Modeling and Optimization of Classifiers with Latent Variables

Sobhan Naderi Parizi

Dissertation defense

13 January, 2016
Modeling and Optimization of Classifiers with Latent Variables

Sobhan Naderi Parizi

Dissertation defense

Thesis advisor:
Prof. Pedro Felzenszwalb

Other committee members:
Prof. Stan Sclaroff
Prof. Erik Sudderth

13 January, 2016
Classification problem
Classification problem

Binary classification

- Input
- Classifier
Classification problem

Binary classification

spam filtering
Classification problem

Binary classification

- spam filtering
- bicycle detector
Classification problem

Binary classification
- spam filtering
- bicycle detector

Multi-class classification

Class #1

Class #2

Class #3
Classification problem

Binary classification
spam filtering
bicycle detector

Multi-class classification
scene classification

classifier

? ?

staircase

? ?

bedroom

classifier

input

laundromat
Classification problem

Binary classification
- spam filtering
- bicycle detector

Multi-class classification
- scene classification
- action recognition

Classifier

Handshaking

Running

Hugging
Latent structure
Latent structure
Latent structure
Latent structure
Latent structure

Felzenszwlab, Girshick, McAleester, Ramanan,
Object Detection with Discriminatively Trained
Part Based Models, PAMI 2010
Latent structure

Felzenszwalb, Girshick, McAlester, Ramanan, Object Detection with Discriminatively Trained Part Based Models, PAMI 2010
Latent structure

Felzenszwalb, Girshick, McAlester, Ramanan,
Object Detection with Discriminatively Trained Part Based Models, PAMI 2010
Felzenszwalb, Girshick, McAlester, Ramanan,
Object Detection with Discriminatively Trained
Part Based Models, PAMI 2010
Latent structure

Felzenszwalb, Girshick, McAlester, Ramanan,
Object Detection with Discriminatively Trained Part Based Models, PAMI 2010
Latent structure

Felzenszwalb, Girshick, McAlester, Ramanan, Object Detection with Discriminatively Trained Part Based Models, PAMI 2010

Microsoft COCO dataset
Latent structure

Felzenszwlab, Girshick, McAleester, Ramanan, Object Detection with Discriminatively Trained Part Based Models, PAMI 2010

Microsoft COCO dataset
Goals of this thesis
Goals of this thesis

Capture latent structure using *latent variables*
Goals of this thesis

Capture latent structure using *latent variables*
Goals of this thesis

Capture latent structure using *latent variables*
Goals of this thesis

Capture latent structure using *latent variables*
Goals of this thesis

Capture latent structure using *latent variables*
Goals of this thesis

Capture latent structure using *latent variables*
Goals of this thesis

Capture latent structure using *latent variables*

Develop training algorithms to learn model parameters
Goals of this thesis

Capture latent structure using \textit{latent variables}

Develop training algorithms to learn model parameters

Training requires non-convex optimization
  - sensitive to initialization
  - hard to find good initialization
Goals of this thesis

Capture latent structure using *latent variables*

Develop training algorithms to learn model parameters

Training requires non-convex optimization
  - sensitive to initialization
  - hard to find good initialization

Avoid hand-engineered choices to drive training
Goals of this thesis

Capture latent structure using *latent variables*

Develop training algorithms to learn model parameters

Training requires non-convex optimization
  - sensitive to initialization
  - hard to find good initialization

Avoid hand-engineered choices to drive training

Improve existing non-convex optimization frameworks
Contributions

- Modeling
  - Automatic part discovery

- Optimization
  - Generalized Majorization-Minimization (G-MM)
  - Adding stochasticity to G-MM
Contributions

- Modeling
  - Automatic part discovery

- Optimization
  - Generalized Majorization-Minimization (G-MM)
  - Adding stochasticity to G-MM
Automatic part discovery for image classification

Image classification on MIT-Indoor scene dataset [1]

| Existing methods | Automatic part discovery |
Existing methods

Automatic part discovery


Existing methods

Automatic part discovery


Existing methods


Existing methods

Automatic part discovery


Part filters (using HOG [1])

\[ w = (w_1, \ldots, w_m), \quad w_j \in \mathbb{R}^d \]

Input image \( x \)

\[ \Psi(x, w) \in \mathbb{R}^m \]


Part filters (using HOG [1])

\[ w = (w_1, \ldots, w_m), \quad w_j \in \mathbb{R}^d \]

Input image \( x \)

\[ \phi(x, z_j) \in \mathbb{R}^d \]

\[ \Psi(x, w) \in \mathbb{R}^m \]


Part filters (using HOG [1])

\[ w = (w_1, \ldots, w_m), \quad w_j \in \mathbb{R}^d \]

Input image \( x \)

\[ \Psi(x, w) \in \mathbb{R}^m \]

Automatic part discovery

\[ r(x, w_j) = \max_{z_j} w_j \cdot \phi(x, z_j) \]

\( \phi(x, z_j) \in \mathbb{R}^d \)


Automatic part discovery

\[ r(x, w_j) = \max_{z_j} w_j \cdot \phi(x, z_j) \]

Part filters (using HOG [1])

\[ w = (w_1, \ldots, w_m), \quad w_j \in \mathbb{R}^d \]

Input image \( x \)

\[ \Psi(x, w) \in \mathbb{R}^m \]


Representation

Automatic part discovery

\[
r(x, w_j) = \max_{z_j} w_j \cdot \phi(x, z_j)
\]

Part filters (using HOG [1])

\[w = (w_1, \ldots, w_m), \quad w_j \in \mathbb{R}^d\]

Input image \(x\)

\[\Psi(x, w) \in \mathbb{R}^m\]


Automatic part discovery

\[ r(x, w_j) = \max_{z_j} w_j \cdot \phi(x, z_j) \]

\[ \Psi(x, w) = [r(x, w_1); \ldots; r(x, w_m)] \in \mathbb{R}^m \]

\[ \phi(x, z_j) \in \mathbb{R}^d \]

Part filters (using HOG [1])

\[ w = (w_1, \ldots, w_m), \quad w_j \in \mathbb{R}^d \]

Input image \( x \)

\[ \Psi(x, w) \in \mathbb{R}^m \]


Automatic part discovery

\[ r(x, w_j) = \max_{z_j} w_j \cdot \phi(x, z_j) \]

\[ \Psi(x, w) = [r(x, w_1); \ldots; r(x, w_m)] \in \mathbb{R}^m \]

\[ \phi(x, z_j) \in \mathbb{R}^d \]

Part filters (using HOG [1])

\[ w = (w_1, \ldots, w_m), \quad w_j \in \mathbb{R}^d \]


Classification

Automatic part discovery

Linear combination of part responses

\[ f_y(x) = u_y \cdot \Psi(x, w) \]

\[ u_y \in \mathbb{R}^m \quad \text{class-specific part-weights} \]

\[ \Psi(x, w) = [r(x, w_1); \ldots; r(x, w_m)] \in \mathbb{R}^m \quad \text{vector of part responses} \]
Classification

Automatic part discovery

Linear combination of part responses

\[ f_y(x) = u_y \cdot \Psi(x, w) \]

\[ \hat{y}(x; u, v) = \arg\max_{y \in \{1, \ldots, N\}} u_y \cdot \Psi(x, w) \]

where

- \( u_y \in \mathbb{R}^m \) are class-specific part-weights
- \( \Psi(x, w) = [r(x, w_1); \ldots; r(x, w_m)] \in \mathbb{R}^m \) is a vector of part responses
Classification

Automatic part discovery

Linear combination of part responses

\[ f_y(x) = u_y \cdot \Psi(x, w) \]
\[ \hat{y}(x; u, v) = \operatorname{argmax}_{y \in \{1, \ldots, N\}} u_y \cdot \Psi(x, w) \]

\( u_y \in \mathbb{R}^m \)  \quad \text{class-specific part-weights}

\( \Psi(x, w) = [r(x, w_1); \ldots; r(x, w_m)] \in \mathbb{R}^m \)  \quad \text{vector of part responses}

Training objective

\[ O(u, w) = R(u, w) + \sum_{i=1}^{n} L(y_i, \hat{y}(x; u, v)) \]
\[ L(a, b) = 1\{a \neq b\} \quad 0 - 1 \text{ loss} \]
Classification

Automatic part discovery

Linear combination of part responses

\[
f_y(x) = u_y \cdot \Psi(x, w)
\]

\[
\hat{y}(x; u, v) = \arg\max_{y \in \{1, \ldots, N\}} u_y \cdot \Psi(x, w)
\]

\[
u_y \in \mathbb{R}^m \quad \text{class-specific part-weights}
\]

\[
\Psi(x, w) = [r(x, w_1); \ldots; r(x, w_m)] \in \mathbb{R}^m \quad \text{vector of part responses}
\]

Training objective

\[
O(u, w) = R(u, w) + \sum_{i=1}^{n} L(y_i, \hat{y}(x; u, v)), \quad L(a, b) = \mathbf{1}_{\{a \neq b\}} 0 - 1 \text{ loss}
\]

\[
\leq R(u, w) + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + \mathbf{1}_{\{y \neq y_i\}}) - u_{y_i} \cdot \Psi(x_i, w) \quad \text{hinge loss}
\]
Classification

Automatic part discovery

Linear combination of part responses

\[ f_y(x) = u_y \cdot \Psi(x, w) \]

\[ \hat{y}(x; u, v) = \arg\max_{y \in \{1, \ldots, N\}} u_y \cdot \Psi(x, w) \]

\[ u_y \in \mathbb{R}^m \quad \text{class-specific part-weights} \]

\[ \Psi(x, w) = [r(x, w_1); \ldots; r(x, w_m)] \in \mathbb{R}^m \quad \text{vector of part responses} \]

Training objective

\[ O(u, w) = R(u, w) + \sum_{i=1}^{n} L(y_i, \hat{y}(x; u, v)), \quad L(a, b) = 1\{a \neq b\} 0 - 1 \text{ loss} \]

\[ \leq R(u, w) + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \quad \text{hinge loss} \]

\[ R(u, w) = \lambda_u ||u||^2 + \lambda_w ||w||^2 \]
Training algorithm

Automatic part discovery

\[ B(u, w) = R(u, w) + \sum_{i=1}^{n} \max_{y} (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \quad \text{hinge loss} \]
Training algorithm

\[ B(u, w) = R(u, w) + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \quad \text{hinge loss} \]

**Algorithm 1** Joint training of model parameters by optimizing \( B(u, w) \).

1. initialize the part filters \( w = (w_1, \ldots, w_m) \)
2. repeat
3. \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \) \quad \text{fix filters, train classifiers}
4. \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \) \quad \text{fix classifier, train filters}
5. until convergence
6. output \((u, w)\)
Training algorithm

$$B(u, w) = R(u, w) + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + \mathbf{1}_{y \neq y_i}) - u_{y_i} \cdot \Psi(x_i, w) \quad \text{hinge loss}$$

**Algorithm 1** Joint training of model parameters by optimizing $B(u, w)$.

1: initialize the part filters $w = (w_1, \ldots, w_m)$
2: repeat
3: $u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w)$ \hspace{1cm} fix filters, train classifiers
4: $w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w')$ \hspace{1cm} fix classifier, train filters
5: until convergence
6: output $(u, w)$

**BTS, DPD, DMS ...**
- differ in step 1
- proceed only to step 3
- do not re-iterate

**BTS:** Blocks that Shout (Juneja et al. @VGG)
**DPD:** Discriminative Part Detectors (Sun et al. @INRIA)
**DMS:** Discriminative Mode Seeking (Doersch et al. @CMU)
Training algorithm

\[ B(u, w) = R(u, w) + \sum_{i=1}^{n} \max_{y} \left( u_{y} \cdot \Psi(x_{i}, w) + 1\{ y \neq y_{i} \} \right) - u_{y_{i}} \cdot \Psi(x_{i}, w) \]  

hinge loss

Algorithm 1 Joint training of model parameters by optimizing \( B(u, w) \).

1: initialize the part filters \( w = (w_{1}, \ldots, w_{m}) \)
2: repeat
3: \( u := \arg \min_{u' \in \mathbb{R}^{n \times m}} B(u', w) \)  
6: output \((u, w)\)

BTS, DPD, DMS ...
- differ in step 1
- proceed only to step 3
- do not re-iterate

BTS: Blocks that Shout (Juneja et al. @VGG)
DPD: Discriminative Part Detectors (Sun et al. @INRIA)
DMS: Discriminative Mode Seeking (Doersch et al. @CMU)
Discovered parts

Parts discovered for 10-class subset of MIT-indoor dataset [1]

HOG features

Part filters ($w$ matrix)

Image classifiers ($u$ matrix)

Parts discovered for 10-class subset of MIT-indoor dataset [1]

HOG features

Part filters (\(w\) matrix)

Image classifiers (\(u\) matrix)

Discovered parts

Parts discovered for 10-class subset of MIT-indoor dataset [1]

HOG features

Part filters (w matrix)

Image classifiers (u matrix)

Parts discovered for 10-class subset of MIT-indoor dataset [1]

HOG features

Part filters (w matrix)

Image classifiers (u matrix)

Discovered parts

Parts discovered for 10-class subset of MIT-indoor dataset [1]

HOG features

Part filters ($\mathbf{w}$ matrix)

Image classifiers ($\mathbf{u}$ matrix)

I. Random Part Initialization
1. Extract feature from a patch at a random image and location.
2. Whiten the feature.
3. Repeat to construct a pool of candidate parts.

II. Part Selection
1. Train part weights $u$ with L1/L2 regularization.
2. Discard parts that are not used according to $u$.

III. Joint Training
1. Train classifiers $u$ keeping parts $w$ fixed.
2. Train parts $w$ keeping classifiers $u$ fixed.
3. Repeat until convergence.
I. Random Part Initialization
1. Extract feature from a patch at a random image and location.
2. Whiten the feature.
3. Repeat to construct a pool of candidate parts.

II. Part Selection
1. Train part weights $u$ with L1/L2 regularization.
2. Discard parts that are not used according to $u$.

III. Joint Training
1. Train classifiers $u$ keeping parts $w$ fixed.
2. Train parts $w$ keeping classifiers $u$ fixed.
3. Repeat until convergence.

Algorithm 1. Joint training of model parameters
1. initialize the part filters $w = (w_1, \ldots, w_m)$
2. repeat
3. \[ u := \arg\min_{u' \in \mathbb{R}^N} B(u', w) \] \text{ fix filters, train classifiers}
4. \[ w := \arg\min_{w' \in \mathbb{R}^m} B(u, w') \] \text{ fix classifier, train filters}
5. until convergence
6. output $(u, w)$
I. Random Part Initialization
1. Extract feature from a patch at a random image and location.
2. Whiten the feature.
3. Repeat to construct a pool of candidate parts.

II. Part Selection
1. Train part weights \( u \) with L1/L2 regularization.
2. Discard parts that are not used according to \( u \).

III. Joint Training
1. Train classifiers \( u \) keeping parts \( w \) fixed.
2. Train parts \( w \) keeping classifiers \( u \) fixed.
3. Repeat until convergence.
$R(u) = \sum_{j=1}^{m} \rho_j, \quad \rho_j = \sqrt{\sum_{y=1}^{N} u_{y,j}^2}$
Part selection

\[ R(u) = \sum_{j=1}^{m} \rho_j, \quad \rho_j = \sqrt{\sum_{y=1}^{N} u_{y,j}^2} \]

\[ O(u) = \lambda \sum_{j=1}^{m} \rho_j + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + \mathbf{1}_{y \neq y_i}) - u_{y_i} \cdot \Psi(x_i, w) \]
Part selection

Automatic part discovery

\[ R(u) = \sum_{j=1}^{m} \rho_j, \quad \rho_j = \sqrt{\sum_{y=1}^{N} u_{y,j}^2} \]

\[ O(u) = \lambda \sum_{j=1}^{m} \rho_j + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]
Part selection

Automatic part discovery

\[ R(u) = \sum_{j=1}^{m} \rho_j, \quad \rho_j = \sqrt{\sum_{y=1}^{N} u_{y,j}^2} \]

\[ O(u) = \lambda \sum_{j=1}^{m} \rho_j + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]
Automatic part discovery

Results

MIT indoor all classes (HOG)

Results

MIT indoor all classes (HOG)

\[ \Psi(x, w) = \Phi(x, w) \in \mathbb{R}^{5m} \]


Results

Automatic part discovery

MIT indoor all classes (HOG)

\[ \Phi(x, w) + \frac{\Phi(x_{\text{flip}}, w)}{2} \]


Results

Automatic part discovery

MIT indoor 10 class subset (HOG)

MIT indoor all classes (HOG)


Results

Automatic part discovery

MIT indoor 10 class subset (HOG)

<table>
<thead>
<tr>
<th>Performance %</th>
<th># parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>$10^1$</td>
</tr>
<tr>
<td>40</td>
<td>$10^2$</td>
</tr>
<tr>
<td>45</td>
<td>$10^3$</td>
</tr>
</tbody>
</table>

- Green: Random parts (no flipping)
- Blue: Random parts (flip invariant)
- Red: Selected parts (from 10K)
- Black: Jointly trained parts

MIT indoor all classes (HOG)

<table>
<thead>
<tr>
<th>Performance %</th>
<th># parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>$10^1$</td>
</tr>
<tr>
<td>40</td>
<td>$10^2$</td>
</tr>
<tr>
<td>45</td>
<td>$10^3$</td>
</tr>
</tbody>
</table>

- Green: Random parts (no flipping)
- Blue: Random parts (flip invariant)
- Red: Selected parts (from 5K)
- Black: Jointly trained parts


Figure 1: Part filters before (left) and after joint training (right) and top scoring detections for each.
CNN features from the pre-trained Hybrid-Places network [1]

Comparing joint training with different baselines

Visualization

Automatic part discovery

Image classifiers (*u* matrix)
Visualization

Automatic part discovery

Image classifiers (u matrix)
Visualization

Automatic part discovery

Image classifiers ($u$ matrix)
**Visualization**

**Automatic part discovery**

**Image classifiers (u matrix)**

Visualization

Automatic part discovery

Image classifiers (\(u\) matrix)
Visualization

Automatic part discovery

Image classifiers (\(u\) matrix)
Visualization

Image classifiers (\(u\) matrix)

Automatic part discovery
Visualization

Automatic part discovery

Image classifiers (\(u\) matrix)
Faces in buffet, classroom, computer_room
Faces in buffet, classroom, computer_room
Part #46: top scoring patches on test images (multiple patches per image)
Connection to ConvNets

Automatic part discovery

Input image
Connection to ConvNets

Automatic part discovery

Input image

Convolutional layer
Connection to ConvNets

Automatic part discovery

Input image

Convolutional layer

Response map for part $w_j$
Connection to ConvNets

Automatic part discovery

Input image

Convolutional layer

Max-pooling layer

Response map for part $w_j$
Connection to ConvNets: Automatic part discovery

Input image → Convolutional layer → Max-pooling layer → Fully-connected layer

Response map for part $w_j$
Contributions

- Modeling
  - Automatic part discovery

- Learning
  - Generalized Majorization-Minimization (G-MM)
  - Adding stochasticity to G-MM
Non-convex optimization schemes

**Majorization-Minimization / Minorization-Maximization (MM)**

- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Non-convex optimization schemes

Majorization-Minimization / Minorization-Maximization (MM)
- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Non-convex optimization schemes

Majorization-Minimization / Minorization-Maximization (MM)
- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Non-convex optimization schemes

Majorization-Minimization / Minorization-Maximization (MM)
- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Non-convex optimization schemes

Majorization-Minimization / Minorization-Maximization (MM)
- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Non-convex optimization schemes

Majorization-Minimization / Minorization-Maximization (MM)
- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Non-convex optimization schemes

Majorization-Minimization / Minorization-Maximization (MM)
- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Algorithm 1 G-MM optimization.

Input: $w_0$
1: $v_0 \leftarrow F(w_0)$
2: $t \leftarrow 1$
3: repeat
4: $b_t := \text{ConstructBound}(w_{t-1}, v_{t-1})$
5: $w_t := \arg\min_w b_t(w)$
6: $v_t = b_t(w_t)$
7: $t \leftarrow t + 1$
8: until convergence
Output: $w_t$

Non-convex optimization schemes

Majorization-Minimization / Minorization-Maximization (MM)
- Concave Convex Procedure (CCCP)
- Expectation Maximization (EM)

Algorithm 1 G-MM optimization.

Input: \( w_0 \)
1: \( v_0 \leftarrow F(w_0) \)
2: \( t \leftarrow 1 \)
3: repeat
4: \( b_t \leftarrow \text{ConstructBound}(w_{t-1}, v_{t-1}) \)
5: \( w_t \leftarrow \arg\min_{w} b_t(w) \)
6: \( v_t = b_t(w_t) \)
7: \( t \leftarrow t + 1 \)
8: until convergence
Output: \( w_t \)

Generalized Majorization Minimization (G-MM)

\[ B_t = \{ b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w b(w) \geq F(w) \} \]

- \( b \) is a convex bound from a family of bounds \( \mathcal{F} \)
- \( b_{t-1} \) is the previous bound and
- \( w_{t-1} \) is its minimizer
Generalized Majorization Minimization (G-MM)

Let \( (w_0, w_1, \ldots, w_T) \) denote the sequence of solutions

\[ B_t = \{ b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w \ b(w) \geq F(w) \} \]

\( b \) is a convex bound from a family of bounds \( \mathcal{F} \)

\( b_{t-1} \) is the previous bound and \( w_{t-1} \) is its minimizer
Generalized Majorization Minimization (G-MM)

Let $(w_0, w_1, \ldots, w_T)$ denote the sequence of solutions

$\text{MM/G-MM measures progress wrt. objective/bound value:}$

$$B_t = \{ b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w b(w) \geq F(w) \}$$

$b$ is a convex bound from a family of bounds $\mathcal{F}$

$b_{t-1}$ is the previous bound and $w_{t-1}$ is its minimizer
Generalized Majorization Minimization (G-MM)

Let \((w_0, w_1, \ldots, w_T)\) denote the sequence of solutions

**MM/G-MM** measures progress wrt. objective/bound value:

**MM**

\[
F(w_0) \geq F(w_1) \geq \cdots \geq F(w_T)
\]

**G-MM**

\[
b_1(w_0) \geq b_1(w_1) \geq b_2(w_1) \geq \cdots \geq b_T(w_T)
\]

\[B_t = \{b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w \ b(w) \geq F(w)\}\]

\(b\) is a convex bound from a family of bounds \(\mathcal{F}\)

\(b_{t-1}\) is the previous bound and \(w_{t-1}\) is its minimizer.
Generalized Majorization Minimization (G-MM)

Let \((w_0, w_1, \ldots, w_T)\) denote the sequence of solutions

**MM/G-MM** measures progress wrt. objective/bound value:

- **MM**:
  \[ F(w_0) \geq F(w_1) \geq \cdots \geq F(w_T) \]

- **G-MM**:
  \[ b_1(w_0) \geq b_1(w_1) \geq b_2(w_1) \geq \cdots \geq b_T(w_T) = F(w_T) \]

\[ \mathcal{B}_t = \{ b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w b(w) \geq F(w) \} \]

\(b\) is a convex bound from a family of bounds \(\mathcal{F}\)

\(b_{t-1}\) is the previous bound and \(w_{t-1}\) is its minimizer
Generalized Majorization Minimization (G-MM)

Let \((w_0, w_1, \ldots, w_T)\) denote the sequence of solutions.

**MM/G-MM** measures progress wrt. objective/bound value:

- **MM**
  \[
  F(w_0) \geq F(w_1) \geq \cdots \geq F(w_T)
  \]

- **G-MM**
  \[
  F(w_0) = b_1(w_0) \geq b_1(w_1) \geq b_2(w_1) \geq \cdots \geq b_T(w_T) = F(w_T)
  \]

\(B_t = \{b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \ \forall w b(w) \geq F(w)\}\)

\(b\) is a convex bound from a family of bounds \(\mathcal{F}\)

\(b_{t-1}\) is the previous bound and \(w_{t-1}\) is its minimizer
Bound selection

Generalized Majorization Minimization

$$
B_t = \{ b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w \ b(w) \geq F(w) \} $$

Bound selection scenarios:
Bound selection \( B_t = \{ b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w b(w) \geq F(w) \} \)

Bound selection scenarios:

1. Stochastic: uniformly at random from \( B_t \) i.e. \( b_t \sim \mathcal{U} B_t \)
Bound selection scenarios:

1. Stochastic: uniformly at random from $\mathcal{B}_t$ i.e. $b_t \sim \mathcal{B}_t$ 
2. Deterministic: one that maximizes a "bias" function $Q$

$$ Q : \mathcal{B}_t \times \mathbb{R}^d \rightarrow \mathbb{R} $$

$$ b_t = \arg \max_{b \in \mathcal{B}_t} Q(b, w_{t-1}) $$
Bound selection scenarios:

1. Stochastic: uniformly at random from $B_t$, i.e. $b_t \sim B_t$

2. Deterministic: one that maximizes a “bias” function $Q$

$$Q : B_t \times \mathbb{R}^d \rightarrow \mathbb{R} \quad \quad b_t = \arg\max_{b \in B_t} Q(b, w_{t-1})$$

Define $Q$ to select bounds that have a desired property:
- balanced clusters
- maximum progress (MM)
Bound selection scenarios:

1. Stochastic: uniformly at random from $\mathcal{B}_t$ i.e. $b_t \sim \mathcal{B}_t$
2. Deterministic: one that maximizes a “bias” function $Q$

Define $Q$ to select bounds that have a desired property:
- balanced clusters
- maximum progress (MM)

\[ Q_{MM}(b, w_{t-1}) = b_{t-1}(w_{t-1}) - b(w_{t-1}) \]
Bound selection

\[ \mathcal{B}_t = \{ b \in \mathcal{F} \mid b(w_{t-1}) \leq b_{t-1}(w_{t-1}), \forall w \; b(w) \geq F(w) \} \]

Bound selection scenarios:

1. Stochastic: uniformly at random from \( \mathcal{B}_t \) i.e. \( b_t \sim \mathcal{B}_t \)

2. Deterministic: one that maximizes a “bias” function \( Q \)

\[ Q : \mathcal{B}_t \times \mathbb{R}^d \rightarrow \mathbb{R} \]

\[ b_t = \arg\max_{b \in \mathcal{B}_t} Q(b, w_{t-1}) \]

Define \( Q \) to select bounds that have a desired property:
- balanced clusters
- maximum progress (MM)

\[ Q_{MM}(b, w_{t-1}) = b_{t-1}(w_{t-1}) - b(w_{t-1}) \]
Bound selection scenarios:

1. **Stochastic**: uniformly at random from $\mathcal{B}_t$ i.e. $b_t \sim \mathcal{B}_t$

2. **Deterministic**: one that maximizes a “bias” function $Q$

   $$Q : \mathcal{B}_t \times \mathbb{R}^d \rightarrow \mathbb{R}$$

   $$b_t = \underset{b \in \mathcal{B}_t}{\text{argmax}} Q(b, w_{t-1})$$

Define $Q$ to select bounds that have a desired property:

- balanced clusters
- maximum progress (MM)

$$Q_{MM}(b, w_{t-1}) = b_{t-1}(w_{t-1}) - b(w_{t-1})$$
Results (k-means) Generalized Majorization Minimization

\[ O(w) = \sum_{i=1}^{n} \min_{z_i=1}^{k} \| x_i - \mu_{z_i} \|^2, \quad w = (\mu_1; \ldots; \mu_k) \]
Results (k-means) Generalized Majorization Minimization

\[ O(w) = \sum_{i=1}^{n} \min_{z_i=1} \| x_i - \mu_{z_i} \|^2, \quad w = (\mu_1; \ldots; \mu_k) \]

\[ b_t(w) = \sum_{i=1}^{n} \| x_i - \mu_{z_i^{(t)}} \|^2 \]

\[ z_i^{(t)} = \arg\min_j \| x_i - \mu_j^{(t-1)} \|^2 \]
\[ O(w) = \sum_{i=1}^{n} \min_{z_i=1}^{k} \| x_i - \mu_{z_i} \|^2, \quad w = (\mu_1; \ldots; \mu_k) \]

\[ \mathcal{F} = \left\{ b(w) = \sum_{i=1}^{n} \| x_i - \mu_{z_i} \|^2 \mid \forall i, z_i \in \{1, \ldots, k\}^n \right\} \]

\[ b_t(w) = \sum_{i=1}^{n} \| x_i - \mu_{z_i(t)} \|^2 \]

\[ z_i^{(t)} = \arg\min_j \| x_i - \mu_j^{(t-1)} \|^2 \]
Results (k-means) Generalized Majorization Minimization

\[ O(w) = \sum_{i=1}^{n} \min_{z_i=1} \left\| x_i - \mu_{z_i} \right\|^2, \quad w = (\mu_1; \ldots; \mu_k) \]

\[ \mathcal{F} = \left\{ b(w) = \sum_{i=1}^{n} \left\| x_i - \mu_{z_i} \right\|^2 \mid \forall i, z_i \in \{1, \ldots, k\}^n \right\} \]

**G-MM**

\[ b_t(w) = \sum_{i=1}^{n} \left\| x_i - \mu_{z_i}^{(t)} \right\|^2 \]

\[ z_i^{(t)} = \arg\min_j \left\| x_i - \mu_j^{(t-1)} \right\|^2 \]

---

MM (k-means algorithm)

G-MM

Both experiments use the **same** initialization (random partition)
Contributions

- Modeling
  - Automatic part discovery

- Learning
  - Generalized Majorization Minimization (G-MM)
  - Adding stochasticity to G-MM
G-MM in expectation
G-MM in expectation

G-MM greedily makes progress
G-MM in expectation

G-MM greedily makes progress
We extend G-MM to make progress *in expectation*
G-MM in expectation

G-MM greedily makes progress

We extend G-MM to make progress in expectation

\( P : \) probability distributions on \( \mathcal{F} \)

\( f : \) r.v. on \( \mathcal{F} \) with distribution \( P \)
G-MM in expectation

G-MM greedily makes progress

We extend G-MM to make progress in expectation

\[ P : \text{probability distributions on } \mathcal{F} \]
\[ f : \text{r.v. on } \mathcal{F} \text{ with distribution } P \]

\[ \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \]
G-MM in expectation

G-MM greedily makes progress

We extend G-MM to make progress in expectation

\[ P : \text{probability distributions on } \mathcal{F} \]
\[ f : \text{r.v. on } \mathcal{F} \text{ with distribution } P \]
\[ \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \]
G-MM in expectation

G-MM greedily makes progress

We extend G-MM to make progress *in expectation*

- \( P \): probability distributions on \( \mathcal{F} \)
- \( f \): r.v. on \( \mathcal{F} \) with distribution \( P \)

\[
\mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1})
\]

Maximum Entropy Principle:

the distribution with the **highest entropy** makes the **fewest assumptions** about the data beyond the observed constraints
G-MM in expectation

G-MM greedily makes progress

We extend G-MM to make progress in expectation

\[ P : \text{probability distributions on } \mathcal{F} \]
\[ f : \text{r.v. on } \mathcal{F} \text{ with distribution } P \]
\[ \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \]

Maximum Entropy Principle:

the distribution with the **highest entropy** makes the **fewest assumptions** about the data beyond the observed constraints

\[ P^* = \arg\max_{P \in \mathcal{P}(\mathcal{F})} H(P) \]
G-MM in expectation

G-MM greedily makes progress

We extend G-MM to make progress in expectation

\[ P : \text{probability distributions on } \mathcal{F} \]
\[ f : \text{r.v. on } \mathcal{F} \text{ with distribution } P \]
\[ \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \]

Maximum Entropy Principle:

the distribution with the \textbf{highest entropy} makes the \textbf{fewest assumptions} about the data beyond the observed constraints

\[
P^* = \arg\max_{P \in \mathcal{P}(\mathcal{F})} H(P)
\]
\[ \text{s.t. } \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \]
Goal is to find

$$P^* = \arg\max_{P \in \mathcal{P}(\mathcal{F})} H(P)$$

s.t. $\mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1})$
Goal is to find

Find the right $\lambda$

\[
P^* = \arg\max_{P \in \mathcal{P}(\mathcal{F})} H(P)
\]

\[
\text{s.t. } \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1})
\]

\[
P_\lambda(f) \propto e^{\lambda f(w_{t-1})}
\]
Goal is to find

\[ P^* = \arg\max_{P \in \mathcal{P}(F)} H(P) \]

\[ \text{s.t. } \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \]

Find the right \( \lambda \)

\[ P_\lambda(f) \propto e^{\lambda f(w_{t-1})} \]

Claim 1: \( H(P_\lambda) \) increases monotonically as \( |\lambda| \) decreases
Goal is to find

\[ P^* = \arg\max_{P \in \mathcal{P}(\mathcal{F})} H(P) \]

s.t. \( \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \)

Find the right \( \lambda \)

\[ P_{\lambda}(f) \propto e^{\lambda f(w_{t-1})} \]

Claim 1: \( H(P_{\lambda}) \) increases monotonically as \( |\lambda| \) decreases

Claim 2: \( \mathbb{E}_{f \sim P_{\lambda}}[f(w_{t-1})] \) decreases monotonically as \( \lambda \) decreases
Goal is to find

\[ P^* = \arg\max_{P \in \mathcal{P}(\mathcal{F})} H(P) \]

subject to \( \mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1}) \)

Find the right \( \lambda \)

\[ P_{\lambda}(f) \propto e^{\lambda f(w_{t-1})} \]

Claim 1: \( H(P_{\lambda}) \) increases monotonically as \( |\lambda| \) decreases

Claim 2: \( \mathbb{E}_{f \sim P_{\lambda}}[f(w_{t-1})] \) decreases monotonically as \( \lambda \) decreases

Corollary: binary search on

\[ \lambda \in (-\infty, 0] \]
Goal is to find $P^* = \arg\max_{P \in \mathcal{P}(\mathcal{F})} H(P)$ s.t. $\mathbb{E}_{f \sim P}[f(w_{t-1})] < b_{t-1}(w_{t-1})$.

Find the right $\lambda$

$$P_\lambda(f) \propto e^{\lambda f(w_{t-1})}$$

Claim 1: $H(P_\lambda)$ increases monotonically as $|\lambda|$ decreases.

Claim 2: $\mathbb{E}_{f \sim P_\lambda}[f(w_{t-1})]$ decreases monotonically as $\lambda$ decreases.

Corollary: binary search on $\lambda$.

To compute $P_\lambda$ in practice:

- If manageable, compute the expectation.
- Otherwise, estimate it by sampling from $P_\lambda$. 
Automatic part discovery for image classification:
all steps are driven by classification loss
avoids using hand-engineered heuristics
joint training algorithm
improves the quality of the discovered parts

Generalized MM (G-MM):
relaxes the “touching” constraint in MM
is more flexible and less sensitive to initialization than MM
can be extended to makes progress *in expectation*
making it likely to avoid getting stuck in local minima
Thank you!
\[ B(u, w) = \lambda_u \|u\|^2 + \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1_{y \neq y_i}) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1** Joint training

1. initialize the part filters \( w = (w_1, \ldots, w_m) \)
2. repeat
3. \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4. \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5. until convergence
6. output \((u, w)\)
Learning Automatic part discovery

\[ B(u, w) = \lambda_u \|u\|^2 + \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_{y} (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1** Joint training

1: initialize the part filters \( w = (w_1, \ldots, w_m) \)
2: repeat
3: \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4: \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5: until convergence
6: output \((u, w)\)

part index

- bookstore
- bowling
- closet
- corridor
- laundromat
- library
- nursery
- shoeshop
- staircase
- winecellar
Learning

Automatic part discovery

\[ B(u, w) = \lambda_u \|u\|^2 + \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_y \left( u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\} \right) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1 Joint training**

1:initialize the part filters \( w = (w_1, \ldots, w_m) \)
2:repeat
3: \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4: \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5:until convergence
6:output \((u, w)\)

Optimize \[ \lambda_u \|u\|^2 + \sum_{i=1}^{n} \max_y \left( u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\} \right) - u_{y_i} \cdot \Psi(x_i, w) \]
Learning Automatic part discovery

\[ B(u, w) = \lambda_u ||u||^2 + \lambda_w ||w||^2 + \sum_{i=1}^{n} \max_{y} (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1 Joint training**

1: initialize the part filters \( w = (w_1, \ldots, w_m) \)
2: repeat
3: \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4: \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5: until convergence
6: output \((u, w)\)

Optimize \( \lambda_u ||u||^2 + \sum_{i=1}^{n} \max_{y} (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \)

Multi-class SVM
Learning Automatic part discovery

$$B(u, w) = \lambda_u \|u\|^2 + \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_y \left( u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\} \right) - u_{y_i} \cdot \Psi(x_i, w)$$

**Algorithm 1** Joint training

1. initialize the part filters \( w = (w_1, \ldots, w_m) \)
2. repeat
3. \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4. \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5. until convergence
6. output \((u, w)\)

Optimize

$$\lambda_u \|u\|^2 + \sum_{i=1}^{n} \max_y \left( u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\} \right) - u_{y_i} \cdot \Psi(x_i, w)$$

Objective function is convex and the global minimum can be found efficiently!
\[ B(u, w) = \lambda_u \| u \|^2 + \lambda_w \| w \|^2 + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1**  Joint training

1. initialize the part filters \( w = (w_1, \ldots, w_m) \)
2. repeat
3. \( u := \text{arg min}_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4. \( w := \text{arg min}_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5. until convergence
6. output \((u, w)\)
\[ B(u, w) = \lambda_u \|u\|^2 + \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1 Joint training**

1: initialize the part filters \( w = (w_1, \ldots, w_m) \)
2: repeat
3: \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4: \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5: until convergence
6: output \((u, w)\)

Optimize \( \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_y \left( \sum_{j=1}^{m} u_{y,j} \max_z w_j \cdot \phi(x_i, z) + 1\{y \neq y_i\} \right) - \sum_{j=1}^{m} u_{y_i,j} \max_z w_j \cdot \phi(x_i, z) \)
Optimize \( \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_{y} \left( \sum_{j=1}^{m} u_{y,j} \max_{z} w_j \cdot \phi(x_i, z) + 1 \{ y \neq y_i \} \right) - \sum_{j=1}^{m} u_{y_i,j} \max_{z} w_j \cdot \phi(x_i, z) \)

Objective function is non-convex 😞
Learning

\[ B(u, w) = \lambda_u \|u\|^2 + \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_y \left( u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\} \right) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1 Joint training**

1: initialize the part filters \( w = (w_1, \ldots, w_m) \)
2: repeat
3: \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4: \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5: until convergence
6: output \((u, w)\)

Optimize

\[ \lambda_w \|w\|^2 + \sum_{i=1}^{n} \max_y \left( \sum_{j=1}^{m} u_{y,j} \max_z w_j \cdot \phi(x_i, z) + 1\{y \neq y_i\} \right) - \sum_{j=1}^{m} u_{y_i,j} \max_z w_j \cdot \phi(x_i, z) \]

Objectives function is non-convex 😞
Learning

Automatic part discovery

\[ B(u, w) = \lambda_u ||u||^2 + \lambda_w ||w||^2 + \sum_{i=1}^{n} \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1** Joint training

1. initialize the part filters \( w = (w_1, \ldots, w_m) \)
2. repeat
3. \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4. \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5. until convergence
6. output \((u, w)\)

Optimize \[ \lambda_w ||w||^2 + \sum_{i=1}^{n} \max_y \left( \sum_{j=1}^{m} u_{y,j} \max_z w_j \cdot \phi(x_i, z) + 1\{y \neq y_i\} \right) - \sum_{j=1}^{m} u_{y_i,j} \max_z w_j \cdot \phi(x_i, z) \]

Objective function is non-convex 😞
Different from well-studied optimization problems such as Latent-SVM 😞
Learning

Automatic part discovery

\[ B(u, w) = \lambda_u ||u||^2 + \lambda_w ||w||^2 + \sum_{i=1}^{n} \max_y \left( u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\} \right) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1 Joint training**

1: initialize the part filters \( w = (w_1, \ldots, w_m) \)
2: repeat
3: \( u := \arg \min_{u' \in \mathbb{R}^{N \times m}} B(u', w) \)
4: \( w := \arg \min_{w' \in \mathbb{R}^{m \times d}} B(u, w') \)
5: until convergence
6: output \((u, w)\)

Optimize

\[
\lambda_w ||w||^2 + \sum_{i=1}^{n} \max_y \left( \sum_{j=1}^{m} u_{y,j} \max_z w_j \cdot \phi(x_i, z) + 1\{y \neq y_i\} \right) - \sum_{j=1}^{m} u_{y_i,j} \max_z w_j \cdot \phi(x_i, z)
\]

\[
\lambda_w ||w||^2 + \sum_{i=1}^{n} \max_{y \neq y_i} \left\{ 0, \max_{y \neq y_i} \left( \sum_{j=1}^{m} (u_{y_i,j} - u_{y,j}) \max_z w_j \cdot \phi(x_i, z) + 1 \right) \right\}
\]

Objective function is non-convex 😞

Different from well-studied optimization problems such as Latent-SVM 😢
Learning

Automatic part discovery

\[ B(u, w) = \lambda_u \|u\|^2 + \lambda_w \|w\|^2 + \sum_{i=1}^n \max_y (u_y \cdot \Psi(x_i, w) + 1\{y \neq y_i\}) - u_{y_i} \cdot \Psi(x_i, w) \]

**Algorithm 1 Joint training**

1. initialize the part filters \( w = (w_1, \ldots, w_m) \)
2. repeat
3. \( u := \arg \min_{u' \in \mathbb{R}^N \times m} B(u', w) \)
4. \( w := \arg \min_{w' \in \mathbb{R}^m \times d} B(u, w') \)
5. until convergence
6. output \((u, w)\)

Optimize

\[ \lambda_w \|w\|^2 + \sum_{i=1}^n \max_y \left( \sum_{j=1}^m u_{y,j} \max_z w_j \cdot \phi(x_i, z) + 1\{y \neq y_i\} \right) - \sum_{j=1}^m u_{y_i,j} \max_z w_j \cdot \phi(x_i, z) \]

Objective function is non-convex 😞

Different from well-studied optimization problems such as Latent-SVM 😞

But we can fix latent variables when \( u_{y,j} - u_{y_i,j} < 0 \) 😊
Weakly supervised object detection on Mammal dataset

<table>
<thead>
<tr>
<th>Opt. Method</th>
<th>center</th>
<th>top-left</th>
<th>random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>objective</td>
<td>test error</td>
<td>objective</td>
</tr>
<tr>
<td>CCCC</td>
<td>1.21 ± 0.03</td>
<td>22.9 ± 9.7</td>
<td>1.35 ± 0.03</td>
</tr>
<tr>
<td>G-MM random</td>
<td>0.79 ± 0.03</td>
<td>17.5 ± 3.9</td>
<td>0.91 ± 0.02</td>
</tr>
<tr>
<td>G-MM biased</td>
<td>0.64 ± 0.02</td>
<td>16.8 ± 3.2</td>
<td>0.70 ± 0.02</td>
</tr>
</tbody>
</table>
Less sensitive to initialization

**forgy**: initialize cluster centers to random examples

**random partition**: initialize each data point to a random cluster center

More flexible: e.g. Multi-Fold Multiple Instance Learning
Latent variable models (LVM) only model evidence
Latent variable models (LVM) only model evidence

_Generalized LVM (GLVM)_ also model counterevidence
Generalized Latent Variable Models

Latent variable models (LVM) only model evidence

**Generalized LVM (GLVM)** also model counterevidence

Game theory analogy:
- an LVM player tries to maximize its score
- a GLVM player tries to maximize its score while minimizing its opponent’s score
Latent variable models (LVM) only model evidence

**Generalized LVM (GLVM)** also model counterevidence

Game theory analogy:
- an LVM player tries to maximize its score
- a GLVM player tries to maximize its score while minimizing its opponent’s score

DPMs lack negative parts: a **saddle** can be a negative part for a **cow** detector.
Latent variable models (LVM) only model evidence.

**Generalized LVM (GLVM)** also model counterevidence.

Game theory analogy:
- an LVM player tries to maximize its score.
- a GLVM player tries to maximize its score while minimizing its opponent’s score.

DPMs lack negative parts: a **saddle** can be a negative part for a **cow** detector.

\[
\begin{align*}
\max_z w_j \cdot \phi(x_i, z) & \quad \text{positive part} \\
\max_z w_j \cdot \phi(x_i, z) & \quad \text{negative part}
\end{align*}
\]
Latent variable models (LVM) only model evidence

**Generalized LVM (GLVM)** also model counterevidence

Game theory analogy:
- an LVM player tries to maximize its score
- a GLVM player tries to maximize its score while minimizing its opponent’s score

DPMs lack negative parts: a **saddle** can be a negative part for a **cow** detector.

\[
\begin{align*}
\max_z w_j \cdot \phi(x_i, z) & \quad \text{positive part} \\
- \max_z w_j \cdot \phi(x_i, z) & \quad \text{negative part}
\end{align*}
\]

this is different from negating part filters
Generalized Latent Variable Models

Latent variable models (LVM) only model evidence

**Generalized LVM (GLVM)** also model counterevidence

Game theory analogy:
- an LVM player tries to maximize its score
- a GLVM player tries to maximize its score while minimizing its opponent’s score

DPMs lack negative parts: a **saddle** can be a negative part for a **cow** detector.

\[
\max_z w_j \cdot \phi(x_i, z) \quad \text{vs} \quad - \max_z w_j \cdot \phi(x_i, z) = \min_z (-w_j) \cdot \phi(x_i, z)
\]

 positive part  negative part

this is different from negating part filters
\[ f_w(x) = \max_{a \in Z} w \cdot \phi(x, a) \]
Formulation: GLVM

LVM

\[ f_w(x) = \max_{a \in Z} w \cdot \phi(x, a) \]

GLVM

\[ f_w(x) = \max_{a \in Z^+} w \cdot \phi(x, a) - \max_{b \in Z^-} w \cdot \phi(x, b) \]

\[ = \max_{a \in Z^+} \min_{b \in Z^-} w \cdot \phi(x, a) - w \cdot \phi(x, b) \]

\[ = \max_{a \in Z^+} \min_{b \in Z^-} w \cdot \phi(x, a, b) \]
Formulation: GLVM

$$f_w(x) = \max_{a \in Z} w \cdot \phi(x, a)$$

GLVM

$$f_w(x) = \max_{a \in Z^+} w \cdot \phi(x, a) - \max_{b \in Z^-} w \cdot \phi(x, b)$$

$$= \max_{a \in Z^+} \min_{b \in Z^-} w \cdot \phi(x, a) - w \cdot \phi(x, b)$$

$$= \max_{a \in Z^+} \min_{b \in Z^-} w \cdot \phi(x, a, b)$$
**Formulation: GLVM**

\[ f_w(x) = \max_{a \in Z} w \cdot \phi(x, a) \]

**Training objective**

\[ O(w) = \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{n} \max \left\{ 0, 1 - y_i f_w(x_i) \right\} \]

Training objective

\[ O(w) = \frac{\lambda}{2} \| w \|^2 + \sum_{i=1}^{n} \max \left\{ 0, \ 1 - y_i f_w(x_i) \right\} \]

Convex bound

\[ b_t(w) = \frac{\lambda}{2} \| w \|^2 + \sum_{i=1}^{n} \max \left\{ 0, \ 1 + \tilde{f}_w(x_i) \right\} + \sum_{i=1}^{n} \max \left\{ 0, \ 1 - \tilde{f}_w(x_i) \right\} \]

\[
 b^{(t)}(x_i, a) = \arg\min_{b_1, \ldots, b_K} w_t \cdot \phi \left( x_i, (a_k, b_k)_{k=1}^{K} \right)
\]

\[
 a^{(t)}(x_i, b) = \arg\min_{a_1, \ldots, a_K} w_t \cdot \phi \left( x_i, (a_k, b_k)_{k=1}^{K} \right)
\]

\[
 \tilde{f}_t(x_i) = \max_{a_1} \max_{a_2} \ldots \max_{a_K} w \cdot \phi \left( x_i, (a_k, b_k^{(t-1)}(x_i, a))_{k=1}^{K} \right)
\]

\[
 \hat{f}_t(x_i) = \min_{b_1} \min_{b_2} \ldots \min_{b_K} w \cdot \phi \left( x_i, (a^{(t-1)}(x_i, b), b_k)_{k=1}^{K} \right)
\]

Algorithm 5 Training GLVM classifiers.

**Input:** \( D = \{(x_i, y_i)\}_{i=1}^{n}, w_0 \)

1: \( t \leftarrow 0 \)
2: **repeat**
3: \( t \leftarrow t + 1 \)
4: Construct \( \tilde{f}_t(x) \)
5: Construct \( \hat{f}_t(x) \)
6: Construct \( b_t(w) \)
7: \( w_t \leftarrow \arg\min_w b_t(w) \)
8: **until** \( w_t \) does not change

**Output:** \( w_t \)
GLVMs can extend popular methods:

– Deformable Part Models
– And-Or trees: Nor node
– Latent Hough Transform

Images taken from [1]