Object recognition

Methods for classification and image representation

Credits

• Slides by Pete Barnum
• Slides by Fei-Fei Li
• Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV’04
• Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
• Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell, Virtual Visual Hull: Example-Based 3D Shape Inference from Silhouettes
• S. Lazebnik, C. Schmid, and J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.
• Yoav Freund Robert E. Schapire, A Short Introduction to Boosting

Object recognition

• What is it?
  – Instance
  – Category
  – Something with a tail
• Where is it?
  – Localization
  – Segmentation
• How many are there?

(C) By Paul Godden
(C) By Peter Hellberg

(C) By Yannic Meyer
(C) By Peter Hellberg
Object recognition

- What is it?
  - Instance
  - Category
  - Something with a tail
- Where is it?
  - Localization
  - Segmentation
- How many are there?

Face detection

- We slide a window over the image
- Extract features for each window
- Classify each window into face/non-face

What is a face?

- Eyes are dark (eyebrows+shadows)
- Cheeks and forehead are bright.
- Nose is bright

Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV 04
Basic feature extraction

- Information type:
  - intensity
- Sum over:
  - gray and white rectangles
- Output: gray-white
- Separate output value for
  - Each type
  - Each scale
  - Each position in the window
- \( \text{FEX}(\text{im})=x=[x_1,x_2,\ldots,x_n] \)

Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV 04

Face detection

- We slide a window over the image
- Extract features for each window
- Classify each window into face/non-face

Classification

- Examples are points in \( \mathbb{R}^n \)
- Positives are separated from negatives by the hyperplane \( w \)
- \( y=\text{sign}(w^Tx-b) \)
Classification

- $x \in \mathbb{R}^n$ - data points
- $P(x)$ - distribution of the data
- $y(x)$ - true value of $y$ for each $x$
- $F$ - decision function:
  $y = F(x, \theta)$
- $\theta$ - parameters of $F$,
  e.g. $\theta = (w, b)$
- We want $F$ that makes few mistakes

Loss function

- Our decision may have severe implications
- $L(y(x), F(x, \theta))$ - loss function
  How much we pay for predicting $F(x, \theta)$ when the true value is $y(x)$
- Classification error:
  $L(y(x), F(x, \theta)) = \begin{cases} 0, & y(x) = \text{sign}(w^T x - b) \\ 1, & \text{otherwise} \end{cases}$
- Hinge loss
  $L(y(x), F(x, \theta)) = \max(0, 1 - y(x) F(x, \theta))$

Learning

- Total loss shows how good a function $(F, \theta)$ is:
  $L(f) = \int_x L(y, F(x)) P(x) dx$
- Learning is to find a function to minimize the loss:
  $(F, \theta) = \arg \min_{F, \theta} \int_x L(y, F(x, \theta)) P(x) dx$
- How can we see all possible $x$?
Datasets

- Dataset is a finite sample \( \{x_i\} \) from \( P(x) \)
- Dataset has labels \( \{(x_i, y_i)\} \)
- Datasets today are big to ensure the sampling is fair

<table>
<thead>
<tr>
<th></th>
<th>#images</th>
<th>#classes</th>
<th>#instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech 256</td>
<td>30608</td>
<td>256</td>
<td>30608</td>
</tr>
<tr>
<td>Pascal VOC</td>
<td>4340</td>
<td>20</td>
<td>10363</td>
</tr>
<tr>
<td>LabelMe</td>
<td>176975</td>
<td>??</td>
<td>414687</td>
</tr>
</tbody>
</table>

Overfitting

- A simple dataset.
- Two models

- Linear
- Non-linear

- Let’s get more data.
- Simple model has better generalization.
Overfitting

- As complexity increases, the model overfits the data
- Training loss decreases
- Real loss increases
- We need to penalize model complexity = to regularize

Overfitting

- Split the dataset
  - Training set
  - Validation set
  - Test set
- Use training set to optimize model parameters
- Use validation test to choose the best model
- Use test set only to measure the expected loss

Classification methods

- K Nearest Neighbors
- Decision Trees
- Linear SVMs
- Kernel SVMs
- Boosted classifiers
K Nearest Neighbors

- Memorize all training data
- Find K closest points to the query
- The neighbors vote for the label:
  \[ \text{Vote}(+) = 2 \]
  \[ \text{Vote}(-) = 1 \]

---

K-Nearest Neighbors

Nearest Neighbors (silhouettes)

Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell, Virtual Visual Hulls: Example-Based 3D Shape Estimation from Silhouettes

---

K-Nearest Neighbors

Silhouettes from other views 3D Visual hull

Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell, Virtual Visual Hulls: Example-Based 3D Shape Estimation from Silhouettes
**Decision Tree**

- Partition data into pure chunks
- Find a good rule
- Split the training data
  - Build left tree
  - Build right tree
- Count the examples in the leaves to get the votes: $V(+)$, $V(-)$
- Stop when
  - Purity is high
  - Data size is small
  - At fixed level

**Decision Trees**

- Stump:
  - 1 root
  - 2 leaves
- If $x_i > a$
  - then positive
  - else negative
- Very simple
- "Weak classifier"

---

Paul Viola, Michael Jones, Robust Real-time Object Detection, IJCV 04
Support vector machines

- Simple decision
- Good classification
- Good generalization

\[
\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_j \xi_j \\
y_j w^T x_j \geq 1 - \xi_j \\
\xi_j \geq 0
\]

Support vectors:

\[
w = \sum_{i \in S.U.} \alpha_i x_i
\]

How do I solve the problem?

- It’s a convex optimization problem
  - Can solve in Matlab (don’t)
- Download from the web
  - SMO: Sequential Minimal Optimization
  - SVM-Light http://svmlight.joachims.org/
  - LibSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - LibLinear http://www.csie.ntu.edu.tw/~cjlin/liblinear/
  - SVM-Perf http://svmlight.joachims.org/
  - Pegasos http://ttic.uchicago.edu/~shai/
Linear SVM for pedestrian detection

- centered
- uncentered
- cubic-corrected
- diagonal
- Sobel
• Histogram of gradient orientations
  - Orientation

8 orientations

$X = \in \mathbb{R}^{840}$

15x7 cells

$0.16 = w^T x - b$

$\text{sign}(0.16) = 1$

$\Rightarrow$ pedestrian
Kernel SVM

Decision function is a linear combination of support vectors:

\[ w = \sum_{i=1}^{n} \alpha_i x_i \]

Prediction is a dot product:

\[ (w; x) = \sum \alpha_i (x_i, x) \]

Kernel is a function that computes the dot product of data points in some unknown space:

\[ (\Psi(x_i), \Psi(x)) = K(x_i, x) \]

We can compute the decision without knowing the space:

\[ (w; \Psi(x)) = \sum \alpha_i K(x_i, x) \]

Useful kernels

\[ K(x_i, x) = \]

- Linear:

\[ (x_i, x) \]

- RBF

\[ exp(-||x_i - x||^2 / 2\sigma^2) \]

- Histogram intersection

\[ \sum_j \min(x_i^{(j)}, x^{(j)}) \]

- Pyramid match

Histogram intersection

Assign to texture cluster

\[ HIK(x_i, x) = \sum \min(x_i^{(j)}, x^{(j)}) \]

\[ x, L., S. L., J. P., Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. \]
(Spatial) Pyramid Match

\[ SPM(x_i, x_j) = \sum_{k=1}^{L} \frac{1}{L-k+1} HIK_k(x_i, x_j) \]


Boosting

\[ L(x, F(x)) = \begin{cases} 0, & y(x) = F(x) \\ 1, & \text{otherwise} \end{cases} \]

\[ Err(F, P(x)) = \int_{P(x)} P(x) L(x, F(x)) = \sum_{x_i} p_i L(x_i, F(x_i)) \]

- Weak classifier
  - Classifier that is slightly better than random guessing
  \[ Err(f, P(x)) \neq \frac{1}{2} \]
  - Weak learner builds weak classifiers
  \[ \forall P(x) \{y_i, P(x_i)\} \rightarrow WL \rightarrow f \]

Boosting

- Start with uniform distribution
- Iterate:
  1. Get a weak classifier \( f_k \)
  2. Compute it's 0-1 error
  3. Take
  4. Update distribution
- Output the final "strong" classifier

\[ p_k^{d+1} = \frac{1}{N} \]

\[ e_k = \sum_{x_i} p_i L(x_i, f_k(x_i)) \]

\[ a_k = \frac{1}{2} \ln \frac{1 - e_k}{e_k} \]

\[ p_{k+1} = \frac{p_k \exp(-a_k y_i f_k(x_i))}{Z_{k+1}} \]

\[ F(x) = \text{sign} \left( \sum_{k} a_k f_k(x) \right) \]

*Yoav Freund Robert E. Schapire. A Short Introduction to Boosting*
Face detection

- We slide a window over the image
- Extract features for each window
- Classify each window into face/non-face

Face detection

- Use haar-like features
- Use decision stumps as weak classifiers
- Use boosting to build a strong classifier
- Use sliding window to detect the face