Learning and Classification

a Really brief introduction

Motivation: almost all computer vision algorithms use some learning.

Classification (Labeling) tasks

Many vision tasks are decisions between few options:

- Edge or not (in some point p)?
- Is there a person in the given image?
- Are two pixels, p1, p2, in the same segment?
- Does point p1 in Image1, corresponds to p2 in Image2?

How to formalize this decision procedure?

Classification: \( \phi(X) \rightarrow Y \)

- \( X \) - image data (feature vector).
- \( y \) - class.

Classification: partition the \( X \) space into regions ...
Probabilistic models

Let $x$ be the measurement. $x$ depends on:
- A discrete variable (label) $Y$ of interest (e.g. edge/non-edge)
- Other factors

How to express the variation?
A popular choice: probabilistic modeling

- $x$ is modeled as a random variable (r.v.) distributed by $p_y(x)$

- Given a distribution, we can make a Bayesian decision.

Example (board): height distributions, uncertainty.

Learning from examples - Generative

Learning: generating a classifier from examples

The Generative approach
- Training: Estimate likelihood/prior functions from data
- Classification (“Testing”): Use Bayesian decision

Examples: (different ways to represent likelihoods)
- Gaussian mixture models
- Bayesian networks

(-) Need model of the distribution
(+ ) Provides decision certainty, allows varying priors
Learning from examples - Discriminative

The discriminative approach
- Training: Direct estimation of the decision function or the posterior probability from examples $\phi(X)$ or $P(Y|X)$
- Classification: Use the decision function ...

Many ways to represent the decision function: Perceptrons, Support vectors machine, k-nearest neighbors, Decision trees, Ada-boost, (Deep/convolutional) neural networks, ...

(+ ) Does not need a distribution model
(+ ) When there is no model, usually works better.
(+ ) Does not learn what is not essential

Discriminative Learning

Given a set of training examples, how should we choose the classification function $\phi(X) \rightarrow Y$?
Equivalently how should we choose the partition?

What do we want?
- Low training error rate - Classify all training examples correctly (consistency).
- Low test error rate - Classify unseen (test) examples correct (generalization).

As a general rule:
- Simple classifier - low consistency, better generalization, work better with a few training examples
- Complex classifier - high consistency, worse generalization, needs many examples
Discriminative Learning Algorithms

Different algorithms: Different methods for specifying implicitly/explicitly the classification function

Some Examples

Discriminative Learning - I

- Linear decision functions
  \[ D(x) = a^T x + b \geq 0 \implies '+' \]
  = linear separation

- Training = finding a,b
- Trained from examples by
  - Perceptron algorithm (board?)
  - Linear programming
  - Logistic regression ...

\[ x_2 \]
\[ x_1 \]
Discriminative Learning - I

- Which linear separation is best?
- Which linear separation leads to the best generalization?
- (Linear) SVM - Support Vector Machines
  Choose the separation that maximizes the margin
  - It makes the “effective dimension” smallest
  - Training is an convex optimization process that maximizes margin while satisfying consistency.

Discriminative Learning - II

What if the space is not linearly separable?

Change the space by explicit feature generation (e.g. Lifting to higher dimension)

Kernel SVM uses inner products to implicitly lift to higher dimension. (comp. expensive)
Discriminative Learning - III

Decision tree
• Old, established, fast
• Best for symbolic features but not only

Training involves finding the thresholded feature and the threshold in each node.

Discriminative Learning - IV

1-nearest neighbors
• “Lazy” learning
• Heuristic but theoretically, its asymptotic error is just twice that of Bayesian

• Training is not needed.
Discriminative Learning

3-nearest neighbors

Discriminative Learning - VI

Decision by committee/ensemble
  • Many "weak classifiers" cast their votes
  • Which weak classifiers? How to merge their votes?
  • "Forests" of decision trees
    - Train trees of subsets of data and subsets of features
    - Classify by voting results

The Adaboost algorithm - Adaptive Boosting
  • Principle - choose independent weak classifiers
  • Change the weight of examples during learning
  • Final classifier = weight sum of weak classifier
AdaBoost example

- Adaboost starts with a uniform distribution of “weights” over training examples.
- Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)
- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.

\[
\begin{align*}
    a_i &= \log \frac{1 - \epsilon_i}{\epsilon_i} \\
    h_{\text{strong}}(x) &= \begin{cases} 
    1 & a_1 h_1(x) + \cdots + a_n h_n(x) \geq \frac{1}{2} (a_1 + \cdots + a_n) \\
    0 & \text{otherwise}
    \end{cases}
\end{align*}
\]

Source: Qing Chen, Discover Lab, University of Ottawa

Feature Selection in Learning

- Many algorithms, no clear winner (until recently)
- The characterization of the class (feature vector) is often more important than the algorithm!
  - Raw feature (e.g. image intensities) are not effective.
- How to select/construct features?
  - Traditionally, hand picked, or selected based on criteria such as information gain, independently of the algorithm.
  - More recently, features are selected in a way integrate with learning: Kernel SVM, boosting, deep neural networks.
Neural network perceptron

- Recall the linear decision functions / perceptron
  \[ D(x) = a^T x + b \geq 0 \implies + \]

- Using other notations, and graphical description
  \[ \varphi(w, x) = \text{sign}\left(\sum_i w_i x_i + b\right) \]

- The perceptron is the simplest neural network

Motivation from biology

Rough description of the brain:
- Many neural cells 10,000,000,000
- Every neural cell collect electrical/chemical activities from others using dendrites, and if the sum is large enough, fires ...
- Then other neurons may respond to its activity
Multilayer perceptron

The main idea:
• Build several perceptrons and use their output as inputs to another set of perceptrons
• Training – adjusting all the weights together so that, given an example’s input X, the network’s output is Y.

• Training by gradient descent (backpropagation alg.)
• Feedforward
• Fully connected
• Several/many hidden layers
• Theoretically: can be any function

Multilayer perceptron

The main difficulty is computational
• With many inputs, wide hidden layers of size W, the # connections is $W^2$.
• Training is an optimization of a nonlinear functions, with many local minima.

Recently however
• Simpler networks (locally connected = convolutional)
• Different nonlinearities & better training algorithms
• Stronger computers
• Richer data
made learning feasible
Sketch Tokens: A Learned Mid-level Representation for Contour and Object Detection

Outline

- Introduction
- Method
- Results
- Discussion
Introduction

- Image boundaries may be identified by non-local information
- Typical neighborhoods *(sketch tokens)* are straight lines, T-junctions, Y-junctions, corners, curves, parallel lines, etc.

Method

- Defining sketch token classes
- Detecting sketch tokens
Defining sketch token classes

- Consider ground truth boundary marking in Berkeley DB
- Consider all 35x35 neighborhoods with center=boundary
- Cluster them to 150 classes

Detecting sketch tokens

1. Feature extraction
2. Classification
Feature extraction

- For every pixel, define 14 channels: color (3), gradient magnitude in 3 scales (3), and oriented gradient (2 scale, 4 orientations) (8)

- Characterizing 35x35 patch
  Two types of features are then employed:
  1. 35x35x14 features = all channels in all pixels
  2. self-similarity features

- The self-similarity features capture the portions of an image patch that contain similar textures based on color or gradient information.
- Divide the patch into 25 7x7 “cells”
- For channel $k$ and grid cells $i$ and $j$, we define the self-similarity feature $f_{ijk}$ as:

  $$f_{ijk} = S_{jk} - S_{ik}$$

- where $S_{jk}$ is the sum of values in cell $j$ in channel $k$. An illustration of self-similarity features is shown in Fig. 3.
Classification

- Classifier should be fast and multi class

Random forest
- Training: Build K trees
  - In every node use the best features out of $\sqrt{F}$ selected randomly.
  - Train the tree until leafs are pure or contain just a few examples
- Run the K trees and take majority

Figure 3. Illustration of the self-similarity features: The $L_1$ distance $\sum_k |f_{ijk}|$ from the anchor cell (yellow box) to the other $5 \times 5$ cells are shown for color and gradient magnitude channels. The original patch is shown to the left.
Which channels are used?

Figure 4. Frequency of example features being selected by the random forest: (first row) color channels, (second row) gradient magnitude channels, (third row) selected orientation channels.

Figure 5. Illustration of the sketch token responses for four tokens. Notice the high selectivity of each sketch token (best viewed in color.)
Figure 10. Examples of contour detection on the BSDS500 [1]. For Sketch Tokens we define edge strength according to Equation 2 and apply smoothing and standard non-maximal suppression to obtain peak edge responses [3]. Note how our method captures finer details such as the structure of Sydney Opera House on the 1st row and human legs on the 2nd row.
Results for boundary detection

Figure 6. Precision/Recall curve for contour detection. Note that while our method achieves similar F-measure (ODS) to gPb-own-term [1] and SCG [24], we achieve improved results with both low and high recall. This results in high average precision (AP) scores.
Example for Learning based vision

Recovering Surface Layout from a Single Image

D. Hoiem, A.A. Efros, M. Hebert
Robotics Institute, CMU

Original Presentation by Derek Hoiem
Modified and shortened

Motivation: “Where” for “What”
Motivation: Detection without context

How to represent scene space?

- Depth Map

Saxena, Chung & Ng 2005, 2007
How to represent scene space?

Gibson’s *Surface Layout*: “The elementary impressions of a visual world are those of surface and edge.”

*The Perception of the Visual World* (1950)
How to represent scene space?

Marr's 2½-D Sketch

Surface Layout (this paper)

Goal: Label image into 7 Geometric Classes:
- Support
- Vertical
  - Planar: facing Left (←), Center ( ), Right (→)
  - Non-planar: Solid (X), Porous or wiry (O)
- Sky
Our Main Challenge

- Recovering 3D geometry from *single* 2D projection
- Infinite number of possible solutions!

Our World is Structured

Image Credit (left): F. Cunin
and M.J. Salor, UCSID
1. Use All Available Cues

- Use All Available Cues
- Color, texture, image location
- Vanishing points, lines
- Texture gradient

<table>
<thead>
<tr>
<th>Surface Cues</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Location and Shape</strong></td>
</tr>
<tr>
<td>1. Location: normalized x and y, n - 1</td>
</tr>
<tr>
<td>2. Location: normalized x and y, 10th and 90th percentile</td>
</tr>
<tr>
<td>3. Location: normalized y at estimated horizon, [10th, 90th] percentile</td>
</tr>
<tr>
<td>4. Location: whether segment is above, below, or in middle of estimated horizon</td>
</tr>
<tr>
<td>5. Shape: number of corners in segment</td>
</tr>
<tr>
<td>6. Shape: normalized area in image</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1: RGB values: mean</td>
</tr>
<tr>
<td>C2: HSV values: C1 in HSV space</td>
</tr>
<tr>
<td>C3: Hue: histogram (15 bins)</td>
</tr>
<tr>
<td>C4: saturation: histogram (2 bins)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Texture</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1: L3 filters: mean absolute response (15 filters)</td>
</tr>
<tr>
<td>T2: L5 filters: histogram of maximum responses (35 bins)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: Long Lines: number of line pixels, sqrt(area)</td>
</tr>
<tr>
<td>P2: Long Lines: percent of nearly parallel pairs of lines</td>
</tr>
<tr>
<td>P3: Line Intersections: histogram over 8 orientations, entropy</td>
</tr>
<tr>
<td>P4: Line Intersections: percent right of image center</td>
</tr>
<tr>
<td>P5: Line Intersections: percent above image center</td>
</tr>
<tr>
<td>P6: Line Intersections: percent far from image center at 8 orientations</td>
</tr>
<tr>
<td>P7: Line Intersections: percent very far from image center at 8 orientations</td>
</tr>
<tr>
<td>P8: Vanishing Points: mean line pixels with vertical VP membership, sqrt(area)</td>
</tr>
<tr>
<td>P9: Vanishing Points: mean line pixels with horizontal VP membership, sqrt(area)</td>
</tr>
<tr>
<td>P10: Vanishing Points: percent of total line pixels with vertical VP membership</td>
</tr>
<tr>
<td>P11: Vanishing Points: x-pos of horizontal VP - segment center (0 if none)</td>
</tr>
<tr>
<td>P12: Vanishing Points: y-pos of highest/lowest vertical VP - segment center</td>
</tr>
<tr>
<td>P13: Vanishing Points: segment bounds with horizontal VP</td>
</tr>
<tr>
<td>P14: Gradient: x, y center of mass of gradient magnitude - vertical segment center</td>
</tr>
</tbody>
</table>
2. Get Good Spatial Support

- Single segmentation won’t work

- Solution: multiple segmentations

\[ P(\text{label} \mid \text{data}) \approx \sum_{\text{segments}} P(\text{good segment} \mid \text{data})P(\text{label} \mid \text{good segment, data}) \]
Decision Trees + Adaboost

Surface Confidence Maps
Surface Estimates: Outdoor

Avg. Accuracy
Main Class: 88%
Subclass: 62%

Input Image | Ground Truth | Our Result
--- | --- | ---
Surface Estimates: Paintings

Input Image  Our Result

Surface Estimates: Indoor

Avg. Accuracy
Main Class: 93%
Subclass:  76%

Input Image  Ground Truth  Our Result
Failures: Reflections and Shadows

Input Image

Our Result

Average Accuracy

Main Class: 88%
Subclasses: 61%

<table>
<thead>
<tr>
<th>Main Class</th>
<th>Support</th>
<th>Vertical</th>
<th>Sky</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>0.84</td>
<td>0.15</td>
<td>0.00</td>
</tr>
<tr>
<td>Vertical</td>
<td>0.09</td>
<td>0.90</td>
<td>0.02</td>
</tr>
<tr>
<td>Sky</td>
<td>0.00</td>
<td>0.10</td>
<td>0.90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vertical Subclass</th>
<th>Left</th>
<th>Center</th>
<th>Right</th>
<th>Porous</th>
<th>Solid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>0.37</td>
<td>0.32</td>
<td>0.08</td>
<td>0.09</td>
<td>0.13</td>
</tr>
<tr>
<td>Center</td>
<td>0.05</td>
<td>0.56</td>
<td>0.12</td>
<td>0.16</td>
<td>0.12</td>
</tr>
<tr>
<td>Right</td>
<td>0.02</td>
<td>0.28</td>
<td>0.47</td>
<td>0.13</td>
<td>0.10</td>
</tr>
<tr>
<td>Porous</td>
<td>0.01</td>
<td>0.07</td>
<td>0.03</td>
<td>0.84</td>
<td>0.06</td>
</tr>
<tr>
<td>Solid</td>
<td>0.04</td>
<td>0.20</td>
<td>0.04</td>
<td>0.17</td>
<td>0.55</td>
</tr>
</tbody>
</table>
Importance of Many Cues

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>All But Position</th>
<th>All But Color</th>
<th>All But Texture</th>
<th>All But Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>88%</td>
<td>84%</td>
<td>87%</td>
<td>87%</td>
<td>88%</td>
</tr>
<tr>
<td>Subclass</td>
<td>61%</td>
<td>60%</td>
<td>60%</td>
<td>58%</td>
<td>57%</td>
</tr>
</tbody>
</table>

Spatial Support Matters

<table>
<thead>
<tr>
<th>Method</th>
<th>Main</th>
<th>Sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixels</td>
<td>82.1</td>
<td>44.3</td>
</tr>
<tr>
<td>Superpixels</td>
<td>86.2</td>
<td>53.5</td>
</tr>
<tr>
<td>Single Segmentation</td>
<td>86.2</td>
<td>56.6</td>
</tr>
<tr>
<td>Multiple Segmentations</td>
<td>88.1</td>
<td>61.5</td>
</tr>
<tr>
<td>Ground Truth Segmentation</td>
<td>95.1</td>
<td>71.5</td>
</tr>
</tbody>
</table>