Q&A of the Deformable Part Model

Philipp Krähenbühl
Deformable Part Model

[P.Felzenszwalb, D.McAllester, and D.Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. CVPR 2008.]


Project 1

• 25% of your final grade
  – 35% technical approach and code
  – 35% experimental evaluation
  – 20% write-up quality
  – 10% project presentation

• Teams of 2 students allowed and encouraged
Starter Code

• [http://vision.stanford.edu/teaching/cs231b_spring1213/project.html](http://vision.stanford.edu/teaching/cs231b_spring1213/project.html)

• Low dimensional HOG feature
  – feature.cc (PAMI paper)

• Root filter detection
  – detect.m

• Root filter training
  – train.m
Data

• VOC 2007 pedestrian detection
  – 2501 positive training images
  – 4952 test images
  – starter code performance 0.126 AP
  – reference implementation 0.362 AP

• 400 image subset
  – Use only for prototyping! Not for final evaluation!
  – Use at least 1000 images for evaluation
Starter Code - HOG

• 9 contrast insensitive orientations
  – unsigned gradient
• 18 contrast sensitive orientations
  – signed gradient
• 4 cells containing gradient energy
  – sum of all gradient values
• See PAMI paper for details
**Starter Code - HOG**

- 31 dimensional feature vector
  - Matlab: \( G = \text{feature}( \text{im}, k ) \)
  - for block size \( k \) (usually 8)
  - \( \text{im} \) is a \( W \times H \times 3 \) double matrix (color image)
  - \( F \) is a matrix of size \( \frac{W}{k} \times \frac{H}{k} \times 3 \times 31 \)
Starter Code - HOG

feature(8)
Starter Code - HOG

31-dim feature
Starter Code - HOG

- Multiple scales
  - featpyramid(im, k, interval)
Starter Code - Detection
Starter Code - Detection

\[ F \cdot G[x_0 : x_1, y_0 : y_1] = \sum_{x',y'} F[x',y']G[x_0 + x',y_0 + y'] \]
Starter Code - Detection

• Can be done efficiently using a convolution
  – Matlab: fconv(G, f, 1, length(f))
  • See detect.m
Starter Code - Detection
Starter Code - Detection

- Root filter response (F•G)
  - detection if greater than threshold
  - rank detections
- Non-maxima suppression
  - Only keep the highest responding box in a certain area
  - Avoid false positives
Detection - Evaluation

• Choosing a threshold
  – Lower threshold -> more false positives
  – Higher threshold -> lower detection rate
  – equivalent to ranking all detections
    • by root filter response
  – Precision recall curve (pascal_eval.m)
  – Average precision (at 11 recall levels of PR curve)
Starter Code - Training

• Why is it challenging?
Starter Code - Training

Negative examples

Positive examples

(warp and flip, poswarp in train.m)
Starter Code - Training

All positive examples + random negative examples (\textit{negrandom} in \textit{train.m})

All positive examples + hard negative examples (\textit{neghard} in \textit{train.m})

Much more negative image examples than positive examples!

“Cascade” in [Viola and Jones, CVPR 2001]
Starter Code - Training

• Collect warped positives
• Collect (random or hard) negatives
• Train svm
  – ./learn command in train.m
  – Finds root filter parameters $F$
Starter Code – Questions?
Your implementation

• Hard negatives
• Training DPM
  – Latent root position
    • Find best root filter position in training examples
  – Latent part position
  – Part filters
• Detection with DPM
Deformable Part Model

- **root** (whole body)
- **parts** (head, arms, legs, ...)

\[
\begin{align*}
F_0 \cdot \phi(H, p_0) + \\
\sum_{i=1}^{n} \left( F_i \cdot \phi(H, p_i) + a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot (\tilde{x}_i^2, \tilde{y}_i^2) \right)
\end{align*}
\]

- **root appearance**
- **part appearance**
- **part location**
DPM - Detection

• Root filter
  – In starter code
• Evaluate convolution with part filters
  – fconv and featpyramid
• Optimize part locations

\[
\max_{p_1 \cdots p_n} F_0 \cdot \phi(H, p_0) + \sum_{i=1}^{n} \left( F_i \cdot \phi(H, p_i) + a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot (\tilde{x}_i, \tilde{y}_i)^2 \right)
\]
DPM - Detection

• Root filter
  – In starter code

• Evaluate convolution with part filters
  – fconv and featpyramid

• Optimize part locations

\[ F_0 \cdot \phi(H, p_0) + \sum_{i=1}^{n} \max_{p_i} \left( F_i \cdot \phi(H, p_i) + a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot (\tilde{x}_i, \tilde{y}_i)^2 \right) \]
Fig. 4. The matching process at one scale. Responses from the root and part filters are computed at different resolutions in the feature pyramid. The transformed responses are combined to yield a final score for each root location. We show the responses and transformed responses for the “head” and “right shoulder” parts. Note how the “head” filter is more discriminative. The combined scores clearly show two good hypotheses for the object at this scale.
DPM - Detection

• Pyramid representation and HOG extraction;
• Scanning window in each pyramid;
• In each window: Latent SVM (finding part locations and obtaining the detection score).

• `detect.m`

• Non-maximum suppression
• Mean-average precision
  (`pascal_eval.m`)
Detection – Questions?
Deformable Part Model

\[ F_0 \cdot \phi(H, p_0) + \sum_{i=1}^{n} \left( F_i \cdot \phi(H, p_i) + a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot (\tilde{x}_i^2, \tilde{y}_i^2) \right) \]

- How to model root and part appearance \( \phi(H, p_i) \)?
  - **HOG feature**

- How to compute the parts location \((\tilde{x}_i, \tilde{y}_i)\)?
  - **Parts initialization and Latent SVM**

- How to compute the feature weights \(F_i, a_i, b_i\)?
  - **Latent SVM**
DPM – Training

pascal_train.m:

• Train initial root filter
• Train latent root filter on hard examples
• Initialize part filters
• For a few iterations
  – Retrain DPM
DPM – Training (in starter code)

train.m:

• Find positive examples
  – Directly from bounding boxes B

• Find negative examples
  – Randomly sample negatives

• Train LSVM
DPM – Training

pascal_train.m:

• Train initial root filter
• Train latent root filter on hard examples
• Initialize part filters
• For a few iterations
  – Retrain DPM
DPM - Training

train.m:

• Find positive examples
  – Find max scoring latent root position, that overlaps at least 50% with B

• For a few iterations
  – Find negative examples
    • Find/add hard negatives
  – Train LSVM
  – Remove easy negatives
DPM – Training

pascal_train.m:

• Train initial root filter
• Train latent root filter on hard examples
• Initialize part filters
• For a few iterations
  – Retrain DPM
Part initialization

• n parts per model (usually 6)
• all parts are symmetric
  – in symmetric pairs
  – symmetric filter in center
• greedily place at high energy regions
  – norm of positive root filter weights
  – block off regions used by other parts
• Initialize part filter to root filter response
DPM – Training

pascal_train.m:

• Train initial root filter
• Train latent root filter on hard examples
• Initialize part filters
• For a few iterations
  – Retrain LSVM
DPM – Latent SVM

\[ F_0 \cdot \phi(H, p_0) + \sum_{i=1}^{n} \left( F_i \cdot \phi(H, p_i) + a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot (\tilde{x}_i, \tilde{y}_i)^2 \right) \]

\[ f_\beta(x) = \max_z \beta \cdot \Phi(x, z) \quad \text{(Section 3.1)} \]

- $\beta$: feature weights;
- $z$: location of the patches;
- $x$: feature values corresponding to $z$. 
DPM – Latent SVM

\[ f_\beta(x) = \max_{z} \beta \cdot \Phi(x, z) \quad (\text{Section 3.1}) \]

• \( \beta \): feature weights;
• \( z \): location of the patches;
• \( x \): feature values corresponding to \( z \).

\[ L_D(\beta) = \frac{1}{2} |\beta|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)) \]
DPM – Latent SVM

\[ f_\beta(x) = \max_z \beta \cdot \Phi(x, z) \]

\[ L_D(\beta) = \frac{1}{2} |\beta|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)) \]

- Semi-convex
  - If we fix the latent variables of the positive example, the problem is convex
DPM – Latent SVM

\[ f_\beta(x) = \max_z \beta \cdot \Phi(x, z) \]

\[ L_D(\beta) = \frac{1}{2} |\beta|^2 + C \sum_{i=1}^n \max(0, 1 - y_i f_\beta(x_i)) \]

- Find highest scoring latent values for positive example
- Optimize \( \beta \)
DPM – Latent SVM

learn.cc
- ./learn C J hdr dat mod inf lob
  - input
    - C: SVM parameter (0.002)
    - J: learning rate (1)
    - hdr: header file (see writeheader.m)
    - dat: data file (next slide)
    - lob: lower bounds on parameters (see writelob.m)
  - output
    - mod: model file
      - stores learned model parameters (see train.m)
    - inf: info file
      - stores learned labels(-1,1) and svm values
DPM – Latent SVM

• Data file
  – Example*
    • labels [label id level x y]
      – label: -1, 1
      – id: example/image id
        » Only one positive example per id
        » Multiple negative examples per id
    • #blocks
    • #dims
      – Number of parameters to learn
  • Data [times #block]
    – block label [used to identify parts]
    – block data [position and HOG features]
HOG Feature Representation


- Part of starter code
- *feature.cc* (mex file for efficiency)
- Resolution of the parts is twice higher as that of the root.

\[
F_0 \cdot \phi(H, p_0) + \sum_{i=1}^{n} \left( F_i \cdot \phi(H, p_i) + a_i \cdot \left( \tilde{x}_i, \tilde{y}_i \right) + b_i \cdot \left( \tilde{x}_i^2, \tilde{y}_i^2 \right) \right)
\]
Location of Root & Parts

Root:

- Root filter update;
- Not a part of the scoring function.

Parts:

- Relative location w.r.t. the root;
- Parts initialization;
- Location updates in Latent SVM;

\[ F_0 \cdot \phi(H, p_0) + \sum_{i=1}^{n} \left( F_i \cdot \phi(H, p_i) + a_i \cdot \left( \begin{array}{c} \tilde{x}_i \\ \tilde{y}_i \end{array} \right) + b_i \cdot \left( \begin{array}{c} \tilde{2} \\ \tilde{2} \end{array} \right) \right) \]
Latent SVM

\[ f_\beta(x) = \max_z \beta \cdot \Phi(x, z) \quad \text{(Section 3.1)} \]

- \( \beta \): feature weights;
- \( z \): location of the patches;
- \( x \): feature values corresponding to \( z \).

Iterative optimization:
- Fix \( z \), update \( \beta \);
  (SVMLight: \url{http://svmlight.joachims.org/} or LIBLINEAR: \url{http://www.csie.ntu.edu.tw/~cjlin/liblinear/})
- Fix \( \beta \), update \( z \).

\[
F_0 \cdot \phi(H, p_0) + \sum_{i=1}^{n} \left( F_i \cdot \phi(H, p_i) + a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot \left( \tilde{x}_i^2, \tilde{y}_i^2 \right) \right)
\]
TIPS

• You do not need to implement the components that are not covered by this slides (e.g. mixture of models, dynamic programming for finding deformable parts).
• Check the online code (run it) and understand the method better;
• Test each component of your method independently;
• Save partially trained models
• First big picture, then details.
• Refer to the PAMI paper if you want to understand the approach better or further improve the performance.
TIPS

• Mex file to accelerate your code.
• Detection might be slow, so it is OK for you to use a subset of the training or testing images. (at least 1000 training / testing images recommended)
• OK to use some parts of the online code, but you need to mention them in your write-up.
• Start early.
Questions?