Deformable Part Models

References:

Felzenszwalb, Girshick, McAllester and Ramanan, “Object Detection with Discriminatively Trained Part Based Models,” PAMI 2010

Code available at http://www.cs.berkeley.edu/~rbg/latent/
Recall: Dalal and Triggs, 2005

- Detect upright pedestrians
- Histogram of oriented gradient feature vector
- Linear SVM classifier; sliding window detector

\[ x_i \in \mathbb{R}^d, \text{ with } d = 1024 \]

\[ f(x) = w^T x + b \]
Today: Deformable Part Models

Model has a root filter plus deformable parts
Motivation

• Simple template-based object models lack ability to handle geometric deformations due to articulation and pose.

• Simple bag of words models have no trouble with deformations, but are limited in their ability to finely localize objects.

• Want to make more expressive models based on geometric deformations of a canonical configuration of object parts.
Motivation

• Problem: More expressive object models are difficult to train because they often use latent (unobserved/unlabeled) information.

• Example: learning a part-based model from images where only bounding boxes are labeled.

What we have: bounding box

What we want: torso, head, arms and legs
DPM Innovations

- Star-structured graph model to represent body parts and their geometric relations
- Mixtures of models to handle large variations in viewpoint / pose
- Latent-SVM formulation
- Data mining of hard negative examples
- PCA on HOG features for dimension reduction
- Performance on PASCAL challenge dataset
History: Pictorial Structures

- Introduced by Fischler and Elschlager in 1973
- Part-based models:
  - Each part represents local visual properties
  - “Springs” capture spatial relationships

Matching model to image involves joint optimization of part locations “stretch and fit”
History: Pictorial Structures

Goal: alignment of part model with features in an image.
History: Pictorial Structures

Felzenswalb showed that the best alignment of parts for a tree-structured model can be computed efficiently using dynamic programming.

Image: [Felzenszwalb and Huttenlocher 05]
Pictorial Structures: Matching

- Model is represented by a graph $G = (V, E)$
  - $V = \{v_1, ..., v_n\}$ are the parts
  - $(v_i, v_j) \in E$ indicates a connection between parts
- $m_i(l_i)$ is score for placing part $i$ at location $l_i$
- $d_{ij}(l_i, l_j)$ is a deformation cost
- Optimal configuration for the object is $L = (l_1, ..., l_n)$ maximizing

$$E(L) = \sum_{i=1}^{n} m_i(l_i) - \sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j)$$
From Pictorial Structures to DPM

Model has a root filter plus deformable parts
Histogram of Gradient (HOG) Features

- Image is partitioned into 8x8 pixel blocks
- In each block we compute a histogram of gradient orientations
  - Invariant to changes in lighting, small deformations, etc.
- We compute features at different resolutions (pyramid)
Filters

- Filters are rectangular templates defining weights for features
- Score is dot product of filter and subwindow of HOG pyramid

HOG pyramid

Score of $H$ at this location is $H \cdot W$

Dalal and Triggs, single template
Object Hypothesis

Score is sum of filter scores minus deformation costs

Multiscale model captures features at two-resolutions
Parts are represented at twice the resolution of the root filter.

Slide from Pedro Felzenswalb
Scoring with Linear Classifiers

- Score of model is sum of filter scores plus deformation scores
  - Bounding box in training data specifies that score should be high for some placement in a range

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

- \(w\) is a model
- \(x\) is a detection window
- \(z\) are filter placements
More Specifically...

\[ z = (p_1, \ldots, p_n) \]

\[
\text{score}(l, p_0) = \max_{p_1, \ldots, p_n} \sum_{i=0}^{n} m_i(l, p_i) - \sum_{i=1}^{n} d_i(p_0, p_i)
\]

Filter scores: \( m_i(l, p_i) = w_i \cdot \phi(l, p_i) \)

Spring costs: \( d_i(p_0, p_i) = d_i \cdot (dx^2, dy^2, dx, dy) \)

Like in Dalal&Triggs but for multiple parts

e.g. \( d_i = [1,1,0,0] \) yields squared distance error

Slide from Ross Girshick
Scoring with Linear Classifiers

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

- \( w \) is a model
- \( x \) is a detection window
- \( z \) are filter placements

concatenation of filters and deformation parameters:

- \([w_0, w_1, \ldots, w_n, \text{d1, d2, \ldots, d}_n]\)

concatenation of features and part displacements:

- \([\Phi(l, p_0), \ldots, \Phi(l, p_n), (dx_1^2, dy_1^2, dx_1, dy_1), \ldots, (dx_n^2, dy_n^2, dx_n, dy_n)]\)
Detection Overview

- Feature map
- Feature map at twice the resolution
- Response of root filter
- Response of part filters
- Transformed responses
- Color encoding of filter response values
- Combined score of root locations
Handling Large Variation in Viewpoints
Handling Large Variation in Viewpoints

Good generalization properties on Doctor Dolittle’s farm

\[
\left( \frac{\text{Horse} + \text{Llama}}{2} \right) = \text{Unknown}
\]

This was supposed to detect horses

Slide from Ross Girshick
Solution: Use Mixture Model

Mixture component 1: Bicycles viewed from side

Mixture component 2: Bicycles viewed from front/rear
Solution: Use a Mixture Model

Data driven: aspect, occlusion modes, subclasses

Person mixture: 2 components

Car mixture: 3 components

FMR CVPR ’08: AP = 0.27 (person)

FGMR PAMI ’10: AP = 0.36 (person)

One model for head+torso, one for full body

One model for side, one for front and one for 45 degrees.

Slide from Ross Girshick
Training

- Training data consists of images with labeled bounding boxes
- Need to learn the model structure, filters and deformation costs
Part 2: DPM parameter learning

fixed model \textit{structure}
Part 2: DPM parameter learning

fixed model *structure*

training images

\[ y +1 \]

Slide from Ross Girshick
Part 2: DPM parameter learning

fixed model structure

training images

Slide from Ross Girshick
Part 2: DPM parameter learning

Parameters to learn:
- biases (per component)
- deformation costs (per part)
- filter weights

Slide from Ross Girshick
Recall: SVM Training

1. Maximize margin $\frac{2}{||w||}$

2. Correctly classify all training data points:
   - $x_i$ positive ($y_i = 1$): $x_i \cdot w + b \geq 1$
   - $x_i$ negative ($y_i = -1$): $x_i \cdot w + b \leq -1$

**Quadratic optimization problem:**

Minimize $\frac{1}{2} w^T w$

Subject to $y_i (w \cdot x_i + b) \geq 1$

One constraint for each training point.

rewrite

$$w^* = \underset{w}{\arg\min} \quad \lambda ||w||^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i))$$
Aside: SVM and Hinge Loss

To maximize the margin, which is $2 / \|w\|$,

This is a consequence of encoding the constraints into the objective function using Lagrange multipliers. For constraints that are satisfied (the appropriate inequality holds), the value of the corresponding Lagrange multiplier becomes 0.

Intuition: the inequality constraints $f_w(x^+) > 1$ and $f_w(x^-) < -1$ contribute a linear penalty when they are not satisfied, but are not penalized when the inequalities hold.
SVM vs LSVM Training

\[
w^* = \arg\min_w \lambda \|w\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i))
\]

**SVM**

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

**LSVM**

\[f_w(x_i) = \max_{z \in Z(x_i)} w \cdot \Phi(x_i, z)\]

Max over latent positions for part \(x_i\)

This introduces some difficulties during training
SVM vs LSVM Training

\[ w^* = \arg\min_w \lambda \|w\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i)) \]

**SVM**

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

Convex problem, guaranteed optimal solution!

**LSVM**

\[ f_w(x_i) = \max_{z \in Z(x_i)} w \cdot \Phi(x_i, z) \]

Only semi-convex, no guaranteed optimal solution
Positive examples ($y = +1$)

$x$ specifies an image and bounding box

We want

$$f_w(x) = \max_{z \in Z(x)} w \cdot \Phi(x, z)$$

to score $\geq +1$

$Z(x)$ includes all $z$ with more than 70% overlap with ground truth
Negative examples ($y = -1$)

$x$ specifies an image and a HOG pyramid location $p_0$

We want

$$f_w(x) = \max_{z \in Z(x)} w \cdot \Phi(x, z)$$

...to score $\leq -1$

$Z(x)$ restricts the root to $p_0$ and allows any placement of the other filters

Slide from Ross Girshick
How we learn parameters: latent SVM

\[ E(\mathbf{w}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i \max \{0, 1 - y_i f_{\mathbf{w}}(x_i)\} \]
How we learn parameters: latent SVM

\[ E(w) = \frac{1}{2} \| w \|^2 + C \sum_i \max\{0, 1 - y_i f_w(x_i)\} \]

\[ E(w) = \frac{1}{2} \| w \|^2 + C \sum_{i \in P} \max\{0, 1 - \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\} \]

\[ + C \sum_{i \in N} \max\{0, 1 + \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\} \]
How we learn parameters: latent SVM

\[ E(w) = \frac{1}{2}\|w\|^2 + C \sum \max\{0, 1 - y_i f_w(x_i)\} \]

\[ E(w) = \frac{1}{2}\|w\|^2 + C \sum_{i \in P} \max\{0, 1 - \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\} \]

\[ + C \sum_{i \in N} \max\{0, 1 + \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\} \]

+ score

Slide from Ross Girshick
How we learn parameters: latent SVM

$$E(w) = \frac{1}{2}||w||^2 + C \sum_{i} \max\{0, 1 - y_if_w(x_i)\}$$

$$E(w) = \frac{1}{2}||w||^2 + C \sum_{i \in P} \max\{0, 1 - \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\}$$

$$+ C \sum_{i \in N} \max\{0, 1 + \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\}$$
Since whole objective function is no longer convex, there is no longer a single global optimum, so no easy solution method.

Latent SVM objective is convex in the negatives

but not in the positives

>> “semi-convex”
Convex upper bound on loss

\[
\max\{0, 1 - \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\}
\]

Approximation by upper bound

\[
\max\{0, 1 - w \cdot \Phi(x_i, Z_{Pi})\}
\]

convex
Auxiliary objective

Let $Z_P = \{Z_{P1}, Z_{P2}, \ldots \}$

$$E(w, Z_P) = \frac{1}{2} \|w\|^2 + C \sum_{i \in P} \max\{0, 1 - w \cdot \Phi(x_i, Z_{P_i})\}$$

$$+ C \sum_{i \in N} \max\{0, 1 + \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\}$$

Current estimates of latent configurations that yield upper bound for each positive example.
Auxiliary objective

Let $Z_P = \{Z_{P1}, Z_{P2}, \ldots \}$

$$E(w, Z_P) = \frac{1}{2}||w||^2 + C \sum_{i \in P} \max \{0, 1 - w \cdot \Phi(x_i, Z_{P_i})\}$$

$$+ C \sum_{i \in N} \max \{0, 1 + \max_{z \in Z(x)} w \cdot \Phi(x_i, z)\}$$

Yields upper bound for whole objective function.

Note that $E(w, Z_P) \geq \min_{Z_P} E(w, Z_P) = E(w)$

Current estimates of latent configurations that yield upper bound for each positive example.
Auxiliary objective

Let \( Z_P = \{ Z_{P1}, Z_{P2}, \ldots \} \)

\[
E(w, Z_P) = \frac{1}{2} \|w\|^2 + C \sum_{i \in P} \max\{0, 1 - w \cdot \Phi(x_i, Z_{P_i})\} + C \sum_{i \in N} \max\{0, 1 + \max_z w \cdot \Phi(x_i, z)\}
\]

Yields upper bound for whole objective function.

Note that \( E(w, Z_P) \geq \min_{Z_P} E(w, Z_P) = E(w) \)

And it is a tight upper bound

\( w^* = \min_{w, Z_P} E(w, Z_P) = \min_w E(w) \)
Auxiliary objective

\[ w^* = \min_{w, Z_p} E(w, Z_p) = \min_w E(w) \]

Unfortunately, this isn’t easy to optimize either, not to mention it is based on estimates of upper bound latent configurations, which could be wrong.
Auxiliary objective

\[ w^* = \min_{w, z_p} E(w, z_p) = \min_w E(w) \]

Iterative approach to solving this.
NOT guaranteed to find optimal solution
unless you start with good initial estimates.

Find stationary point by coordinate descent on \( E(w, z_p) \)
Auxiliary objective

\[ w^* = \min_{w, Z_P} E(w, Z_P) = \min_w E(w) \]

Iterative approach to solving this.
NOT guaranteed to find optimal solution
unless you start with good initial estimates.

Find stationary point by coordinate descent on \( E(w, Z_P) \)

Initialization: either by picking a \( w(0) \) (or \( Z_P \))
Auxiliary objective

\[ w^* = \min_{w, Z_P} E(w, Z_P) = \min_{w} E(w) \]

Iterative approach to solving this.
NOT guaranteed to find optimal solution unless you start with good initial estimates.

Find stationary point by coordinate descent on \( E(w, Z_P) \)

Initialization: either by picking a \( w_0 \) (or \( Z_P \))

Step 1:

\[ Z_{Pi} = \arg\max_{w_t} \Phi(x_i, z) \quad \forall i \in P \]
Auxiliary objective

\[ \mathbf{w}^* = \min_{\mathbf{w}, Z_P} E(\mathbf{w}, Z_P) = \min_{\mathbf{w}} E(\mathbf{w}) \]

Iterative approach to solving this. NOT guaranteed to find optimal solution unless you start with good initial estimates.

Find stationary point by coordinate descent on \( E(\mathbf{w}, Z_P) \)

Initialization: either by picking a \( \mathbf{w}_{(0)} \) (or \( Z_P \))

Step 1:
\[ Z_{Pi} = \arg\max_{z \in Z(x_i)} \mathbf{w}_{(t)} \cdot \Phi(x_i, z) \quad \forall i \in P \]

Step 2:
\[ \mathbf{w}_{(t+1)} = \arg\min_{\mathbf{w}} E(\mathbf{w}, Z_P) \]
Step 1

\[ Z_{Pi} = \underset{z \in Z(x_i)}{\text{argmax}} \; w(t) \cdot \Phi(x_i, z) \quad \forall i \in P \]

This is just detection: Easy to compute
Step 2

$$\min_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i \in P} \max \{ 0, 1 - \mathbf{w} \cdot \Phi(x_i, Z_{pi}) \}$$

$$+ C \sum_{i \in N} \max \{ 0, 1 + \max_{z \in Z(x)} \mathbf{w} \cdot \Phi(x_i, z) \}$$

Convex  Easy to solve
What about the model structure? Can we learn that as well?

Model structure
- # components
- # parts per component
- root and part filter shapes
- part anchor locations

training images

Slide from Ross Girshick
Learning model structure

Initialize a mixture model to handle different viewpoints

Initialization:
Split positives by aspect ratio

Warp to common size

Train Dalal & Triggs model for each aspect ratio on its own
Learning model structure

(a) Car component 1 (Phase 1)  (b) Car component 2 (Phase 1)  (c) Car component 3 (Phase 1)

(d) Car component 1 (Phase 2)  (e) Car component 2 (Phase 2)  (f) Car component 3 (Phase 2)

Use D&T filters as initial $\mathbf{w}$ for LSVM training

Merge components

Root filter placement and component choice are latent
Initialize parts for each component

Refine by training.

Add parts to cover high-energy areas of root filters

Note: parts are constrained to be symmetrically placed

Continue training model with LSVM
Summary: Model Learning

- Learn mixture model (component root filters)
- Learn part model for each mixture component
- LSVM used during both phases
Mining Hard Negatives

Typical dataset

300 – 8,000 positive examples

500 million to 1 billion negative examples (not including latent configurations!)

Detectors problems are highly unbalanced (many more negatives than positives). We REALLY don’t want to consider all negative examples while training!

Slide from Ross Girshick
Mining “Hard Negative” Examples

• Set of negative training examples is HUGE (a single image can yield $10^5$ examples for a scanning window classifier).

• Construct training data from the positive instances and a selection of the “hard negative” instances, via bootstrapping
  – Train using subset of negative examples
  – Apply resulting classifier to all negative examples, and take those incorrectly classified as a new set of hard negatives
  – Retrain, and perhaps repeat
PCA Analysis of HOG

• Collected 36-dimensional HOG features at a variety of scales over large number of images

• PCA analysis to identify top eigenvectors
Recall: HOG Features

- Tile window into 8 x 8 pixel cells
- Each cell represented by HOG

Each cell contains a histogram of gradient orientations, weighted by gradient magnitude.
HoG Feature Extraction: Blocks

2x2 block of cells → normalize → \[ \ldots \]

Independent contrast normalization over 2x2 blocks of cells.
HoG Feature Extraction: Blocks

2x2 block of cells

normalize

[ , , , , ..., ]

Independent contrast normalization over 2x2 blocks of cells.

Each histogram gets normalized 4 different times.

9 orientations x 4 orientations = 36 feature values for each cell
PCA Analysis of HOG

- Collected 36-dimensional HOG features at a variety of scales over a large number of images
- PCA analysis to identify top eigenvectors
PCA Analysis of HOG

• Collected 36-dimensional HOG features at a variety of scales over large number of images
• PCA analysis to identify top eigenvectors
• Top 11 account for nearly all of the variation
PCA Analysis of HOG

- Projection onto PCA bases is expensive
- However, note that top eigenvectors are constant over rows or cols
- Analytic projection to 13 dimensions: sum over each of 4 rows and 9 columns
**Bounding Box Prediction**

Problem: Location and size of root filter (red rectangle) may not correlate perfectly with bounding boxes in human-labeled data.

Solution: Learn four linear regression functions to predict $x_1, y_1, x_2, y_2$ of bounding box from a vector containing locations of each part (including root) and the width (scale) of the root. 

$$(2n+3) \rightarrow x_1 \text{ (or } y_1, x_2, y_2)$$
Non-Maximum Suppression

- Sliding window detectors tend to produce many overlapping detections on same object
Non-Maximum Suppression

• Greedy solution: run through list of detections in descending order of confidence, removing ones that overlap more than 50% with a higher-scoring detection.
Part III: PASCAL Challenge

- ~10,000 images, with ~25,000 target objects
  - Objects from 20 categories (person, car, bicycle, cow, table...)
  - Objects are annotated with labeled bounding boxes
Example Annotations from the Dataset
Formalizing the object detection task

Many possible ways, this one is popular:

Input

Desired output
Formalizing the object detection task

Many possible ways, this one is popular:

Performance summary:

Average Precision (AP)
0 is worst 1 is perfect

Bounding box overlap of 50% needed, to detect
Learned Models

Slide from Pedro Felzenswalb
Example Results

Slide from Pedro Felzenswalb
More Results

Slide from Pedro Felzenswalb
Some False Positives

Car

Horse

Person (Insufficient overlap)
Some False Positives

Sofa

Bottle

Cat

(Due to insufficient overlap)
Overall Results

- 9 systems competed in the 2007 challenge
- Out of 20 classes we get:
  - First place in 10 classes
  - Second place in 6 classes
- Some statistics:
  - It takes ~2 seconds to evaluate a model in one image
  - It takes ~3 hours to train a model
  - MUCH faster than most systems
Summary: DPM Innovations

- Star-structured graph model to represent body parts and their geometric relations
- Mixtures of models to handle large variations in viewpoint / pose
- Latent-SVM formulation
- Data mining of hard negative examples
- PCA on HOG features for dimension reduction
- Performance on PASCAL challenge dataset