Introduction to Recognition

Computer Vision

CS 543 / ECE 549

University of Illinois

Many Slides from D. Hoiem, L. Lazebnik.
Outline

• Overview of image and region categorization
  – Task description
  – What is a category

• Example of spatial pyramids bag-of-words scene categorizer

• Key concepts: features and classification

• Deep convolutional neural networks (CNNs)
Recognition as 3D Matching

Recognizing solid objects by alignment with an image. Huttenlocher and Ullman IJCV 1990.

“Instance” Recognition

“Category-level” Recognition
Detection, semantic segmentation, instance segmentation

Image source
“Classic” recognition pipeline

Image Pixels → Feature representation → Trainable classifier → Class label
Overview

Training

Training Images

Image Features

Classifier Training

Training Labels

Trained Classifier

Testing

Test Image

Image Features

Trained Classifier

Prediction

Outdoor
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!
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:

\[ f(x) = \text{sgn}(w \cdot x + b) \]
Nonlinear SVMs

• Linearly separable dataset in 1D:

• Non-separable dataset in 1D:

• We can map the data to a *higher-dimensional space*:
Bag 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”
Digit Classification Case Study
The MNIST DATABASE of handwritten digits
Yann LeCun & Corinna Cortes

• Has a training set of 60 K examples (6K examples for each digit), and a test set of 10K examples.
• Each digit is a 28 x 28 pixel grey level image. The digit itself occupies the central 20 x 20 pixels, and the center of mass lies at the center of the box.
Figure 1: Comparison of kernel SVM for various training size on the full training set (60,000 examples). Using the gradient features the error rates are 0.79% using intersection kernel and 1.44% using linear kernel SVM. The performance using the raw pixel is 1.41% using rbf and 1.34% using the polynomial kernels. The gradient features perform better using the linear and intersection kernels compared to rbf and polynomial kernels significantly when the number of training data is small suggesting that the gradient features capture the invariances in the digits quite well. We did not train the polynomial and rbf kernel SVMs on the gradient features as both the training and test time were very high.

Table 5: Summary of various results on the USPS dataset. Both the linear and the intersection kernel SVMs outperform the existing numbers using SVMs which is at 4%. The VSV method which iterates the Support Vectors to create additional training examples, and re-train a SVM, lead to an improved accuracy of 3.2%. Using polynomial and rbf kernel SVM on PHOG features reduces the error rate even further to 3.2% and 2.7% respectively. Some of the results shown in * use a different training dataset which has been enhanced by adding machine-printed characters. Note that our numbers are the best in the unmodified version of the dataset.
Bias and Variance
Bias-Variance Trade-off

Performance as a function of model complexity (SVM)
Model Selection
Bias-Variance Trade-off

As a function of dataset size
Generalization Error

Fixed classifier

Error

Generalization Error

Number of Training Examples
Features vs Classifiers

Figure 1: Comparison of kernel SVM for various training size

Performance on MNIST Dataset

Table 5: Summary of various results on the USPS dataset. Both the linear and the intersection kernel SVMs outperform the existing numbers using SVMs which is at 4%. The VSV method which hits the Support Vectors to create additional training examples, and re-trains a SVM, leads to an improved accuracy of 3.2%. Using polynomial and Rbf kernel SVM on PHOG features reduces the error rate even further to 3.2% and 2.7% respectively. Some of the results shown in * use a different training dataset which has been enhanced by adding machine-printed characters. Note that our numbers are the best in the unmodified version of the dataset.
What are the right features?

Depend on what you want to know!

• Object: shape
  – Local shape info, shading, shadows, texture

• Scene: geometric layout
  – Linear perspective, gradients, line segments

• Material properties: albedo, feel, hardness
  – Color, texture

• Action: motion
  – Optical flow, tracked points
Stuff vs Objects

- recognizing cloth fabric vs recognizing cups
Feature Design Process

1. Start with a model
2. Look at errors on development set
3. Think of features that can improve performance
4. Develop new model, test whether new features help.
5. If not happy, go to step 1.
6. “Ablations”: Simplify system, prune out features that don’t help anymore in presence of other features.
Features vs Classifiers

Figure 1: Comparison of kernel SVM for various training size.

- Using the gradient features, the error rates are 0.79% using the intersection kernel and 1.44% using the linear kernel SVM.
- The performance using the raw pixel is 1.41% using the rbf kernel and 1.34% using the polynomial kernels.

Table 5: Summary of various results on the USPS dataset.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Classifier</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw Pixels</td>
<td>SVM (linear)</td>
<td>11.3%</td>
</tr>
<tr>
<td>PHOG</td>
<td>SVM (linear)</td>
<td>3.4%</td>
</tr>
<tr>
<td>PHOG</td>
<td>SVM (intersection)</td>
<td>3.4%</td>
</tr>
<tr>
<td>PHOG</td>
<td>SVM (poly, (d=5))</td>
<td>3.2%</td>
</tr>
<tr>
<td>Human Error</td>
<td>SVM (rbf, (\gamma=0.1))</td>
<td>2.7%</td>
</tr>
<tr>
<td>Raw Pixels</td>
<td>Boosted Neural Nets ([8])</td>
<td>2.6%</td>
</tr>
<tr>
<td>Human Error</td>
<td>SVM (rbf, (\gamma=0.1))</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

Both the linear and the intersection kernel SVMs outperform the existing numbers using SVMs, which is at 4%. The VSS method, which iterates the Support Vectors to create additional training examples, and re-trains a SVM, leads to an improved accuracy of 3.2%. Using polynomial and rbf kernel SVMs on the gradient features reduces the error rate even further to 3.2% and 2.7% respectively. Some of the results shown in * use a different training dataset which has been enhanced by adding machine-printed characters. Note that our numbers are the best in the unmodified version of the dataset.
“Classic” recognition pipeline

Image Pixels → Feature representation → Trainable classifier → Class label
Categorization involves **features** and a classifier.

- **Training**
  - Training Images
  - Training Labels
  - Image Features
  - Classifier Training
  - Trained Classifier

- **Testing**
  - Test Image
  - Image Features
  - Trained Classifier
  - Prediction: Outdoor
New training setup with moderate sized datasets

- **Training Images**
- **Training Labels**
- **Tune CNN features and Neural Network classifier**
- **Trained Classifier**

- **Dataset similar to task with millions of labeled examples**

- Initialize CNN Features
Categorization involves **features** and a classifier.

**Training**

- Training Images
- Image Features
- Classifier Training
- Trained Classifier

**Testing**

- Test Image
- Image Features
- Trained Classifier
- Prediction: Outdoor
New training setup with moderate sized datasets

Dataset similar to task with millions of labeled examples

- Training Images
- Training Labels
- Tune CNN features and Neural Network classifier
- Trained Classifier

Initialize CNN Features