.6 ±0.3</td>
</tr>
<tr>
<td>2 (4 × 4)</td>
<td>61.7 ±0.6</td>
</tr>
<tr>
<td>3 (8 × 8)</td>
<td>63.3 ±0.8</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>32.8 ±1.3</td>
</tr>
<tr>
<td>1</td>
<td>31.4 ±1.2</td>
<td>49.3 ±1.4</td>
</tr>
<tr>
<td>2</td>
<td>47.2 ±1.1</td>
<td><strong>54.0 ±1.1</strong></td>
</tr>
<tr>
<td>3</td>
<td>52.2 ±0.8</td>
<td></td>
</tr>
</tbody>
</table>
“Classic” recognition pipeline

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

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

- All we need is a distance or similarity function for our inputs
- No training required!
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/
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

What is the label for $x$?
Quiz: K-nearest neighbor classifier

Which classifier is more robust to outliers?

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

Visualizing linear classifiers

Nearest neighbor vs. linear classifiers

Nearest Neighbors

- **Pros:**
  - Simple to implement
  - Complex decision boundaries
  - Works for any number of classes
  - *Nonparametric* method

- **Cons:**
  - Need good distance function
  - Slow at test time

Linear Models

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

- **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

Hyperplane “supported” by least # examples, in 2D this would be 3 “support” vectors

Which separator is best?
Support Vector Machines

Using complex features, decision boundary in original space can be complex.
• Goal: obtain a classifier with **good generalization** or performance on never before seen data

1. Learn *parameters* on the **training set**
2. Tune *hyperparameters* (implementation choices) on the **held out validation set**
3. Evaluate performance on the **test set**
   – Crucial: do not peek at the test set when iterating steps 1 and 2!
Bias-variance tradeoff

- Prediction error of learning algorithms has two main components:
  - **Bias**: error due to simplifying model assumptions
  - **Variance**: error due to randomness of training set

- **Bias-variance tradeoff** can be controlled by turning “knobs” that determine model complexity
Underfitting and overfitting

**Underfitting:** training and test error are both *high*
- Model does an equally poor job on the training and the test set
- The model is too “simple” to represent the data or the model is not trained well

**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