Graphical Models in Computer Vision

Gerard Pons-Moll

Max Planck Institute for Intelligent Systems
Perceiving Systems

June 27, 2016
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.04.2016</td>
<td>Introduction</td>
</tr>
<tr>
<td>18.04.2016</td>
<td>Graphical Models 1</td>
</tr>
<tr>
<td>25.04.2016</td>
<td>Graphical Models 2 <em>(Sand 6/7)</em></td>
</tr>
<tr>
<td>02.05.2016</td>
<td>Graphical Models 3</td>
</tr>
<tr>
<td>09.05.2016</td>
<td>Graphical Models 4</td>
</tr>
<tr>
<td>23.05.2016</td>
<td>Body Models 1</td>
</tr>
<tr>
<td>30.05.2016</td>
<td>Body Models 2</td>
</tr>
<tr>
<td>06.06.2016</td>
<td>Body Models 3</td>
</tr>
<tr>
<td>13.06.2016</td>
<td>Body Models 4</td>
</tr>
<tr>
<td>20.06.2016</td>
<td>Object Detection 1</td>
</tr>
<tr>
<td>27.06.2016</td>
<td>Object Detection 2</td>
</tr>
<tr>
<td>04.07.2016</td>
<td>Stereo</td>
</tr>
<tr>
<td>11.07.2016</td>
<td>Optical Flow</td>
</tr>
<tr>
<td>18.07.2016</td>
<td>Segmentation</td>
</tr>
</tbody>
</table>
Todays topic

Object Detection
   - Recap
   - Part-based Models (DPM)

Object Tracking
   - Introduction
   - Bayes Filter
   - Assignment Problem
   - Graph-based Tracking
Part-based Models
Structureless vs. Rigid Models

Bag of words

Deformable Part Models [P. Felzenszwalb et al., PAMI 2010]

Dalal and Triggs, CVPR 2005

Structureless

Rigid
Why Parts?

Why do want to model parts?

- Useful to handle intra-class geometry variation

[Fergus, 2005]
Why do want to model parts?

- Useful to handle intra-class geometry variation
- Objects may be globally different but they have parts in common

[Fergus, 2005]
Why Parts?

Why do want to model parts?

- Useful to handle intra-class geometry variation
- Objects may be globally different but they have parts in common
- Model prior knowledge of relative location and size

[Fergus, 2005]
Part-based Models

Why Parts?

Deformable parts can handle slight variations in pose:

![Diagram showing examples of deformable parts handling pose variations.](image)

Figure 1. Matching with a single template. The schematic template of a frontal face is shown in a). Slight rotations of the face in the image plane b) and in depth c) lead to considerable discrepancies between template and face.

Figure 2. Matching with a set of component templates. The schematic component templates for a frontal face are shown in a). Shifting the component templates can compensate for slight rotations of the face in the image plane b) and in depth c).

[Heisele et al, 2001]
Why Parts?

Easier to handle occlusions:

[Felzenszwalb et al, 2010]
Connectivity Structures

a) Constellation [13]  
b) Star shape [9, 14]  
c) k-fan (k = 2) [9]  
d) Tree [12]

e) Bag of features [10, 21]  
f) Hierarchy [4]  
g) Sparse flexible model

Fig. 1. Graphical geometric models of priors. Note that Xi represents a model part.  
[Carneiro & Lowe, 2006]
Connectivity Structures

**Constellation Model** [Fergus et al, 2003]

**Efficient Pictorial Structures** [Felzenszwalb & Huttenlocher, 2000]
Connectivity Structures

**Implicit Shape Model [Leibe et al, 2004]**

**Poselets [Bourdev et al, 2009]**
Deformable Part-based Model (DPM)

- 2-scale model
  - Whole object (root)
  - Deformable parts

- HoG representation + SVM training to obtain robust root and part detectors
- Efficient algorithm for detection

[Felzenszwalb et al., 2010]
Deformable Part-based Model (DPM)

- 2-scale model
  - Whole object (root)
  - Deformable parts
- HoG representation + SVM training to obtain robust root and part detectors
Deformable Part-based Model (DPM)

- 2-scale model
  - Whole object (root)
  - Deformable parts
- HoG representation + SVM training to obtain robust root and part detectors
- Efficient algorithm for detection
Deformable Part-based Model (DPM)

- 2-scale model
  - Whole object (root)
  - Deformable parts
- HoG representation + SVM training to obtain robust root and part detectors
- Efficient algorithm for detection
- [Felzenszwalb et al., 2010]
Deformable Part-based Model (DPM)

Models are fully trained from bounding boxes alone (weak labels). The part locations are unknown (i.e., latent variables).
Different viewpoints are modeled using different models (=components). Each component has a global template (root) + part templates.
DPM Bicycle Model with 2 Components

Each component has a root filter $F_0$ and $n$ part filters $(F_i,v_i,d_i)$. 

- root filter
- part filter
- deformation models
Part-based Models

Multiscale Model captures Features at two Resolutions

The score is a sum of filter scores minus part deformation costs.
Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)
$$
Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)
\]

where:

- \(p_0, \ldots, p_n\) denotes an object hypothesis, specified by root \((i = 0)\) and part \((i \geq 1)\) locations, with \(p_i = (x_i, y_i, l_i)^T\)
Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

where:

▶ $\mathbf{p}_0, \ldots, \mathbf{p}_n$ denotes an object hypothesis, specified by root ($i = 0$) and part ($i \geq 1$) locations, with $\mathbf{p}_i = (x_i, y_i, l_i)^T$
  ▶ $x_i, y_i$ denote pixel location
  ▶ $l_i$ specifies the level in the pyramid
Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $p_0, \ldots, p_n$ denotes an object hypothesis, specified by root ($i = 0$) and part ($i \geq 1$) locations, with $p_i = (x_i, y_i, l_i)^T$
  - $x_i, y_i$ denote pixel location
  - $l_i$ specifies the level in the pyramid
- $f_0, \ldots, f_n$ are learned filter weights of the model
Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $p_0, \ldots, p_n$ denotes an object hypothesis, specified by root ($i = 0$) and part ($i \geq 1$) locations, with $p_i = (x_i, y_i, l_i)^T$
  - $x_i, y_i$ denote pixel location
  - $l_i$ specifies the level in the pyramid
- $f_0, \ldots, f_n$ are learned filter weights of the model
- $\phi(p_i)$ are the HoG features for the region specified by $p_i$
Part-based Models

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)
\]

where:

- \( p_0, \ldots, p_n \) denotes an object hypothesis, specified by root \((i = 0)\) and part \((i \geq 1)\) locations, with \( p_i = (x_i, y_i, l_i)^T \)
  - \( x_i, y_i \) denote pixel location
  - \( l_i \) specifies the level in the pyramid
- \( f_0, \ldots, f_n \) are learned filter weights of the model
- \( \phi(p_i) \) are the HoG features for the region specified by \( p_i \)
- \( d_i \in \mathbb{R}^2 \) are deformation parameters
Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)
\]

where:

- \( p_0, \ldots, p_n \) denotes an object hypothesis, specified by root \((i = 0)\) and part \((i \geq 1)\) locations, with \( p_i = (x_i, y_i, l_i)^T \)
  - \( x_i, y_i \) denote pixel location
  - \( l_i \) specifies the level in the pyramid
- \( f_0, \ldots, f_n \) are learned filter weights of the model
- \( \phi(p_i) \) are the HoG features for the region specified by \( p_i \)
- \( d_i \in \mathbb{R}^2 \) are deformation parameters
- \( dx_i, dy_i \) denote the rel. displacement of \( p_i \) from its anchor \( v_i \):
  \[
  (dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i)
  \]
Score of a Hypothesis

DPM score from previous slide:

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)
\]
Score of a Hypothesis

DPM score from previous slide:

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)
\]

This can be also written as a linear combination

\[
\text{score}(p_0, \ldots, p_n) = \beta^T \cdot \psi(p_0, \ldots, p_n)
\]
Score of a Hypothesis

DPM score from previous slide:

$$\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)$$

This can be also written as a linear combination

$$\text{score}(p_0, \ldots, p_n) = \beta^T \cdot \psi(p_0, \ldots, p_n)$$

where:

- $$\beta = (f_0, \ldots, f_n, d_1, \ldots, d_n)$$
Score of a Hypothesis

DPM score from previous slide:

$$\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} f_i^T \cdot \phi(p_i) - \sum_{i=1}^{n} d_i^T \cdot (dx_i^2, dy_i^2)$$

This can be also written as a linear combination

$$\text{score}(p_0, \ldots, p_n) = \beta^T \cdot \psi(p_0, \ldots, p_n)$$

where:

- $\beta = (f_0, \ldots, f_n, d_1, \ldots, d_n)$
- $\psi(p_0, \ldots, p_n) = (\phi(p_0), \ldots, \phi(p_n), -(dx_1^2, dy_1^2), \ldots, -(dx_n^2, dy_n^2))$
Score of a Hypothesis

DPM score from previous slide:

\[
\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)
\]

This can be also written as a linear combination

\[
\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \mathbf{\beta}^T \cdot \psi(\mathbf{p}_0, \ldots, \mathbf{p}_n)
\]

where:

- \( \mathbf{\beta} = (\mathbf{f}_0, \ldots, \mathbf{f}_n, \mathbf{d}_1, \ldots, \mathbf{d}_n) \)
- \( \psi(\mathbf{p}_0, \ldots, \mathbf{p}_n) = (\phi(\mathbf{p}_0), \ldots, \phi(\mathbf{p}_n), -(dx_1^2, dy_1^2), \ldots, -(dx_n^2, dy_n^2)) \)

This illustrates the connection to linear classifiers:
The DPM learns the model parameters using the latent SVM framework.
Object Detection with DPM

**Inference:** Given a root location $p_0$, calculate the detection score as:

$$\text{score}(p_0) = \max_{p_1, \ldots, p_n} \text{score}(p_0, \ldots, p_n)$$

$$= \max_{p_1, \ldots, p_n} \beta^T \cdot \psi(p_0, \ldots, p_n)$$

$$= \max_z \beta^T \cdot \psi(p_0, z)$$

This maximizes the score by varying the parts $z = (p_1, \ldots, p_n)$ given the root location $p_0$.

High scoring root locations define detections.

This maximization (which is exponential in the number of parts $n$) can be efficiently computed using dynamic programming and generalized distance transforms.

Which graphical model/inference problem do we have here?
Object Detection with DPM

**Inference:** Given a root location $p_0$, calculate the detection score as:

\[
\text{score}(p_0) = \max_{p_1, \ldots, p_n} \text{score}(p_0, \ldots, p_n)
\]

\[
= \max_{p_1, \ldots, p_n} \beta^T \cdot \psi(p_0, \ldots, p_n)
\]

\[
= \max_z \beta^T \cdot \psi(p_0, z)
\]

- This maximizes the score by varying the parts $z = (p_1, \ldots, p_n)^T$ given the root location $p_0$.
**Object Detection with DPM**

**Inference:** Given a root location \( p_0 \), calculate the detection score as:

\[
\text{score}(p_0) = \max_{p_1, \ldots, p_n} \text{score}(p_0, \ldots, p_n) \\
= \max_{p_1, \ldots, p_n} \beta^T \cdot \psi(p_0, \ldots, p_n) \\
= \max_z \beta^T \cdot \psi(p_0, z)
\]

- This maximizes the score by varying the parts \( z = (p_1, \ldots, p_n)^T \) given the root location \( p_0 \).
- High scoring root locations define detections.
**Inference:** Given a root location $p_0$, calculate the detection score as:

$$
score(p_0) = \max_{p_1, \ldots, p_n} score(p_0, \ldots, p_n)
$$

$$= \max_{p_1, \ldots, p_n} \beta^T \cdot \psi(p_0, \ldots, p_n)
$$

$$= \max_z \beta^T \cdot \psi(p_0, z)$$

- This maximizes the score by varying the parts $z = (p_1, \ldots, p_n)^T$ given the root location $p_0$.
- High scoring root locations define detections.
- This maximization (which is exponential in the number of parts $n$) can be efficiently computed using dynamic programming and generalized distance transforms.
Object Detection with DPM

**Inference:** Given a root location $p_0$, calculate the detection score as:

$$\text{score}(p_0) = \max_{p_1, \ldots, p_n} \text{score}(p_0, \ldots, p_n)$$

$$= \max_{p_1, \ldots, p_n} \beta^T \cdot \psi(p_0, \ldots, p_n)$$

$$= \max_z \beta^T \cdot \psi(p_0, z)$$

- This maximizes the score by varying the parts $z = (p_1, \ldots, p_n)^T$ given the root location $p_0$

- High scoring root locations define detections

- This maximization (which is exponential in the number of parts $n$) can be efficiently computed using dynamic programming and generalized distance transforms

- Which graphical model/inference problem do we have here?
Fast Evaluation of Filter Responses

Head filter

Input image

Filter response at level $l$:
$$R_l(x, y) = f^T \cdot \phi(x, y, l)$$

Transformed response:
$$D_l(x, y) = \max_{dx, dy} (R_l(x + dx, y + dy) - d^T \cdot (dx^2, dy^2))$$
Part-based Models

Pipeline

feature map

response of root filter

feature map at twice the resolution

model

transformed responses

response of part filters

combined score of root locations

color encoding of filter response values
Detection Results

Detection results after non-maxima-suppression (mode finding)
Given annotated images and background images we need to find:

- Root and part filter weights
- Deformation weights
Latent SVM Training

Learn a classifier that scores an example $p_0$ as

$$\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$$
Latent SVM Training

Learn a classifier that scores an example $p_0$ as

$$\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$$

where

- $\beta$ are the model parameters from before
  (filter and deformation weights)
Latent SVM Training

Learn a classifier that scores an example $p_0$ as

$$\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$$

where

- $\beta$ are the model parameters from before (filter and deformation weights)
- $z = (p_1, \ldots, p_n)$ are latent values
Latent SVM Training

Learn a classifier that scores an example $p_0$ as

$$\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$$

where

- $\beta$ are the model parameters from before (filter and deformation weights)
- $z = (p_1, \ldots, p_n)$ are latent values
- Training data: $\{(p_0^{(1)}, y^{(1)}), \ldots, (p_0^{(n)}, y^{(n)})\}$ with $y^{(i)} \in \{-1, +1\}$
Latent SVM Training

Learn a classifier that scores an example $p_0$ as

$$score_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$$

where

- $\beta$ are the model parameters from before (filter and deformation weights)
- $z = (p_1, \ldots, p_n)$ are latent values
- Training data: $\{(p_0^{(1)}, y^{(1)}), \ldots, (p_0^{(n)}, y^{(n)})\}$ with $y^{(i)} \in \{-1, +1\}$
- We want to find $\beta$ such that: $y^{(i)} \cdot score_\beta(p_0^{(i)}) > 0$
Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$
\text{score}_\beta(\mathbf{p}_0) = \max_z \beta^T \cdot \psi(\mathbf{p}_0, z)
$$

where

- $\beta$ are the model parameters from before (filter and deformation weights)
- $z = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ are latent values
- Training data: $\{(\mathbf{p}_0^{(1)}, y^{(1)}), \ldots, (\mathbf{p}_0^{(n)}, y^{(n)})\}$ with $y^{(i)} \in \{-1, +1\}$
- We want to find $\beta$ such that: $y^{(i)} \cdot \text{score}_\beta(\mathbf{p}_0^{(i)}) > 0$
- Positive examples: pos. score, negative examples: neg. score
Latent SVM Training

Learn a classifier that scores an example \( p_0 \) as

\[
\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)
\]

where

- \( \beta \) are the model parameters from before (filter and deformation weights)
- \( z = (p_1, \ldots, p_n) \) are latent values
- Training data: \( \{(p_0^{(1)}, y^{(1)}), \ldots, (p_0^{(n)}, y^{(n)})\} \) with \( y^{(i)} \in \{-1, +1\} \)
- We want to find \( \beta \) such that: \( y^{(i)} \cdot \text{score}_\beta(p_0^{(i)}) > 0 \)
- Positive examples: pos. score, negative examples: neg. score

We minimize the following regularized latent SVM objective:

\[
L_D(\beta) = \frac{1}{2} ||\beta||^2 + C \sum_{i=1}^n \max \left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)})\right)
\]

regularizer  loss function
Loss Functions

\[
L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_{\beta}(p^{(i)}_0) \right)
\]

- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)

![Graph showing comparison of zero-one loss, hinge loss, and logistic loss](image.png)
Loss Functions

\[ L_D(\beta) = \frac{1}{2}||\beta||^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)}) \right) \]

- Hinge and logistic loss approximate the misclassification error! (they are upper bounds)
- Important properties:
  - Robustness
Part-based Models

Loss Functions

\[ L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)})\right) \]

- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity

![Graph showing comparison between zero-one loss, hinge loss, and logistic loss](Image)
Loss Functions

\[ L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_{\beta}(p_0^{(i)}) \right) \]

- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity
  - Smoothness

![Graph showing comparison between zero-one loss, Hinge loss, and Logistic loss](image-url)
Loss Functions

\[ L_D(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y(i) \cdot \text{score}_{\beta}(p_0^{(i)}) \right) \]

- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity
  - Smoothness
- 0-1 loss NP hard
Part-based Models

**Loss Functions**

\[ L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)})\right) \]

- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity
  - Smoothness
- 0-1 loss NP hard
- SVM uses Hinge loss
Semi Convexity

We guaranteed to find minimizer \( \beta^* = \arg\min_{\beta} L_D(\beta) \) using gradient decent if and exactly if \( L_D(\beta) \) is convex! Is \( L_D(\beta) \) convex?
Semi Convexity

We guaranteed to the find minimizer $\beta^* = \arg\min_{\beta} L_D(\beta)$ using gradient decent if and exactly if $L_D(\beta)$ is convex! Is $L_D(\beta)$ convex?

- The sum and maximum of convex functions is convex
Semi Convexity

We guaranteed to the find minimizer $\beta^* = \arg\min_{\beta} L_D(\beta)$ using gradient decent if and exactly if $L_D(\beta)$ is convex! Is $L_D(\beta)$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$ is convex in $\beta$! Why?
Semi Convexity

We guaranteed to find minimizer $\beta^* = \arg\min_{\beta} L_D(\beta)$ using gradient decent if and exactly if $L_D(\beta)$ is convex! Is $L_D(\beta)$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$ is convex in $\beta$! Why?
- $\max \left( 0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)}) \right)$ is convex?
Semi Convexity

We guaranteed to the find minimizer $\beta^* = \arg\min_{\beta} L_D(\beta)$ using gradient decent if and exactly if $L_D(\beta)$ is convex! Is $L_D(\beta)$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$ is convex in $\beta$! Why?
- $\max \left( 0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)}) \right)$ is convex? Yes, iff $y^{(i)} < 0$
Semi Convexity

We guaranteed to the find minimizer $\beta^* = \arg\min_{\beta} L_D(\beta)$ using gradient decent if and exactly if $L_D(\beta)$ is convex! Is $L_D(\beta)$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$ is convex in $\beta$! Why?
- $\max \left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)})\right)$ is convex? Yes, iff $y^{(i)} < 0$
- Thus

$$L_D(\beta) = \frac{1}{2} ||\beta||^2 + C \sum_{i=1}^{n} \max \left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)})\right)$$

is convex if all latent values for the positive examples are fixed!
Semi Convexity

We guaranteed to the find minimizer $\beta^* = \arg\min_\beta L_D(\beta)$ using gradient decent if and exactly if $L_D(\beta)$ is convex! Is $L_D(\beta)$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_\beta(p_0) = \max_z \beta^T \cdot \psi(p_0, z)$ is convex in $\beta$! Why?
- $\max \left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)})\right)$ is convex? Yes, iff $y^{(i)} < 0$
- Thus

$$L_D(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \max \left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)})\right)$$

is convex if all latent values for the positive examples are fixed!
- This is called “Semi Convexity” in [Felzenszwalb et al., 2010]
Optimization

\[ L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_\beta(p^{(i)}_0) \right) \]

\( L_D(\beta) \) is convex if we fix \( z \) for all positive examples \( (y^{(i)} > 0) \).
Optimization

\[
L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)}) \right)
\]

\(L_D(\beta)\) is convex if we fix \(z\) for all positive examples \((y^{(i)} > 0)\)

▶ Alternating Optimization
Optimization

$$L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)}) \right)$$

$L_D(\beta)$ is convex if we fix $z$ for all positive examples ($y^{(i)} > 0$)

- Alternating Optimization
  - Initialize $\beta$ and iterate:
    1. Pick best $z$ for each positive example
Optimization

\[ L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_\beta(p_0^{(i)}) \right) \]

\( L_D(\beta) \) is convex if we fix \( z \) for all positive examples \((y^{(i)} > 0)\)

- **Alternating Optimization**
  - Initialize \( \beta \) and iterate:
    1. Pick best \( z \) for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores
Optimization

\[ L_D(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_{\beta}(p_0^{(i)}) \right) \]

\( L_D(\beta) \) is convex if we fix \( z \) for all positive examples \( (y^{(i)} > 0) \)

- Alternating Optimization
  - Initialize \( \beta \) and iterate:
    1. Pick best \( z \) for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores
    3. Throw away negative examples with low score
Optimization

\[
L_D(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(p_{0}^{(i)})\right)
\]

\(L_D(\beta)\) is convex if we fix \(z\) for all positive examples \((y^{(i)} > 0)\)

- **Alternating Optimization**
  - Initialize \(\beta\) and iterate:
    1. Pick best \(z\) for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores
    3. Throw away negative examples with low score
    4. Optimize \(\beta\) via gradient descent
Optimization

\[ L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y^{(i)} \cdot \text{score}_{\beta}(p_0^{(i)}) \right) \]

\( L_D(\beta) \) is convex if we fix \( z \) for all positive examples \((y^{(i)} > 0)\)

- Alternating Optimization
  - Initialize \( \beta \) and iterate:
    1. Pick best \( z \) for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores
    3. Throw away negative examples with low score
    4. Optimize \( \beta \) via gradient descent

- The data mining / harvesting step is required as there exists an extremely large number of negatives which can’t all be included
Training Procedure

▶ For one object category, several models (=components) are trained to deal with significant appearance variations which can’t be handled by the deformable part filters (e.g., front vs. side view)
Training Procedure

- For one object category, several models (=components) are trained to deal with significant appearance variations which can’t be handled by the deformable part filters (e.g., front vs. side view)

- Coarse-to-fine training:
  1. Train root filters
For one object category, several models (=components) are trained to deal with significant appearance variations which can’t be handled by the deformable part filters (e.g., front vs. side view).

Coarse-to-fine training:
1. Train root filters
2. Initialize parts from root (greedy selection of strong coefficients)
Training Procedure

- For one object category, several models (=components) are trained to deal with significant appearance variations which can’t be handled by the deformable part filters (e.g., front vs. side view).

- Coarse-to-fine training:
  1. Train root filters
  2. Initialize parts from root (greedy selection of strong coefficients)
  3. Train final model
Trained DPM Models

Car Model

- root filters
  - coarse resolution
- part filters
  - finer resolution
- deformation models
Trained DPM Models

Person Model

root filters
coarse resolution

part filters
finer resolution

deformation models
Trained DPM Models

Cat Model

root filters
course resolution

part filters
finer resolution

deformation
models
Trained DPM Models

Bottle Model

root filters
coarse resolution

part filters
finer resolution

deformation models
Results on PASCAL VOC

high scoring true positives

high scoring false positives
Results on PASCAL VOC

- high scoring true positives
- high scoring false positives (not enough overlap)
Results on PASCAL VOC

high scoring true positives

high scoring false positives
Results on PASCAL VOC

high scoring true positives

high scoring false positives (not enough overlap)
Part-based Models

Results on PASCAL VOC

class: car, year 2006

precision

recall

- 1 Root (0.48)
- 2 Root (0.58)
- 1 Root+Parts (0.55)
- 2 Root+Parts (0.62)
- 2 Root+Parts+BB (0.64)
Results on PASCAL VOC

Precision/Recall results on Person 2008
Part-based Models

Results on PASCAL VOC

Precision/Recall results on Bird 2008
Code and Datasets

Try it yourself!

- MATLAB code available at:
  http://www.cs.berkeley.edu/~rbg/latent/
- Training requires about 4 hours for PASCAL
- Detection on one image runs in a few seconds
- Pre-trained models are available
Code and Datasets

Try it yourself!

- MATLAB code available at:
  http://www.cs.berkeley.edu/~rbg/latent/
- Training requires about 4 hours for PASCAL
- Detection on one image runs in a few seconds
- Pre-trained models are available

Useful datasets:

- PASCAL VOC:
  http://pascalvin.eecs.soton.ac.uk/challenges/VOC/
- KITTI:
  http://www.cvlibs.net/datasets/kitti/eval_object.php
Results on KITTI
Detecting 100k classes via Hashing [Dean, 2013]

**Training:**
- Learn part filters using latent SVM
- Store index of each filter in hash table
Detecting 100k classes via Hashing [Dean, 2013]

**Training:**
- Learn part filters using latent SVM
- Store index of each filter in hash table

**Detection:**
- Lookup hash table and retrieve matching filters
- Detect objects using sparse filter scores
Richer Hierarchies: Stochastic Grammars [Zhu & Mumford, 2007]
Part-based Models

3D Urban Scene Understanding [Geiger at al., 2011-2013]

Goal: Jointly infer from short videos (moving observer)

- Topology and geometry of the scene
- Semantic information (e.g., traffic situation)
3D Urban Scene Understanding [Geiger at al., 2011-2013]

http://www.cvlibs.net/projects/intersection/
http://www.cvlibs.net/software/trackbydet/