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 Hulls: Example-Based 3D Shape Inference from Silhouettes
- 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?
Object recognition

• What is it?
  – Instance
  – Category
  – Something with a tail

• Where is it?
  – Localization
  – Segmentation

• How many are there?

(CC) By Dunechaser
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 $\mathbf{w}$
- $y = \text{sign}(\mathbf{w}^T \mathbf{x} - b)$
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>Dataset</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
Overfitting

- 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

![Diagram showing loss vs model complexity with stopping point indicated]
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 Inference 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 Inference from Silhouettes
Decision tree

- $V(+) = 8$
- $V(-) = 8$
- $V(+) = 2$
- $V(-) = 8$
- $V(+) = 0$
- $V(-) = 4$
- $V(+) = 8$
- $V(-) = 2$
Decision Tree Training

- 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: $V(\pm)$, $V(\mp)$
- Stop when
  - Purity is high
  - Data size is small
  - At fixed level

$V(-) = 80\%$
$V(\pm) = 84\%$
$V(-) = 57\%$
$V(-) = 100\%$
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

\[
\begin{align*}
\min_{w, \xi} & \quad \frac{||w||^2}{2} + C \sum_j \xi_j \\
\text{s.t.} & \quad y_j w^T x_j \geq 1 - \xi_j \\
& \quad \xi_j \geq 0
\end{align*}
\]
Support vector machines

\[
\begin{align*}
\min_{w, \xi} & \quad \frac{||w||^2}{2} + C \sum_j \xi_j \\
\text{subject to} & \quad y_j w^T x_j \geq 1 - \xi_j \\
& \quad \xi_j \geq 0
\end{align*}
\]

Support vectors:

\[w = \sum_{x_i - \text{s.v.}} \alpha_i x_i\]
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
Input image → Normalize gamma & colour → Compute gradients → Weighted vote into spatial & orientation cells → Contrast normalize over overlapping spatial blocks → Collect HOG’s over detection window → Linear SVM → Person/non-person classification
Input image → Normalize gamma & colour → Compute gradients → Weighted vote into spatial & orientation cells → Contrast normalize over overlapping spatial blocks → Collect HOG’s over detection window → Linear SVM → Person/non-person classification

-1 0 1  centered

-1 1  uncentered

1 -8 0 8 -1  cubic-corrected

0 1
-1 0  diagonal

-1 0 1
-2 0 2
-1 0 1  Sobel

Slides by Pete Barnum
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
- Histogram of gradient orientations

Slides by Pete Barnum
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
Input image → Normalize gamma & colour → Compute gradients → Weighted vote into spatial & orientation cells → Contrast normalize over overlapping spatial blocks → Collect HOG’s over detection window → Linear SVM → Person/ non-person classification

8 orientations

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

15x7 cells

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
0.16 = \mathbf{w}^T \mathbf{x} - b

\text{sign}(0.16) = 1

\Rightarrow \text{pedestrian}
Kernel SVM

Decision function is a linear combination of support vectors:

\[ w = \sum_{x_i - s.v.} \alpha_i x_i \]

Prediction is a dot product:

\[ (w, x) = \sum_{x_i - s.v.} \alpha_i (x_i, x) \]

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:

\[ (w, \Psi(x)) = \sum_{x_i - s.v.} \alpha_i K(x_i, x) \]
Useful kernels

\[ K(x_i, x) = \]

- Linear!

- RBF

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

- Histogram intersection

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

- Pyramid match
Histogram intersection

Assign to texture cluster \[ HIK(x_i, x) = \sum_{j} \min(x_i^{(j)}, x^{(j)}) \]

Count

(Spatial) Pyramid Match

\[
SPM(x_i, x) = \frac{1}{2L} HIK_0(x_i, x) + \ldots + \frac{1}{L - l + 1} HIK_l(x_i, x) + \ldots + HIK_L(x_i, x)
\]

Boosting

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

\[ Err(F, P(x)) = \int_{P(x)} P(x)L(x, F(x)) \approx \sum_{x_j} p_jL(x_j, F(x_j)) \]

- 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

\[
(x_i, y_i), \quad p_i^{k=1} = \frac{1}{N}
\]

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

\[
\alpha_k = \frac{1}{2} \ln \left( \frac{1 - \epsilon_k}{\epsilon_k} \right)
\]

\[
p_i^{k+1} = \frac{p_i^k \exp(-\alpha_k y_i f_k(x_i))}{Z_{k+1}}
\]

\[
F(x) = \text{sign} \left( \sum_k \alpha_k f_k(x) \right)
\]

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