Recap: Tracking as Online Classification
• Tracking as binary classification problem

Recap: AdaBoost – "Adaptive Boosting"
• Main idea
  [Freund & Schapire, 1996]
  – Iteratively select an ensemble of classifiers
  – Reweight misclassified training examples after each iteration
    to focus training on difficult cases.
• Components
  – \( h_m(x) \): “weak” or base classifier
  – Condition: <50% training error over any distribution
  – \( H(x) \): “strong” or final classifier
• AdaBoost:
  – Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:
    \[
    H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)
    \]
From Offline to Online Boosting

- Main issue
  - Computing the weight distribution for the samples.
  - We do not know a priori the difficulty of a sample!
  - (Could already have seen the same sample before...)

- Idea of Online Boosting
  - Estimate the importance of a sample by propagating it through a set of weak classifiers.
  - This can be thought of as modeling the information gain w.r.t. the first $n$ classifiers and code by the importance weight $\lambda$ for the $n+1$ classifier.
  - Proven [Oza]: Given the same training set, Online Boosting converges to the same weak classifiers as Offline Boosting in the limit of $N \to \infty$ iterations.


Recap: Tracking by Online Classification

- From time $t$ to $t+1$
  - Update classifier (tracker)
  - Analyze map and set new object position
  - Create confidence map
  - Search region

Evaluate classifier on sub patches

Recap: Tracking by Detection

- Computing the weight distribution for the samples.
- We do not know a priori the difficulty of a sample!
- (Could already have seen the same sample before...)

- Idea of Online Boosting
- Estimate the importance of a sample by propagating it through a set of weak classifiers.
- This can be thought of as modeling the information gain w.r.t. the first $n$ classifiers and code by the importance weight $\lambda$ for the $n+1$ classifier.
- Proven [Oza]: Given the same training set, Online Boosting converges to the same weak classifiers as Offline Boosting in the limit of $N \to \infty$ iterations.


Recap: From Offline to Online Boosting

<table>
<thead>
<tr>
<th>off-line</th>
<th>on-line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given:</td>
<td>Given:</td>
</tr>
<tr>
<td>- set of labeled training samples $X = [(x_1, y_1), ..., (x_m, y_m)]$</td>
<td>- ONE labeled training sample $(x, y) \in \mathbb{Y}$</td>
</tr>
<tr>
<td>- weight distribution over them $D_0 = 1/L$</td>
<td>- strong classifier to update</td>
</tr>
<tr>
<td>for $n = 1$ to $N$</td>
<td>for $n = 1$ to $N$</td>
</tr>
<tr>
<td>- train a weak classifier using $h_n = \text{Classifier}_{X, D_n-1}$</td>
<td>- update the weak classifier using samples and importance</td>
</tr>
<tr>
<td>$h_n^\text{weak}(x) = \mathbb{I}(X, D_n-1)$</td>
<td>$h_{n+1} = \text{Classifier}_{X, D_n}^\text{weak}(x, y)\lambda$</td>
</tr>
<tr>
<td>- calculate error $e_n = f(x_n)$</td>
<td>- update error estimation $e_{n+1}$</td>
</tr>
<tr>
<td>- calculate weight $\alpha_n = \alpha \cdot e_n^{-\lambda}$</td>
<td>- update weight $\omega_n = f(x_n)$</td>
</tr>
<tr>
<td>- update weight dist. (classifier) $D_n = D_{n-1} - \alpha_n$</td>
<td>- update importance weight $\lambda$</td>
</tr>
<tr>
<td>next $h_{n+1} = \text{sign}(\sum_{k=1}^n \alpha_k \cdot h_n^\text{weak}(x))$</td>
<td>next $h_{n+1} = \text{sign}(\sum_{k=1}^n \alpha_k \cdot h_n^\text{weak}(x))$</td>
</tr>
</tbody>
</table>

Recap: Direct Feature Selection

- Introducing “Selector”
  - Selects one feature from its local feature pool $h_n^\text{weak} = \{h_1^\text{weak}, ..., h_m^\text{weak}\}$
  - $X = \{f_1, ..., f_m\}$
  - $h_n^\text{weak}(x) = h_n^\text{weak}(x)$
  - $m = \arg \min_i e_i$

On-line boosting is performed on the Selectors and not on the weak classifiers directly.


Recap: Drifting Due to Self-Learning Policy

- Not only does it drift, it also remains confident about it!

Tracked Patches

Confidence

100% 50% 0%
Today: Tracking by Detection

Can we use generic object detection to track people?

Topics of This Lecture

- Tracking by Detection
  - Motivation
  - Recap: Object detection
- SVM based Detectors
  - Recap: HOG
  - DPM
- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
- CNN-based Detectors
  - Recap: CNNs
  - R-CNN, Faster R-CNN
  - YOLO, SSD

Detection-Based Tracking

- Main ideas
  - Apply a generic object detector to find objects of a certain class
  - Based on the detections, extract object appearance models
    - Even possible to derive figure-ground segmentations from detection results
  - Link detections into trajectories

Spacetime Trajectory Analysis

- Pedestrian detection
- Car detections
- Own vehicle

Tracking-by-Detection in 3D

- Object detections
- Spacetime trajectories
- 3D Camera path estimation
- Main Issue: Data Association (We’ll come to that later…)

Elements of Tracking

- Detection
  - Where are candidate objects?
- Data association
  - Which detection corresponds to which object?
- Prediction
  - Where will the tracked object be in the next time step?
Recap: Sliding-Window Object Detection

• Basic component: a binary classifier

![Car/non-car Classifier](image)

Recap: Sliding-Window Object Detection

• If object may be in a cluttered scene, slide a window around looking for it.

![Car/non-car Classifier](image)

• Essentially, this is a brute-force approach with many local decisions.

What is a Sliding Window Approach?

• Search over space and scale

![JUDYBATS](image)

• Detection as subwindow classification problem

“In the absence of a more intelligent strategy, any global image classification approach can be converted into a localization approach by using a sliding-window search.”

Recap: Non-Maximum Suppression

After multi-scale dense scan

![Goal](image)

Map each detection to 3D box world space

Apply robust mode detection, e.g. mean shift

Non-maximum suppression

Recap: Sliding-Window Object Detection

• Fleshing out this pipeline a bit more, we need to:
  1. Obtain training data
  2. Define features
  3. Define classifier

![Feature extraction](image)

Object Detector Design

• In practice, the classifier often determines the design.
  – Types of features
  – Speedup strategies

• Today, we’ll look at 3 state-of-the-art detector designs
  – Based on SVMs
    – Based on Boosting
  – Based on CNNs

![Training examples](image)

![Car/non-car Classifier](image)
Topics of This Lecture

- Tracking by Detection
  - Motivation
  - Recap: Object detection
- SVM based Detectors
  - Recap: HOG
  - DPM
- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
- CNN-based Detectors
  - Recap: CNNs
  - R-CNN

Recap: Histograms of Oriented Gradients (HOG)

- Holistic object representation
  - Localized gradient orientations

  - 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
  \[ y(x) = w^T x + b \]

Pedestrian detection with HoGs & SVMs

N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

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: Part-Based Models

- Pictorial Structures model
  - [Fischler & Elschlager 1973]

- Model has two components
  - Parts (2D image fragments)
  - Structure (configuration of parts)

- Use in Deformable Part-based Model (DPM)
  - Parts $\approx 5$–$7$ semantically meaningful parts
  - Probabilistic model enabling efficient inference

Starting Point: HOG Sliding-Window Detector

- Array of weights for features in window of HOG pyramid
- Score is dot product of filter and vector

Deformable Part-based Models

- Mixture of deformable part models (Pictorial Structures)
- Each component has global template + deformable parts
- Fully trained from bounding boxes alone

2-Component Bicycle Model

- Root filters coarse resolution
- Part filters finer resolution
- Deformation models

Object Hypothesis

- Multiscale model captures features at two resolutions

Score of a Hypothesis

$$
score(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i, dy_i)
$$

- “data term”
- “spatial prior”

$$
score(z) = \beta \cdot \Psi(H, z)
$$

- concatenation filters and deformation parameters
- concatenation of HOG features and part displacement features
Recognition Model

\[ f_w(x) = w \cdot \Phi(x) \]

\[ f_w(x) = \max_z w \cdot \Phi(x, z) \]

• Difference to standard HOG model
  – Hidden variable \( z \): vector of part offsets
  – \( \Phi(x, z) \): vector of HOG features (from root filter & appropriate part sub-windows) and part offsets
  \( \Rightarrow \) Need to optimize over all possible part positions

Results: Persons

• Results (after non-maximum suppression)
  – \( \sim 1s \) to search all scales

Results: Bicycles

Extensions and Detailed Improvements

• More efficient features
  – Very simplified version of HOG
• Latent part (re-)learning
  – Perform several rounds of training, adapting the annotation boxes
• Multi-aspect detection
  – Mixture model of different aspects to capture different viewpoints of objects
• Bounding box prediction
  – Infer final detection bounding box from detected part locations
• Multi-resolution models
• Cascaded evaluation

You Can Try It At Home...

• Deformable part-based models have been very successful in several evaluations.
  \( \Rightarrow \) Approach was state-of-the-art until few years ago
• Source code and models trained on PASCAL 2006, 2007, and 2008 data are available here:
  http://www.cs.uchicago.edu/~pff/latent
**Topics of This Lecture**

- Tracking by Detection
  - Motivation
  - Recap: Object detection
- SVM based Detectors
  - Recap: HOG
  - DPM
- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
- CNN-based Detectors
  - Recap: CNNs
  - R-CNN

**Recap: Viola-Jones Face Detector**

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

**Recap: Haar Wavelets**

- "Rectangular" filters
  - Feature output is difference between adjacent regions
  - Value at (x,y) is sum of pixels above and to the left of (x,y)
  - Efficiently computable with integral image: any sum can be computed in constant time
  - Avoid scaling images → Scale features directly for same cost

**Recap: Cascading Classifiers for Detection**

- Even if the filters are fast to compute, each new image has a lot of possible windows to search...
- Idea: Classifier cascade
  - Observation: most image windows are negative and look very different from the searched object class.
  - 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

**Viola-Jones Face Detector: Results**

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

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

Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection
• SVM based Detectors
  – Recap: HOG
  – DPM
• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei
• CNN-based Detectors
  – Recap: CNNs
  – R-CNN

Integral Channel Features

• Generalization of Haar Wavelet idea from Viola-Jones
  – Instead of only considering intensities, also take into account other feature channels (gradient orientations, color, texture).
  – Still efficiently represented as integral images.

P. Dollar, Z. Tu, P. Perona, S. Belongie. Integral Channel Features, BMVC'09.

Integral Channel Features

• Generalize also block computation
  – 1st order features:
    • Sum of pixels in rectangular region.
  – 2nd-order features:
    • Haar-like difference of sum-over-blocks
  – Generalized Haar:
    • More complex combinations of weighted rectangles
  – Histograms
    • Computed by evaluating local sums on quantized images.

Results: Integral Channel Features

[Viola & Jones 2004]
fastHOG
~10 Hz on GPU
[Prisacaru 2009]
DPM
[Felzenszwalb 2008]
ChnFtrs/FPDW
~5 Hz on CPU
[Dollar 2009+2010]

Performance Comparison of Detectors

INRIA dataset

[Viola & Jones 2004]
fastHOG
~10 Hz on GPU
[Prisacaru 2009]
DPM
[Felzenszwalb 2008]
ChnFtrs/FPDW
~5 Hz on CPU
[Dollar 2009+2010]
Performance Comparison of Detectors

Performance Comparison of Detectors

INRIA dataset

VeryFast 50 Hz

Issues for Efficient Detection

• One template cannot detect at multiple scales...

Issues for Efficient Detection

• Typically, features are computed many times

Issues for Efficient Detection

• Typically, features are computed many times

Issues for Efficient Detection

VeryFast Detector

• Idea 1: Invert the relation

VeryFast Detector

Practical Considerations

• Training and running 1 model/scale is too expensive

Practical Considerations

**VeryFast Detector**

- **Idea 2**: Reduce training time by feature interpolation

  5 models, 1 image scale  
  50 models, 1 image scale

- Shown to be possible for Integral Channel features
  

  ⇒ Result: 3x reduction in feature computation

**VeryFast: Classifier Construction**

- Ensemble of short trees, learned by AdaBoost

\[
\text{score} = w_1 h_1 + w_2 h_2 + \ldots + w_N h_N
\]

**Learned Models**

- Integral Channel features
- Models
• Detection without resizing improves quality of results

Comparison to State-of-the-Art

• Extension: Roerei detector
  - Detailed evaluation of design space
  - Non-regular pooling regions found to work best.

Roerei Results


Topics of This Lecture

• Tracking by Detection
  - Motivation
  - Recap: Object detection
• SVM based Detectors
  - Recap: HOG
  - DPM
• AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
• CNN-based Detectors
  - Recap: CNNs
  - R-CNN
Recap: Convolutional Neural Networks

- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end


Recap: Intuition of CNNs

- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels
- Learn multiple filters
  - E.g. 1000 x 1000 image
  - 100 filters
  - 10 x 10 filter size
  - Only 10k parameters
- Result: Response map
  - Size: 1000 x 1000 x 100
  - Only memory, not parameters!

Recap: Convolution Layers

- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single [1 x 1 x depth] depth column in output volume

Recap: Activation Maps

Recap: Pooling Layers

- Effect:
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

Recap: R-CNN for Object Detection

- Box reg
- SVMs
- ConvNet
- Warped image regions
- Regions of Interest (RoI) from a proposal method
- Forward each region through ConvNet
Recap: Faster R-CNN

- One network, four losses
  - Remove dependence on external region proposal algorithm.
  - Instead, infer region proposals from same CNN.
  - Feature sharing
  - Joint training
  => Object detection in a single pass becomes possible.

Most Recent Version: Mask R-CNN

Mask R-CNN Results

- Detection + Instance segmentation
- Detection + Pose estimation

YOLO / SSD

YOLO-v2 Results

You Can Try All of This At Home…

- Detector code is publicly available
  - HOG:
    - Dalal’s original implementation: [http://www.navneetdalal.com/software/](http://www.navneetdalal.com/software/)
    - Our CUDA-optimized groundhog code (>80 fps on GTX 580) [http://www.vision.rwth-aachen.de/software/groundhog](http://www.vision.rwth-aachen.de/software/groundhog)
  - DPM:
    - Felzenswalb’s original implementation: [http://www.cs.uchicago.edu/~pff/latent](http://www.cs.uchicago.edu/~pff/latent)
  - VeryFast:
    - Benenson’s original implementation: [https://bitbucket.org/rodrigob/doppia/](https://bitbucket.org/rodrigob/doppia/)
  - YOLO: