Computer Vision - Lecture 11
Sliding-Window based Object Detection II
01.12.2015

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de

Course Outline
- Image Processing Basics
- Segmentation
  - Segmentation and Grouping
  - Segmentation as Energy Minimization
- Recognition & Categorization
  - Global Representations
  - Sliding-Window Object Detection
  - Image Classification
- Local Features & Matching
- 3D Reconstruction
- Motion and Tracking

Topics of This Lecture
- Recap: Classification with SVMs
  - Support Vector Machines
  - HOG Detector
- Classification with Boosting
  - AdaBoost
  - Viola-Jones Face Detection
- Discussion

Recap: Sliding-Window Object Detection
- If the object may be in a cluttered scene, slide a window around looking for it.
  - Essentially, this is a brute-force approach with many local decisions.

Recap: Support Vector Machine (SVM)
- Basic idea
  - The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
  - Up to now: consider linear classifiers
  - $w^T x + b = 0$
- Formulation as a convex optimization problem
  - Find the hyperplane satisfying
    $\arg\min_{w,b} \frac{1}{2}||w||^2$
  - under the constraints
    $t_n (w^T x_n + b) \geq 1 \quad \forall n$
  - based on training data points $x_n$ and target values $t_n \in \{-1, 1\}$

Recap: Non-Linear SVMs
- General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:
  - $\Phi: x \rightarrow \phi(x)$

Slide from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html
Recap: Gradient-based Representations

- Consider edges, contours, and (oriented) intensity gradients
- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination

Recap: HOG Descriptor Processing Chain

- Collect HOGs over detection window
- Contrast normalize over overlapping spatial cells
- Weighted vote in spatial & orientation cells
- Compute gradients
- Gamma compression

Recap: Pedestrian Detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template

Recap: Non-Maximum Suppression

- After multi-scale detection
- Map each detection to 3D space
- Apply robust mode detection, e.g., mean shift
- Non-maximum suppression

Applications: Mobile Robot Navigation

- Select tour: Moving robot, Dialog view, Web view

Classifier Construction: Many Choices...

- Nearest Neighbor
- Neural networks
- Boosting
- Support Vector Machines
- Randomized Forests
AdaBoost: Intuition

Consider a 2D feature space with positive and negative examples.

Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

AdaBoost: Detailed Training Algorithm

1. Initialization: Set $w_1^{(1)} = \frac{1}{N}$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$ iterations
   a) Train a new weak classifier $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function
      $$J_m = \sum_{n=1}^{N} w_1^{(m)} I(h_m(x_n) \neq t_n)$$
      where $I$ is
      $$I(\text{true}) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{if } A \text{ is false} \end{cases}$$
   b) Estimate the weighted error of this classifier on $X$:
      $$\epsilon_m = \frac{1}{N} \sum_{n=1}^{N} w_1^{(m)} I(h_m(x_n) \neq t_n)$$
   c) Calculate a weighting coefficient for $h_m(x)$:
      $$\alpha_m = \ln \frac{1 - \epsilon_m}{\epsilon_m}$$
   d) Update the weighting coefficients:
      $$w_1^{(m+1)} = w_1^{(m)} \exp \{\alpha_m I(h_m(x_n) \neq t_n)\}$$

AdaBoost - Formalization

- 2-class classification problem
  - Given: training set $X = \{x_1, \ldots, x_N\}$ with target values $T = \{t_1, \ldots, t_N\}, t_n \in \{-1, 1\}$.
  - Associated weights $W = \{w_1, \ldots, w_N\}$ for each training point.

- Basic steps
  - In each iteration, AdaBoost trains a new weak classifier $h_m(x)$ based on the current weighting coefficients $W^{(m)}$.
  - We then adapt the weighting coefficients for each point:
    - Increase $w_n$ if $x_n$ was misclassified by $h_m(x)$.
    - Decrease $w_n$ if $x_n$ was classified correctly by $h_m(x)$.
  - Make predictions using the final combined model:
    $$H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)$$
AdaBoost: Recognition

- Evaluate all selected weak classifiers on test data.
  \[ h_1(x), \ldots, h_m(x) \]
- Final classifier is weighted combination of selected weak classifiers:
  \[ H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right) \]
- Very simple procedure!
  - Less than 10 lines in Matlab!
  - But works extremely well in practice...

Example: Face Detection

- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a "patch"/window
- Now we'll take AdaBoost and see how the Viola-Jones face detector works

Feature extraction

"Rectangular" filters

Feature output is difference between adjacent regions

Easily computable with integral image: any sum can be computed in constant time

Avoid scaling images \( \rightarrow \) scale features directly for same cost

Example

Integral Image

Value at \((x,y)\) is sum of pixels above and to the left of \((x,y)\)

Large Library of Filters

Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

Weak classifier: filter output > \(\theta\)?

AdaBoost for Feature+Classifier Selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error.

Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

For next round, reweight the examples according to errors, choose another filter/threshold combo.
AdaBoost for Efficient Feature Selection

- Image features = weak classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this features is a simple function of error rate
  - Reweight examples

Cascading Classifiers for Detection

- Even if the filters are fast to compute, each new image has a lot of possible windows to search.
- For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,
  - Filter for promising regions with an initial inexpensive classifier
  - Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain

Cascading Classifiers

- Chain classifiers that are progressively more complex and have lower false positive rates:

Viola-Jones Face Detector: Summary

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]

Viola-Jones Face Detector: Results

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]

Practical Issue: Bootstrapping

- Problem: 1 face in 116,440 examined windows
  - Can easily find negative examples, but which ones are useful?
  - Apply iterative training approach
  - False positives on negative validation images are included in training set as “hard negatives”
Viola-Jones Face Detector: Results

You Can Try It At Home...

- The Viola & Jones detector was a huge success
  - First real-time face detector available
  - Many derivative works and improvements

- C++ implementation available in OpenCV [Lienhart, 2002]
  - http://sourceforge.net/projects/opencvlibrary/
- Matlab wrappers for OpenCV code available, e.g. here

Summary: Sliding-Windows

- Pros
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes
  - Good detectors available (Viola & Jones, HOG, etc.)

- Cons/Limitations
  - High computational complexity
    - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
    - This puts tight constraints on the classifiers we can use.
    - If training binary detectors independently, this means cost increases linearly with number of classes.
  - With so many windows, false positive rate better be low

Feature Computation Trade-Off

- Linear SVM Detectors
  - Same computations performed for each image window
  - It pays off to precompute the features once
  - Complex features can be used

- AdaBoost Cascaded Detectors
  - Potentially different computations for each window location
  - May be more efficient to evaluate the features on-the-fly for each image window
  - If cascading shall be used, simple features are preferable

Example Application

Everingham, M., Sivic, J. and Zisserman, A.
"Hello! My name is... Buffy" - Automatic naming of characters in TV video, BMVC 2006.
http://www.robots.ox.ac.uk/~vgg/research/nface/index.html
What Slows Down HOG (CUDA Implem.)

- Results from fastHOG (10fps) [Prisacariu & Reid 2009]

Limitations: Low Training Resolutions
- Many (older) S/W detectors operate on tiny images
  - Viola&Jones: 24x24 pixels
  - Torralba et al.: 32x32 pixels
  - Dalal&Triggs: 64x96 pixels (notable exception)
- Main reasons
  - Training efficiency (exhaustive feature selection in AdaBoost)
  - Evaluation speed
  - Want to recognize objects at small scales
- But...
  - Limited information content available at those resolutions
  - Not enough support to compensate for occlusions!

Limitations: Changing Aspect Ratios
- Sliding window requires fixed window size
  - Basis for learning efficient cascade classifier
- How to deal with changing aspect ratios?
  - Fixed window size
    - Wastes training dimensions
  - Adapted window size
    - Difficult to share features
    - “Squashed” views [Dalal&Triggs]
      - Need to squash test image, too

Limitations (continued)
- Non-rigid, deformable objects not captured well with representations assuming a fixed 2D structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions

Limitations (continued)
- If considering windows in isolation, context is lost

Figure credit: Derek Hoiem
Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

References and Further Reading

- Read the Viola-Jones paper
  - (first version appeared at CVPR 2001)
- Viola-Jones Face Detector
  - C++ implementation available in OpenCV [Lienhart, 2002]
    - http://sourceforge.net/projects/opencvlibrary/
  - Matlab wrappers for OpenCV code available, e.g. here
- HOG Detector
  - Code available: http://pascal.inrialpes.fr/software/olt/