Visual Recognition: Examples of Graphical Models

Raquel Urtasun

TTI Chicago

March 6, 2012
Graphical models

- Applications
- Representation
- Inference
  - message passing (LP relaxations)
  - graph cuts
- Learning
Learning in graphical models
The MAP problem was defined as

$$\max_{y_1, \ldots, y_n} \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

Learn parameters $w$ for more accurate prediction

$$\max_{y_1, \ldots, y_n} \sum_i w_i \phi_i(y_i) + \sum_\alpha w_\alpha \phi_\alpha(y_\alpha)$$
Loss functions

- Regularized loss minimization: Given input pairs \((x, y) \in S\), minimize

\[
\sum_{(x,y) \in S} \hat{\ell}(w, x, y) + \frac{C}{p} \|w\|_p^p
\]

- Different learning frameworks depending on the surrogate loss \(\hat{\ell}(w, x, y)\)
  - Hinge for Structural SVMs [Tsochantaridis et al. 05, Taskar et al. 04]
  - log-loss for Conditional Random Fields [Lafferty et al. 01]
- Unified by [Hazan and Urtasun, 10]
Regularized loss minimization: Given input pairs \((x, y) \in S\), minimize

\[
\sum_{(x, y) \in S} \hat{\ell}(w, x, y) + \frac{C}{p} \|w\|_p^p,
\]

Different learning frameworks depending on the surrogate loss \(\hat{\ell}(w, x, y)\)
- Hinge for Structural SVMs [Tsochantaridis et al. 05, Taskar et al. 04]
- log-loss for Conditional Random Fields [Lafferty et al. 01]

Unified by [Hazan and Urtasun, 10]
Recall SVM

In SVMs we minimize the following program

\[
\min_w \frac{1}{2} \|w\|^2 + \sum_i \xi_i
\]

subject to \( y_i(b + w^T x_i) - 1 + \xi_i \geq 0, \quad \forall i = 1, \ldots, N. \)

with \( y_i \in \{-1, 1\} \) binary.

We need to extend this to reason about more complex structures, not just binary variables.
We want to construct a function

\[ f(x, y) = \arg \max_{y \in Y} \mathbf{w}^T \phi(x, y) \]

which is parameterized in terms of \( \mathbf{w} \), the parameters to learn.

We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

\( \Delta(y, y') = 0 \) if \( y = y' \).
Structural SVM [Tsochantaridis et al., 05]

- We want to construct a function

\[ f(x, y) = \arg \max_{y \in Y} w^T \phi(x, y) \]

which is parameterized in terms of \( w \), the parameters to learn.

- We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

- This is the expected loss under the empirical distribution induced
We want to construct a function

\[ f(x, y) = \arg \max_{y \in Y} w^T \phi(x, y) \]

which is parameterized in terms of \( w \), the parameters to learn.

We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

This is the expected loss under the empirical distribution induced

\( \Delta(y_i, f(x_i, w)) \) is the "task loss" which depends on the application

- segmentation: per pixel segmentation error
- detection: intersection over the union
Structural SVM [Tsochantaridis et al., 05]

- We want to construct a function

\[ f(x, y) = \arg \max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x, y) \]

which is parameterized in terms of \( \mathbf{w} \), the parameters to learn.

- We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

- This is the expected loss under the empirical distribution induced

- \( \Delta(y_i, f(x_i, w)) \) is the "task loss" which depends on the application
  - segmentation: per pixel segmentation error
  - detection: intersection over the union

- Typically, \( \Delta(y, y') = 0 \) if \( y = y' \)
We want to construct a function

\[ f(x, y) = \arg \max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x, y) \]

which is parameterized in terms of \( \mathbf{w} \), the parameters to learn.

We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

This is the expected loss under the empirical distribution induced.

\( \Delta(y_i, f(x_i, w)) \) is the "task loss" which depends on the application

- segmentation: per pixel segmentation error
- detection: intersection over the union

Typically, \( \Delta(y, y') = 0 \) if \( y = y' \)
Separable case

- We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

- We will have 0 train error if we satisfy

\[ \max_{y \in \mathcal{Y} \setminus y_i} \{w^T \phi(x_i, y)\} \leq w^T \phi(x_i, y_i) \]

since \( \Delta(y_i, y_i) = 0 \) and \( \Delta(y_i, y) > 0 \), \( \forall y \in \mathcal{Y} \setminus y_i \).
Separable case

- We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

- We will have 0 train error if we satisfy

\[ \max_{y \in \mathcal{Y} \setminus y_i} \{ w^T \phi(x_i, y) \} \leq w^T \phi(x_i, y_i) \]

since \( \Delta(y_i, y_i) = 0 \) and \( \Delta(y_i, y) > 0, \forall y \in \mathcal{Y} \setminus y_i \).

- This can be replaced by \(|\mathcal{Y}| - 1\) inequalities

\[ \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i : \quad w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 0 \]
Separable case

- We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

- We will have 0 train error if we satisfy

\[ \max_{y \in \mathcal{Y} \setminus y_i} \{ w^T \phi(x_i, y) \} \leq w^T \phi(x_i, y_i) \]

since \( \Delta(y_i, y_i) = 0 \) and \( \Delta(y_i, y) > 0, \forall y \in \mathcal{Y} \setminus y_i. \)

- This can be replaced by \(|\mathcal{Y}| - 1\) inequalities

\[ \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i : \quad w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 0 \]

- What’s the problem of this?
Separable case

- We will like to minimize the empirical risk

\[ R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w)) \]

- We will have 0 train error if we satisfy

\[ \max_{y \in \mathcal{Y} \setminus y_i} \{ w^T \phi(x_i, y) \} \leq w^T \phi(x_i, y_i) \]

since \( \Delta(y_i, y_i) = 0 \) and \( \Delta(y_i, y) > 0, \forall y \in \mathcal{Y} \setminus y_i. \)

- This can be replaced by \( |\mathcal{Y}| - 1 \) inequalities

\[ \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i : \ w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 0 \]

- What’s the problem of this?
Separable case

- Satisfying the inequalities might have more than one solution.
- Select the $w$ with the maximum margin.

We can thus form the following optimization problem

$$\min_w \|w\|_2$$

subject to

$$w^T \phi(x_i, y_i) - w^T \phi(x_i, y_i) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in Y \setminus y_i$$

This is a quadratic program, so it's convex but it involves exponentially many constraints!
Satisfying the inequalities might have more than one solution.

Select the $w$ with the maximum margin.

We can thus form the following optimization problem

$$\min_w \frac{1}{2} \|w\|^2$$

subject to $w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$
Separable case

- Satisfying the inequalities might have more than one solution.
- Select the $\mathbf{w}$ with the maximum margin.
- We can thus form the following optimization problem

$$
\min_{\mathbf{w}} \quad \frac{1}{2} \| \mathbf{w} \|^2 \\
\text{subject to} \quad \mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i
$$

- This is a quadratic program, so it’s convex
Separable case

- Satisfying the inequalities might have more than one solution.
- Select the $\mathbf{w}$ with the maximum margin.
- We can thus form the following optimization problem

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to $\mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$

- This is a quadratic program, so it’s convex
- But it involves exponentially many constraints!
Satisfying the inequalities might have more than one solution.

Select the $\mathbf{w}$ with the maximum margin.

We can thus form the following optimization problem

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to $\mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$

This is a quadratic program, so it’s convex

But it involves exponentially many constraints!
Non-separable case

Multiple formulations

- Multi-class classification [Crammer & Singer, 03]
- Slack re-scaling [Tsochantaridis et al. 05]
- Margin re-scaling [Taskar et al. 04]

Let’s look at them in more details
Multi-class classification [Crammer & Singer, 03]

- Enforce a large margin and do a batch convex optimization
- The minimization program is then

\[
\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \\
\text{s.t. } w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 1 - \xi_i \quad \forall i \in \{1, \cdots, n\}, \forall y \neq y_i
\]

- Can also be written in terms of kernels
Frame structured prediction as a multiclass problem to predict a single element of $Y$ and pay a penalty for mistakes.

Not all errors are created equally, e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes.
- Frame structured prediction as a multiclass problem to predict a single element of $Y$ and pay a penalty for mistakes

- Not all errors are created equally, e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes

- Pay a loss proportional to the difference between true and predicted error (task dependent)

$$\Delta(y_i, y)$$

[Source: M. Blaschko]
Structured Output SVMs

- Frame structured prediction as a multiclass problem to predict a single element of $Y$ and pay a penalty for mistakes
- Not all errors are created equally, e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes
- Pay a loss proportional to the difference between true and predicted error (task dependent)

$$\Delta(y_i, y)$$

[Source: M. Blaschko]
Slack re-scaling

- Re-scale the slack variables according to the loss incurred in each of the linear constraints
- Violating a margin constraint involving a \( y \neq y_i \) with high loss \( \Delta(y_i, y) \) should be penalized more than a violation involving an output value with smaller loss

The minimization program is then

\[
\min_{w} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 1 - \xi_i \Delta(y_i, y) \quad \forall i \in \{1, \ldots, n\}, \forall y \notin y_i
\]

The justification is that \( \frac{1}{n} \sum_{i=1}^{n} \xi_i \) is an upper-bound on the empirical risk.

Easy to prove
Re-scale the slack variables according to the loss incurred in each of the linear constraints.

Violating a margin constraint involving a $y \neq y_i$ with high loss $\Delta(y_i, y)$ should be penalized more than a violation involving an output value with smaller loss.

The minimization program is then

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i$$

s.t. $w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 1 - \frac{\xi_i}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$
Slack re-scaling

- Re-scale the slack variables according to the loss incurred in each of the linear constraints
- Violating a margin constraint involving a \( y \neq y_i \) with high loss \( \Delta(y_i, y) \) should be penalized more than a violation involving an output value with smaller loss
- The minimization program is then

\[
\min_{\mathbf{w}} \quad \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad \mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, y) \geq 1 - \frac{\xi_i}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i
\]

- The justification is that \( \frac{1}{n} \sum_{i=1}^{n} \xi_i \) is an upper-bound on the empirical risk.
- Easy to proof
Slack re-scaling

- Re-scale the slack variables according to the loss incurred in each of the linear constraints.
- Violating a margin constraint involving a $y \neq y_i$ with high loss $\Delta(y_i, y)$ should be penalized more than a violation involving an output value with smaller loss.
- The minimization program is then

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$

s.t. $w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 1 - \frac{\xi_i}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$

- The justification is that $\frac{1}{n} \sum_{i=1}^{n} \xi_i$ is an upper-bound on the empirical risk.
- Easy to proof.
In this case the minimization problem is formulated as

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$

s.t. \( w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq \Delta(y_i, y) - \xi_i \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i \)

The justification is that \( \frac{1}{n} \sum_{i=1}^{n} \xi_i \) is an upper-bound on the empirical risk.

Also easy to proof.
Algorithm 1 Algorithm for solving SVM_0 and the loss re-scaling formulations SVM_1^± and SVM_2^±.

1: **Input:** \((x_1, y_1), \ldots, (x_n, y_n), C, \varepsilon\)
2: \(S_i \leftarrow \emptyset\) for all \(i = 1, \ldots, n\)
3: **repeat**
4: **for** \(i = 1, \ldots, n\) **do**
5: /* prepare cost function for optimization */
   set up cost function
   \[
   H(y) \equiv \begin{cases}
   1 - \langle \delta \Psi_i(y), w \rangle & \text{(SVM}_0) \\
   (1 - \langle \delta \Psi_i(y), w \rangle) \triangle(y_i, y) & \text{(SVM}_1^s) \\
   \triangle(y_i, y) - \langle \delta \Psi_i(y), w \rangle & \text{(SVM}_1^m) \\
   (1 - \langle \delta \Psi_i(y), w \rangle) \sqrt{\triangle(y_i, y)} & \text{(SVM}_2^s) \\
   \sqrt{\triangle(y_i, y)} - \langle \delta \Psi_i(y), w \rangle & \text{(SVM}_2^m) 
   \end{cases}
   \]
   where \(w \equiv \sum_j \sum_{y' \in S_j} \alpha_{(j, y')}' \delta \Psi_j(y')\).
6: /* find cutting plane */
   compute \(\hat{y} = \arg \max_{y \in y} H(y)\)
7: /* determine value of current slack variable */
   compute \(\xi_i = \max \{0, \max_{y \in S_i} H(y)\}\)
8: **if** \(H(\hat{y}) > \xi_i + \varepsilon\) **then**
9: /* add constraint to the working set */
   \(S_i \leftarrow S_i \cup \{\hat{y}\}\)
10a: /* Variant (a): perform full optimization */
   \(\alpha_S \leftarrow \) optimize the dual of SVM_0, SVM_1^± or SVM_2^± over \(S, S = \bigcup_i S_i\).
10b: /* Variant (b): perform subspace ascent */
   \(\alpha_{S_i} \leftarrow \) optimize the dual of SVM_0, SVM_1^± or SVM_2^± over \(S_i\)
12: **end if**
13: **end for**
14: **until** no \(S_i\) has changed during iteration
To find the most violated constraint, we need to maximize w.r.t. $y$ for margin rescaling

$$ w^T \phi(x_i, y) + \Delta(y_i, y) $$

and for slack rescaling

$$ \{w^T \phi(x_i, y) + 1 - w^T \phi(x_i, y_i)\} \Delta(y_i, y) $$

For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$.
To find the most violated constraint, we need to maximize w.r.t. $y$ for margin rescaling

$$w^T \phi(x_i, y) + \Delta(y_i, y)$$

and for slack rescaling

$$\{w^T \phi(x_i, y) + 1 - w^T \phi(x_i, y_i)\} \Delta(y_i, y)$$

For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$

Use Graph-cuts or message passing
To find the most violated constraint, we need to maximize w.r.t. $y$ for margin rescaling

$$w^T \phi(x_i, y) + \Delta(y_i, y)$$

and for slack rescaling

$$\{w^T \phi(x_i, y) + 1 - w^T \phi(x_i, y_i)\} \Delta(y_i, y)$$

For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$.

Use Graph-cuts or message passing.

When the MAP cannot be computed exactly, but only approximately, this algorithm does not behave well [Fidley et al., 08]
To find the most violated constraint, we need to maximize w.r.t. $y$ for margin rescaling

$$w^T \phi(x_i, y) + \Delta(y_i, y)$$

and for slack rescaling

$$\{w^T \phi(x_i, y) + 1 - w^T \phi(x_i, y_i)\} \Delta(y_i, y)$$

For arbitrary output spaces, we would need to iterate over all elements in $Y$

- Use Graph-cuts or message passing

When the MAP cannot be computed exactly, but only approximately, this algorithm does not behave well [Fidley et al., 08]
One Slack Formulation

- Margin rescaling

\[
\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \xi \\
\text{s.t. } w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq \Delta(y_i, y) - \xi \quad \forall i \in \{1, \cdots, n\}, \forall y \in Y \setminus y_i
\]

- Slack rescaling

\[
\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \xi \\
\text{s.t. } w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 1 - \frac{\xi}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in Y \setminus y_i
\]
One Slack Formulation

- Margin rescaling

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \xi$$

s.t. $$w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq \Delta(y_i, y) - \xi \quad \forall i \in \{1, \ldots, n\}, \forall y \in Y \setminus y_i$$

- Slack rescaling

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{n} \xi$$

s.t. $$w^T \phi(x_i, y_i) - w^T \phi(x_i, y) \geq 1 - \frac{\xi}{\Delta(y_i, y)} \quad \forall i \in \{1, \ldots, n\}, \forall y \in Y \setminus y_i$$

- Same optima as previous formulation [Joachims et al, 09]
One Slack Formulation

- Margin rescaling

\[
\min_w \quad \frac{1}{2}\|w\|^2 + \frac{C}{n}\xi \\
\text{s.t. } w^T\phi(x_i, y_i) - w^T\phi(x_i, y) \geq \Delta(y_i, y) - \xi \quad \forall i \in \{1, \ldots, n\}, \forall y \in \mathcal{Y} \setminus y_i
\]

- Slack rescaling

\[
\min_w \quad \frac{1}{2}\|w\|^2 + \frac{C}{n}\xi \\
\text{s.t. } w^T\phi(x_i, y_i) - w^T\phi(x_i, y) \geq 1 - \frac{\xi}{\Delta(y_i, y)} \quad \forall i \in \{1, \ldots, n\}, \forall y \in \mathcal{Y} \setminus y_i
\]

- Same optima as previous formulation [Joachims et al, 09]
Example: Handwritten Recognition

- Predict text from image of handwritten characters
  \[
  \arg \max_y \ w^T \phi(x, y) = \text{"brace"}
  \]

- Equivalently:
  \[
  w^T \phi(x, \text{"brace"}) > w^T \phi(x, \text{"aaaaa"})
  \]
  \[
  w^T \phi(x, \text{"brace"}) > w^T \phi(x, \text{"aaaab"})
  \]
  \[
  \ldots
  \]
  \[
  w^T \phi(x, \text{"brace"}) > w^T \phi(x, \text{"zzzzz"})
  \]

- Iterate
  - Estimate model parameters \( w \) using active constraint set
  - Generate the next constraint

[Source: B. Taskar]
Conditional Random Fields

- Regularized loss minimization: Given input pairs \((x, y) \in S\), minimize
  \[
  \sum_{(x, y) \in S} \hat{\ell}(w, x, y) + \frac{C}{p} \|w\|_p^p,
  \]

- CRF loss: The conditional distribution is
  \[
  p_{x,y}(\hat{y}; w) = \frac{1}{Z(x, y)} \exp(\ell(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
  \]
  \[
  Z(x, y) = \sum_{\hat{y} \in Y} \exp(\ell(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
  \]
  where \(\ell(y, \hat{y})\) is a prior distribution and \(Z(x, y)\) the partition function, and
  \[
  \bar{\ell}_{log}(w, x, y) = \ln \frac{1}{p_{x,y}(y; w)}.
  \]
Conditional Random Fields

- Regularized loss minimization: Given input pairs \((x, y) \in S\), minimize

\[
\sum_{(x, y) \in S} \hat{\ell}(w, x, y) + \frac{C}{p} \| w \|^p_p,
\]

- CRF loss: The conditional distribution is

\[
p_{x,y}(\hat{y}; w) = \frac{1}{Z(x, y)} \exp (\ell(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
\]

\[
Z(x, y) = \sum_{\hat{y} \in \mathcal{Y}} \exp (\ell(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
\]

where \(\ell(y, \hat{y})\) is a prior distribution and \(Z(x, y)\) the partition function, and

\[
\bar{\ell}_{\log}(w, x, y) = \ln \frac{1}{p_{x,y}(y; w)}.
\]
CRF learning

In CRFs one aims to minimize the regularized negative log-likelihood of the conditional distribution

\[
\text{(CRF)} \quad \min_w \left\{ \sum_{(x,y) \in S} \ln Z(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

where \((x, y) \in S\) ranges over the training pairs and

\[
d = \sum_{(x,y) \in S} \Phi(x, y)
\]

is the vector of empirical means.

In coordinate descent methods, each coordinate \(w_r\) is iteratively updated in the direction of the negative gradient, for some step size \(\eta\).
CRF learning

- In CRFs one aims to minimize the regularized negative log-likelihood of the conditional distribution

\[
\begin{align*}
\text{(CRF)} \quad & \min_w \left\{ \sum_{(x,y) \in S} \ln Z(x, y) - d^T w + \frac{C}{p} ||w||^p_p \right\}, \\
\end{align*}
\]

where \((x, y) \in S\) ranges over the training pairs and

\[
d = \sum_{(x,y) \in S} \Phi(x, y)
\]

is the vector of empirical means.

- In coordinate descent methods, each coordinate \(w_r\) is iteratively updated in the direction of the negative gradient, for some step size \(\eta\).

- The gradient of the log-partition function corresponds to the probability distribution \(p(\hat{y}|x, y; w)\), and the direction of descent takes the form

\[
\sum_{(x,y) \in S} \sum_{\hat{y}} p(\hat{y}|x, y; w) \phi_r(x, \hat{y}) - d_r + |w_r|^{p-1} \text{sign}(w_r).
\]
CRF learning

- In CRFs one aims to minimize the regularized negative log-likelihood of the conditional distribution

\[
\text{(CRF)} \quad \min_w \left\{ \sum_{(x,y) \in S} \ln Z(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

where \((x, y) \in S\) ranges over the training pairs and

\[
d = \sum_{(x,y) \in S} \Phi(x, y)
\]

is the vector of empirical means.

- In coordinate descent methods, each coordinate \(w_r\) is iteratively updated in the direction of the negative gradient, for some step size \(\eta\).

- The gradient of the log-partition function corresponds to the probability distribution \(p(\hat{y} | x, y; w)\), and the direction of descent takes the form

\[
\sum_{(x,y) \in S} \sum_{\hat{y}} p(\hat{y} | x, y; w) \phi_r(x, \hat{y}) - d_r + |w_r|^{p-1} \text{sign}(w_r).
\]

- Problem: Requires computing the partition function!
In CRFs one aims to minimize the regularized negative log-likelihood of the conditional distribution:

\[
\text{(CRF)} \quad \min_w \left\{ \sum_{(x,y) \in S} \ln Z(x, y) - d^T w + \frac{C}{p} \|w\|^p_p \right\},
\]

where \((x, y) \in S\) ranges over the training pairs and

\[
d = \sum_{(x,y) \in S} \Phi(x, y)
\]

is the vector of empirical means.

In coordinate descent methods, each coordinate \(w_r\) is iteratively updated in the direction of the negative gradient, for some step size \(\eta\).

The gradient of the log-partition function corresponds to the probability distribution \(p(\hat{y}|x, y; w)\), and the direction of descent takes the form

\[
\sum_{(x,y) \in S} \sum_{\hat{y}} p(\hat{y}|x, y; w) \phi_r(x, \hat{y}) - d_r + |w_r|^{p-1} \text{sign}(w_r).
\]

Problem: Requires computing the partition function!
Loss functions

- Regularized loss minimization: Given input pairs \((x, y) \in S\), minimize
  \[
  \sum_{(x, y) \in S} \hat{\ell}(w, x, y) + \frac{C}{p} \|w\|_p^p,
  \]

- In structure SVMs
  \[
  \hat{\ell}_{\text{hinge}}(w, x, y) = \max_{\hat{y} \in Y} \left\{ \ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \right\}
  \]
Regularized loss minimization: Given input pairs \((x, y) \in S\), minimize
\[
\sum_{(x,y) \in S} \hat{\ell}(w, x, y) + \frac{C}{p} \|w\|^p_p,
\]

In structure SVMs
\[
\bar{\ell}_{hinge}(w, x, y) = \max_{\hat{y} \in Y} \{\ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y)\}
\]

CRF loss: The conditional distribution is
\[
p_{x,y}(\hat{y}; w) = \frac{1}{Z(x,y)} \exp (\ell(y, \hat{y}) + w^T \Phi(x, \hat{y}))
\]
\[
Z(x,y) = \sum_{\hat{y} \in Y} \exp (\ell(y, \hat{y}) + w^T \Phi(x, \hat{y}))
\]
where \(\ell(y, \hat{y})\) is a prior distribution and \(Z(x,y)\) the partition function, and
\[
\bar{\ell}_{log}(w, x, y) = \ln \frac{1}{p_{x,y}(y; w)}.
\]
Loss functions

- Regularized loss minimization: Given input pairs \((x, y) \in S\), minimize
  \[
  \sum_{(x, y) \in S} \hat{\ell}(w, x, y) + \frac{C}{p} \|w\|_p^p.
  \]

- In structure SVMs
  \[
  \bar{\ell}_{hinge}(w, x, y) = \max_{\hat{y} \in \mathcal{Y}} \left\{ \ell(y, \hat{y}) + w^\top \Phi(x, \hat{y}) - w^\top \Phi(x, y) \right\}
  \]

- CRF loss: The conditional distribution is
  \[
  p_{x,y}(\hat{y}; w) = \frac{1}{Z(x, y)} \exp \left( \ell(y, \hat{y}) + w^\top \Phi(x, \hat{y}) \right)
  \]
  \[
  Z(x, y) = \sum_{\hat{y} \in \mathcal{Y}} \exp \left( \ell(y, \hat{y}) + w^\top \Phi(x, \hat{y}) \right)
  \]
  where \(\ell(y, \hat{y})\) is a prior distribution and \(Z(x, y)\) the partition function, and
  \[
  \bar{\ell}_{log}(w, x, y) = \ln \frac{1}{p_{x,y}(y; w)}.
  \]
Relation between loss functions

- The CRF program is

\[
(CRF) \quad \min_w \left\{ \sum_{(x,y) \in S} \ln Z(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

where \((x, y) \in S\) ranges over training pairs and \(d = \sum_{(x,y) \in S} \Phi(x, y)\) is the vector of empirical means, and

\[
Z(x, y) = \sum_{\hat{y} \in \mathcal{Y}} \exp \left( \ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right)
\]

- In structured SVMs

\[
(\text{structured SVM}) \quad \min_w \left\{ \sum_{(x,y) \in S} \max_{\hat{y} \in \mathcal{Y}} \left\{ \ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right\} - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]
Relation between loss functions

- The CRF program is

\[
\text{(CRF)} \quad \min_w \left\{ \sum_{(x,y) \in S} \ln Z(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

where \((x, y) \in S\) ranges over training pairs and \(d = \sum_{(x,y) \in S} \Phi(x, y)\) is the vector of empirical means, and

\[
Z(x, y) = \sum_{\hat{y} \in Y} \exp \left( \ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right)
\]

- In structured SVMs

\[
\text{(structured SVM)} \quad \min_w \left\{ \sum_{(x,y) \in S} \max_{\hat{y} \in Y} \left\{ \ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right\} - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

A family of structure prediction problems

[T. Hazan and R. Urtasun, NIPS 2010]

- One parameter extension of CRFs and structured SVMs

\[
\min_w \left\{ \sum_{(x,y) \in S} \ln Z_{\epsilon}(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\} ,
\]

\(d\) is the empirical means, and

\[
\ln Z_{\epsilon}(x, y) = \epsilon \ln \sum_{\hat{y} \in \mathcal{Y}} \exp \left( \frac{\ell(y, \hat{y}) + w^T \Phi(x, \hat{y})}{\epsilon} \right)
\]

- CRF if \(\epsilon = 1\), Structured SVM if \(\epsilon = 0\) respectively.
A family of structure prediction problems

One parameter extension of CRFs and structured SVMs

\[
\min_w \left\{ \sum_{(x,y) \in S} \ln Z_\epsilon(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

\(d\) is the empirical means, and

\[
\ln Z_\epsilon(x, y) = \epsilon \ln \sum_{\hat{y} \in Y} \exp \left( \frac{\ell(y, \hat{y}) + w^T \Phi(x, \hat{y})}{\epsilon} \right)
\]

CRF if \(\epsilon = 1\), Structured SVM if \(\epsilon = 0\) respectively.

Introduces the notion of loss in CRFs.
A family of structure prediction problems

[T. Hazan and R. Urtasun, NIPS 2010]

- One parameter extension of CRFs and structured SVMs
  \[
  \min_w \left\{ \sum_{(x,y) \in S} \ln Z_\epsilon(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
  \]
  \(d\) is the empirical means, and
  \[
  \ln Z_\epsilon(x, y) = \epsilon \ln \sum_{\hat{y} \in Y} \exp \left( \frac{\ell(y, \hat{y}) + w^T \Phi(x, \hat{y})}{\epsilon} \right)
  \]

- CRF if \(\epsilon = 1\), Structured SVM if \(\epsilon = 0\) respectively.

- Introduces the notion of loss in CRFs.

- Dual takes the form
  \[
  \max_{p_{x,y}(\hat{y}) \in \Delta Y} \sum_{(x,y) \in S} \left( \epsilon H(p_{x,y}) + \sum_{\hat{y}} p_{x,y}(\hat{y}) \ell(y, \hat{y}) \right) - \frac{C^{1-q}}{q} \left\| \sum_{(x,y) \in S} \sum_{\hat{y} \in Y} p_{x,y}(\hat{y}) \Phi(x, \hat{y}) - d \right\|_q^q
  \]
  over the probability simplex over \(Y\).
A family of structure prediction problems

[T. Hazan and R. Urtasun, NIPS 2010]

- One parameter extension of CRFs and structured SVMs

\[
\min_w \left\{ \sum_{(x,y) \in S} \ln Z_\epsilon(x, y) - d^T w + \frac{C}{p} \|w\|^p_p \right\},
\]

\(d\) is the empirical means, and

\[
\ln Z_\epsilon(x, y) = \epsilon \ln \sum_{\hat{y} \in \mathcal{Y}} \exp \left( \frac{\ell(y, \hat{y}) + w^T \Phi(x, \hat{y})}{\epsilon} \right)
\]

- CRF if \(\epsilon = 1\), Structured SVM if \(\epsilon = 0\) respectively.

- Introduces the notion of loss in CRFs.

- Dual takes the form

\[
\max_{p_{x,y}(\hat{y}) \in \Delta_{\mathcal{Y}}} \sum_{(x,y) \in S} \left( \epsilon H(p_{x,y}) + \sum_{\hat{y}} p_{x,y}(\hat{y}) \ell(y, \hat{y}) \right) - \frac{C^{1-q}}{q} \left\| \sum_{(x,y) \in S} \sum_{\hat{y} \in \mathcal{Y}} p_{x,y}(\hat{y}) \Phi(x, \hat{y}) - d \right\|_q^q
\]

over the probability simplex over \(\mathcal{Y}\).
Primal-Dual approximated learning algorithm

[T. Hazan and R. Urtasun, NIPS 2010]

- In many applications the features decompose

\[ \phi_r(x, \hat{y}_1, \ldots, \hat{y}_n) = \sum_{v \in V_r, x} \phi_{r,v}(x, \hat{y}_v) + \sum_{\alpha \in E_r, x} \phi_{r,\alpha}(x, \hat{y}_\alpha). \]

- Using this we can write the approximate program as

\[
\min_{\lambda_{x,y,v \rightarrow \alpha}, w} \sum_{(x,y) \in S, v} \epsilon_{c_v} \ln \sum_{\hat{y}_v} \exp \left( \frac{\ell_v(y_v, \hat{y}_v) + \sum_{r: v \in V_r, x} w_r \phi_{r,v}(x, \hat{y}_v) - \sum_{\alpha \in N(v)} \lambda_{x,y,v \rightarrow \alpha}(\hat{y}_v)}{\epsilon_{c_v}} \right) + \sum_{(x,y) \in S, \alpha} \epsilon_{c_\alpha} \ln \sum_{\hat{y}_\alpha} \exp \left( \frac{\sum_{r: \alpha \in E_r} w_r \phi_{r,\alpha}(x, \hat{y}_\alpha) + \sum_{v \in N(\alpha)} \lambda_{x,y,v \rightarrow \alpha}(\hat{y}_v)}{\epsilon_{c_\alpha}} \right) - d^T w - \frac{C}{p} \|w\|_p^p
\]
In many applications the features decompose

\[ \phi_r(x, \hat{y}_1, ..., \hat{y}_n) = \sum_{v \in V_{r,x}} \phi_{r,v}(x, \hat{y}_v) + \sum_{\alpha \in E_{r,x}} \phi_{r,\alpha}(x, \hat{y}_\alpha). \]

Using this we can write the approximate program as

\[
\begin{aligned}
\min_{\lambda_{x,y,v} \rightarrow \alpha, w} & \sum_{(x,y) \in S,v} \epsilon c_v \ln \sum_{\hat{y}_v} \exp \left( \frac{\ell_v(y_v, \hat{y}_v) + \sum_{r:v \in V_{r,x}} w_r \phi_{r,v}(x, \hat{y}_v) - \sum_{\alpha \in N(v)} \lambda_{x,y,v} \rightarrow \alpha(\hat{y}_v)}{\epsilon c_v} \right) \\
& + \sum_{(x,y) \in S,\alpha} \epsilon c_\alpha \ln \sum_{\hat{y}_\alpha} \exp \left( \frac{\sum_{r:\alpha \in E_r} w_r \phi_{r,\alpha}(x, \hat{y}_\alpha) + \sum_{v \in N(\alpha)} \lambda_{x,y,v} \rightarrow \alpha(\hat{y}_v)}{\epsilon c_\alpha} \right) - d^T w - \frac{C}{p} \|w\|_p^p
\end{aligned}
\]

Coordinate descent algorithm that alternates between sending messages and updating parameters.
In many applications the features decompose

$$\phi_r(x, \hat{y}_1, \ldots, \hat{y}_n) = \sum_{v \in V_{r,x}} \phi_{r,v}(x, \hat{y}_v) + \sum_{\alpha \in E_{r,x}} \phi_{r,\alpha}(x, \hat{y}_\alpha).$$

Using this we can write the approximate program as

$$\min_{\lambda_{x,y,v} \rightarrow \alpha, w, \sum_{(x,y) \in S, v} \epsilon c_v \ln \sum_{\hat{y}_v} \exp \left( \frac{\ell_v(y_v, \hat{y}_v) + \sum_{r:v \in V_{r,x}} w_r \phi_{r,v}(x, \hat{y}_v) - \sum_{\alpha \in N(v)} \lambda_{x,y,v} \rightarrow \alpha(\hat{y}_v)}{\epsilon c_v} \right)$$

$$+ \sum_{(x,y) \in S, \alpha} \epsilon c_\alpha \ln \sum_{\hat{y}_\alpha} \exp \left( \frac{\sum_{r:\alpha \in E_r} w_r \phi_{r,\alpha}(x, \hat{y}_\alpha) + \sum_{v \in N(\alpha)} \lambda_{x,y,v} \rightarrow \alpha(\hat{y}_v)}{\epsilon c_\alpha} \right) - d^T w - \frac{C}{p} \|w\|_p^p$$

Coordinate descent algorithm that alternates between sending messages and updating parameters.

Advantage: doesn’t need the MAP or marginal at each gradient step.
Primal-Dual approximated learning algorithm

In many applications the features decompose

\[ \phi_r(x, \hat{y}_1, \ldots, \hat{y}_n) = \sum_{v \in V_r, x} \phi_{r,v}(x, \hat{y}_v) + \sum_{\alpha \in E_r, x} \phi_{r,\alpha}(x, \hat{y}_\alpha). \]

Using this we can write the approximate program as

\[
\begin{align*}
\min_{\lambda_{x,y,v \rightarrow \alpha} \cdot w} & \sum_{(x,y) \in S, v} \epsilon_{c_v} \ln \sum_{\hat{y}_v} \exp \left( \frac{\ell_v(y_v, \hat{y}_v) + \sum_{r: v \in V_r, x} w_r \phi_{r,v}(x, \hat{y}_v) - \sum_{\alpha \in N(v)} \lambda_{x,y,v \rightarrow \alpha}(\hat{y}_v)}{\epsilon_{c_v}} \right) \\
& + \sum_{(x,y) \in S, \alpha} \epsilon_{c_\alpha} \ln \sum_{\hat{y}_\alpha} \exp \left( \frac{\sum_{r: \alpha \in E_r} w_r \phi_{r,\alpha}(x, \hat{y}_\alpha) + \sum_{v \in N(\alpha)} \lambda_{x,y,v \rightarrow \alpha}(\hat{y}_v)}{\epsilon_{c_\alpha}} \right) - d^T w - \frac{C}{p} \|w\|^p_p
\end{align*}
\]

Coordinate descent algorithm that alternates between sending messages and updating parameters.

Advantage: doesn’t need the MAP or marginal at each gradient step.

Can learn a large set of parameters.
In many applications the features decompose
\[ \phi_r(x, \hat{y}_1, \ldots, \hat{y}_n) = \sum_{v \in V_r, x} \phi_{r, v}(x, \hat{y}_v) + \sum_{\alpha \in E_r, x} \phi_{r, \alpha}(x, \hat{y}_\alpha). \]

Using this we can write the approximate program as
\[
\min_{\lambda(x, y, v \rightarrow \alpha), w} \sum_{(x, y) \in S, v} \epsilon c_v \ln \sum_{\hat{y}_v} \exp \left( \frac{\ell_v(y_v, \hat{y}_v) + \sum_{r: v \in V_r, x} w_r \phi_{r, v}(x, \hat{y}_v) - \sum_{\alpha \in N(v)} \epsilon c_v \lambda(x, y, v \rightarrow \alpha(\hat{y}_v))}{\epsilon c_v} \right) \\
+ \sum_{(x, y) \in S, \alpha} \epsilon c_\alpha \ln \sum_{\hat{y}_\alpha} \exp \left( \frac{\sum_{r: \alpha \in E_r} w_r \phi_{r, \alpha}(x, \hat{y}_\alpha) + \sum_{v \in N(\alpha)} \epsilon c_\alpha \lambda(x, y, v \rightarrow \alpha(\hat{y}_v))}{\epsilon c_\alpha} \right) - d^T w - \frac{C}{p} \|w\|_p^p
\]

Coordinate descent algorithm that alternates between sending messages and updating parameters.

Advantage: doesn’t need the MAP or marginal at each gradient step.

Can learn a large set of parameters.

Code will be available soon, including parallel implementation.
In many applications the features decompose

\[ \phi_r(x, \hat{y}_1, \ldots, \hat{y}_n) = \sum_{v \in V_{r,x}} \phi_{r,v}(x, \hat{y}_v) + \sum_{\alpha \in E_{r,x}} \phi_{r,\alpha}(x, \hat{y}_\alpha). \]

Using this we can write the approximate program as

\[
\min_{\lambda_{x,y,v} \rightarrow \alpha, w} \sum_{(x,y) \in S, v} \epsilon c_v \ln \sum_{\hat{y}_v} \exp \left( \frac{\ell_v(y_v, \hat{y}_v) + \sum_{r: v \in V_{r,x}} w_r \phi_{r,v}(x, \hat{y}_v) - \sum_{\alpha \in N(v)} \lambda_{x,y,v} \rightarrow \alpha(\hat{y}_v)}{\epsilon c_v} \right) \\
+ \sum_{(x,y) \in S, \alpha} \epsilon c_\alpha \ln \sum_{\hat{y}_\alpha} \exp \left( \frac{\sum_{r: \alpha \in E_r} w_r \phi_{r,\alpha}(x, \hat{y}_\alpha) + \sum_{v \in N(\alpha)} \lambda_{x,y,v} \rightarrow \alpha(\hat{y}_v)}{\epsilon c_\alpha} \right) - d^T w - \frac{C}{p} \|w\|_p^p
\]

Coordinate descent algorithm that alternates between sending messages and updating parameters.

Advantage: doesn’t need the MAP or marginal at each gradient step.

Can learn a large set of parameters.

Code will be available soon, including parallel implementation.
Message-Passing algorithm for Approximated Structured Prediction:
Set \( \bar{e}_{y,v}(\hat{y}_v) = \exp(e_{y,v}(\hat{y}_v)) \) and similarly \( \bar{\phi}_{r,v}, \bar{\phi}_{r,\alpha} \).

1. For \( t = 1, 2, \ldots \)
   (a) For every \( v = 1, \ldots, n \), every \( (x, y) \in S \), every \( \alpha \in N(v) \), every \( \hat{y}_v \in Y_v \) do:

   \[
   m_{x,y,\alpha \rightarrow v}(\hat{y}_v) = \left\| \prod_{r: \alpha \in E_r} \bar{\phi}_{r,\alpha}^\theta(x, \hat{y}_\alpha) \prod_{u \in N(\alpha) \setminus v} n_{x,y,u \rightarrow \alpha}(\hat{y}_u) \right\|^{1/\epsilon_{\alpha}_v}
   \]

   \[
   n_{x,y,v \rightarrow \alpha}(\hat{y}_v) \propto \left( \bar{e}_{y,v}(\hat{y}_v) \prod_{r: v \in V_r} \bar{\phi}_{r,v}^\theta(x, \hat{y}_r) \prod_{\beta \in N(v)} m_{x,y,\beta \rightarrow v}(\hat{y}_v) \right)^{c_{\alpha}/\epsilon_v} / m_{x,y,\alpha \rightarrow v}(\hat{y}_v)
   \]

   (b) For every \( r = 1, \ldots, d \) do:

   For every \( (x, y) \in S \), every \( v \in V_{r,x} \), \( \alpha \in E_{r,x} \), every \( \hat{y}_v \in Y_v \), \( \hat{y}_\alpha \in Y_\alpha \) set:

   \[
   b_{x,y,v}(\hat{y}_v) \propto \left( \bar{e}_{y,r}(\hat{y}_v) \prod_{r: v \in V_{r,x}} \bar{\phi}_{r,v}^\theta(x, \hat{y}_r) \prod_{\alpha \in N(v)} n_{x,y,v \rightarrow \alpha}(\hat{y}_v) \right)^{1/\epsilon_{C_r}}
   \]

   \[
   b_{x,y,\alpha}(\hat{y}_\alpha) \propto \left( \prod_{r: \alpha \in E_{r,x}} \bar{\phi}_{r,\alpha}^\theta(x, \hat{y}_\alpha) \prod_{v \in N(\alpha)} n_{x,y,v \rightarrow \alpha}(\hat{y}_v) \right)^{1/\epsilon_{C_{\alpha}}}
   \]

   \[
   \theta_r \leftarrow \theta_r - \eta \left( \sum_{(x,y) \in S, v \in V_{r,x}, \hat{y}_v} b_{x,y,v}(\hat{y}_v) \phi_{r,v}(x, \hat{y}_v) + \sum_{(x,y) \in S, \alpha \in E_{r,x}, \hat{y}_\alpha} b_{x,y,\alpha}(\hat{y}_\alpha) \phi_{r,\alpha}(x, \hat{y}_\alpha) - c_r + C \cdot |\theta_r|^{p-1} \cdot \text{sign}(\theta_r) \right)
   \]
Examples in computer vision
Examples

- Depth estimation
- Multi-label prediction
- Object detection
- Non-maxima suppression
- Segmentation
- Sentence generation
- Holistic scene understanding
- 2D pose estimation
- Non-rigid shape estimation
- 3D scene understanding
- \ldots
For each application ...

... what do we need to decide?

- Random variables
- Graphical model
- Potentials
- Loss for learning
- Learning algorithm
- Inference algorithm

Let’s look at some examples
Depth Estimation

• Images rectified
• Ignore occlusion for now

Energy:

\[ E(d): \{0, \ldots, D-1\}^n \rightarrow \mathbb{R} \]

Labels: d (depth/shift)

[Source: P. Kohli]
Energy:

\[ E(d): \{0,...,D-1\}^n \rightarrow \mathbb{R} \]

\[ E(d) = \sum_i \Theta_i(d_i) + \sum_{i,j \in N_4} \Theta_{ij}(d_i,d_j) \]

Unary:

\[ \Theta_i(d_i) = (l_j - r_{i-d_i}) \]

"SAD; Sum of absolute differences"

(many others possible, NCC,...)

Pairwise:

\[ \Theta_{ij}(d_i,d_j) = g(|d_i - d_j|) \]

[Source: P. Kohli]
Stereo matching: energy

\[ \theta_{ij}(d_i, d_j) = g(|d_i - d_j|) \]

No truncation (global min.)

[Source: P. Kohli]
Stereo matching: energy

\[ \theta_{ij}(d_i, d_j) = g(|d_i - d_j|) \]

- No truncation (global min.)
- with truncation (NP hard optimization)

*discontinuity preserving potentials*

[Blake & Zisserman '83, '87]

[Source: P. Kohli]
More on pairwise [O. Veksler]

\[ \theta_{ij}(d_i,d_j) = g(|d_i-d_j|) \]

- Left image
- (Potts model)
- Smooth disparities

[Source: P. Kohli]
Graph Structure

- No MRF
  - Pixel independent (WTA)
- No horizontal links
  - Efficient since independent chains
- Pairwise MRF
  - [Boykov et al. ‘01]
- Ground truth

see http://vision.middlebury.edu/stereo/
- There is only one parameter to learn: importance of pairwise with respect to unitary!
- Sum of square differences: outliers are more important
- % of pixels that have disparity error bigger than $\epsilon$.
- The latter is how typically stereo algorithms are scored
- Which inference method will you choose?
- And for learning?
We can formulate object localization as a regression from an image to a bounding box

\[ g : \mathcal{X} \rightarrow \mathcal{Y} \]

- \( \mathcal{X} \) is the space of all images
- \( \mathcal{Y} \) is the space of all bounding boxes
Joint Kernel Between bboxes

- Note: $x|_{y}$ (the image restricted to the box region) is again an image.
- Compare two images with boxes by comparing the images within the boxes:

$$k_{\text{joint}}((x, y), (x', y')) = k_{\text{image}}(x|_{y}, x'|_{y'})$$

- Any common image kernel is applicable:
  - linear on cluster histograms: $k(h, h') = \sum_i h_i h'_i$,
  - $\chi^2$-kernel: $k_{\chi^2}(h, h') = \exp \left( -\frac{1}{\gamma} \sum_i \frac{(h_i - h'_i)^2}{h_i + h'_i} \right)$
  - pyramid matching kernel, ...
- The resulting joint kernel is positive definite.

[Source: M. Blascko]
Restriction Kernel example

\[ k_{\text{joint}}(\begin{array}{c} \text{cow} \\ \text{mountains} \end{array}, \begin{array}{c} \text{cow} \\ \text{mountains} \end{array}) = k(\begin{array}{c} \text{cow} \\ \text{mountains} \end{array}, \begin{array}{c} \text{cow} \\ \text{mountains} \end{array}) \]

is large.

\[ k_{\text{joint}}(\begin{array}{c} \text{cow} \\ \text{mountains} \end{array}, \begin{array}{c} \text{cow} \\ \text{mountains} \end{array}) = k(\begin{array}{c} \text{cow} \\ \text{mountains} \end{array}, \begin{array}{c} \text{cow} \\ \text{mountains} \end{array}) \]

is small.

\[ k_{\text{joint}}(\begin{array}{c} \text{plane} \\ \text{mountains} \end{array}, \begin{array}{c} \text{cow} \\ \text{mountains} \end{array}) = k(\begin{array}{c} \text{plane} \\ \text{mountains} \end{array}, \begin{array}{c} \text{cow} \\ \text{mountains} \end{array}) \]

could also be large.

- Note: This behaves differently from the common tensor products

\[ k_{\text{joint}}( (x, y), (x', y') ) \neq k(x, x')k(y, y') \]

[Source: M. Blascko]
\[ \langle w, \varphi(x_i, y_i) \rangle - \langle w, \varphi(x_i, y) \rangle \geq \Delta(y_i, y) - \xi_i, \ \forall i, \forall y \in \mathcal{Y} \setminus y_i \]

\[ \mathcal{Y} \equiv \{(\omega, t, b, l, r) \mid \omega \in \{+1, -1\}, \ (t, b, l, r) \in \mathbb{R}^4\} \]

\[ \Delta(y_i, y) = 1 - \frac{\text{Area}(y_i \cap y)}{\text{Area}(y_i \cup y)} \]

[Source: M. Blascko]
Constraint Generation with Branch and Bound

- As before, we must solve

\[
\max_{y \in \mathcal{Y}} \langle w, \varphi(x_i, y) \rangle + \Delta(y_i, y)
\]

where

\[
\Delta(y_i, y) = 1 - \frac{\text{Area}(y_i \cap y)}{\text{Area}(y_i \cup y)}
\]

- Solution: use branch-and-bound over the space of all rectangles in the image

(Blaschko & Lampert, 2008)

[Source: M. Blascko]
Branch-and-Bound works with subsets of the search space.

- Instead of four numbers \([l, t, r, b]\), store four intervals \([L, T, R, B]\):

\[
L = [l_{lo}, l_{hi}]
\]
\[
T = [t_{lo}, t_{hi}]
\]
\[
R = [r_{lo}, r_{hi}]
\]
\[
B = [b_{lo}, b_{hi}]
\]

[Source: M. Blascko]
• Train using constraint generation
  ▶ Train an SVM with margin rescaling
  ▶ Identify the most violated constraint with branch and bound and add it to the constraint set

\[
\max_{y \in \mathcal{Y} \setminus y_i} \sum_{j=1}^{n} \sum_{\tilde{y} \in \mathcal{Y}} \alpha_{j\tilde{y}} \left( k_x(x_j|y_j, x_i|y) - k_x(x_j|\tilde{y}, x_i|y) \right) + \Delta(y_i, y)
\]

upper bound this term

• iterate until convergence criterion is reached

[Source: M. Blascko]
Results: PASCAL VOC2006

- ≈5,000 images: ≈2,500 train/val, ≈2,500 test
- ≈9,500 objects in 10 predefined classes:
  - bicycle, bus, car, cat, cow, dog, horse, motorbike, person, sheep
- Task: predict locations and confidence scores for each class
- Evaluation: Precision-Recall curves

![Precision-Recall curves](image)

VOC 2006 detection, class **cat**: old and new training vs. VOC2006 participants

[Source: M. Blascko]
Results: PASCAL VOC2006 cats

[Source: M. Blascko]
Problem

- The restriction kernel is like having tunnel vision

[Source: M. Blascko]
Problem

- The restriction kernel is like having tunnel vision

[Source: M. Blascko]
Global and Local Context Kernels

- Augment restriction kernel with contextual cues
- Global context kernel:
  \[ k_{global}( (x_i, y_i), (x_j, y_j) ) = k_I(x_i, x_j) \]

- Local context kernel:
  \[ k_{local}( (x_i, y_i), (x_j, y_j); \theta) = k_I(x_i|\Theta(y_i), x_j|\Theta(y_j)) \]

- Putting it all together:
  \[ k( (x_i, y_i), (x_j, y_j) ) = \beta_1 k_{restr}( (x_i, y_i), (x_j, y_j) ) + \beta_2 k_{local}( (x_i, y_i), (x_j, y_j); \theta) + \beta_3 k_{global}( (x_i, y_i), (x_j, y_j) ) \]

- \( \beta \) can be learned using multiple kernel learning  
  Blaschko & Lampert, 2009

[Source: M. Blascko]
Define local context as region *between* bounding box \((l, t, r, b)\) and

\[
\bar{\Theta}(y) = (l - \theta(r - l), t - \theta(b - t), r + \theta(r - l), b + \theta(b - t))
\]

The spatial extent of a local context kernel is indicated by the shaded region.

Model the statistics of an object’s neighborhood

Don’t model the statistics of the object itself

[Source: M. Blascko]
Context is a very busy area of research in vision!

<table>
<thead>
<tr>
<th></th>
<th>bicycle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>dog</th>
<th>cow</th>
</tr>
</thead>
<tbody>
<tr>
<td>learned</td>
<td>0.410</td>
<td>0.253</td>
<td>0.268</td>
<td>0.415</td>
<td>0.332</td>
<td>0.286</td>
</tr>
<tr>
<td>fixed</td>
<td>0.429</td>
<td>0.177</td>
<td>0.263</td>
<td>0.251</td>
<td>0.178</td>
<td>0.194</td>
</tr>
<tr>
<td>no context</td>
<td>0.396</td>
<td>0.100</td>
<td>0.145</td>
<td>0.259</td>
<td>0.170</td>
<td>0.118</td>
</tr>
</tbody>
</table>

[Source: M. Blascko]
**Task**: Given an image, predict the 3D parametric cuboid that best describes the layout.
Variables are not independent of each other, i.e. structured prediction

- \( x \): Input image
- \( y \): Room layout
- \( \phi(x, y) \): Multidimensional feature vector
- \( w \): Predictor
- Estimate room layout by solving inference task

\[
\hat{y} = \arg \max_y w^T \phi(x, y)
\]

- Learning \( w \) via structured SVMs or CRFs
Approaches of [Hedau et al. 09] and [Lee et al. 10].

One random variable $y$ for the entire layout.

Every state denotes a different candidate layout.

Limits the amount of candidate layouts.

Not really a structured prediction task.

$n$ states/3D layouts have to be evaluated exhaustively, e.g., $50^4$. 

Four Variable Parameterization

- Approach of [Wang et al. 10].
- 4 variables $y_i \in \mathcal{Y}$, $i \in \{1, \ldots, 4\}$ corresponding to the four degrees of freedom of the problem.
- One state of $y_i$ denotes the angle of ray $r_i$.
- High order potentials, e.g., $50^4$ for fourth-order.

For both parameterizations is even worst when reasoning about objects.
We follow [Wang et al. 10] and parameterize with four random variables.

We follow [Lee et al. 10] and employ orientation map [Lee09 et al.] and geometric context [Hoiem et al. 07] as image cues.
Integral Geometry for Features

- Faces $\mathcal{F} = \{\text{left-wall, right-wall, ceiling, floor, front-wall}\}$
- Faces are defined by four (front-wall) or three angles (otherwise)

$$w^T \cdot \phi(x, y) = \sum_{\alpha \in \mathcal{F}} w^T_{o,\alpha} \phi_{o,\alpha}(x, y_\alpha) + \sum_{\alpha \in \mathcal{F}} w^T_{g,\alpha} \phi_{g,\alpha}(x, y_\alpha)$$

- Features count frequencies of image cues

Orientation map and proposed left wall
Using inspiration from integral images, we decompose

\[
\phi_{.,\alpha}(x, y_\alpha) = \phi_{.,\{i,j,k\}}(x, y_i, y_j, y_k) = \\
= H_{.,\{i,j\}}(x, y_i, y_j) - H_{.,\{j,k\}}(x, y_j, y_k)
\]

Integral geometry
Integral Geometry for Features

- Decomposition:
  \[ H_{\{i,j\}}(x, y_i, y_j) - H_{\{j,k\}}(x, y_j, y_k) \]

- Corresponding factor graph:

- The front-wall:
  \[ \phi_{\cdot, \text{front-wall}} = \phi(x) - \phi_{\cdot, \text{left-wall}} - \phi_{\cdot, \text{right-wall}} - \phi_{\cdot, \text{ceiling}} - \phi_{\cdot, \text{floor}} \]
Integral Geometry

- Same concept as integral images, but in accordance with the vanishing points.

**Figure:** Concept of integral geometry
Learning

- Family of structure prediction problems including CRF and structured-SVMs as special cases.
- Primal-dual algorithm based on local updates.
- Fast and works well with large number of parameters.
- Code coming soon!

[T. Hazan and R. Urtasun, NIPS 2010]

Inference

- Inference using parallel convex belief propagation
- Convergence and other theoretical guarantees
- **Code available online**: general potentials, cross-platform, Amazon EC2!

[A. Schwing, T. Hazan, M. Pollefeys and R. Urtasun, CVPR 2011]
Learning very fast: State-of-the-art after less than a minute!

Inference as little as 10ms per image!
Table: Pixel classification error in the layout dataset of [Hedau et al. 09].

<table>
<thead>
<tr>
<th></th>
<th>OM</th>
<th>GC</th>
<th>OM + GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Hoiem07]</td>
<td>-</td>
<td>28.9</td>
<td>-</td>
</tr>
<tr>
<td>[Hedau09] (a)</td>
<td>-</td>
<td>26.5</td>
<td>-</td>
</tr>
<tr>
<td>[Hedau09] (b)</td>
<td>-</td>
<td>21.2</td>
<td>-</td>
</tr>
<tr>
<td>[Wang10]</td>
<td>22.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Lee10]</td>
<td>24.7</td>
<td>22.7</td>
<td>18.6</td>
</tr>
<tr>
<td>Ours (SVM\textsuperscript{struct})</td>
<td>19.5</td>
<td>18.2</td>
<td>16.8</td>
</tr>
<tr>
<td>Ours (struct-pred)</td>
<td>18.6</td>
<td>15.4</td>
<td>13.6</td>
</tr>
</tbody>
</table>

Table: Pixel classification error in the bedroom data set [Hedau et al. 10].

<table>
<thead>
<tr>
<th></th>
<th>[Luca11]</th>
<th>[Hoiem07]</th>
<th><a href="a">Hedau09</a></th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o box</td>
<td>29.59</td>
<td>23.04</td>
<td>22.94</td>
<td>16.46</td>
</tr>
</tbody>
</table>
Simple object reasoning

- Compatibility of 3D object candidates and layout
Results

[A. Schwing, T. Hazan, M. Pollefeys and R. Urtasun, CVPR12]

**Table:** Pixel classification error in the layout dataset of [Hedau et al. 09].

<table>
<thead>
<tr>
<th></th>
<th>OM</th>
<th>GC</th>
<th>OM + GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Hoiem07]</td>
<td>-</td>
<td>28.9</td>
<td>-</td>
</tr>
<tr>
<td>[Hedau09] (a)</td>
<td>-</td>
<td>26.5</td>
<td>-</td>
</tr>
<tr>
<td>[Hedau09] (b)</td>
<td>-</td>
<td>21.2</td>
<td>-</td>
</tr>
<tr>
<td>[Wang10]</td>
<td>22.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Lee10]</td>
<td>24.7</td>
<td>22.7</td>
<td>18.6</td>
</tr>
<tr>
<td>Ours (SVM\textsuperscript{struct})</td>
<td>19.5</td>
<td>18.2</td>
<td>16.8</td>
</tr>
<tr>
<td>Ours (struct-pred)</td>
<td>18.6</td>
<td>15.4</td>
<td>13.6</td>
</tr>
</tbody>
</table>

**Table:** WITH object reasoning.

<table>
<thead>
<tr>
<th></th>
<th>OM</th>
<th>GC</th>
<th>OM + GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Wang10]</td>
<td>20.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Lee10]</td>
<td>19.5</td>
<td>20.2</td>
<td>16.2</td>
</tr>
<tr>
<td>Ours (SVM\textsuperscript{struct})</td>
<td>18.5</td>
<td>17.7</td>
<td>16.4</td>
</tr>
<tr>
<td>Ours (struct-pred)</td>
<td>17.1</td>
<td>14.2</td>
<td>12.8</td>
</tr>
</tbody>
</table>
[A. Schwing, T. Hazan, M. Pollefeys and R. Urtasun, CVPR12]

Table: Pixel classification error in the layout dataset of [Hedau et al. 09] with object reasoning.

<table>
<thead>
<tr>
<th></th>
<th>OM</th>
<th>GC</th>
<th>OM + GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Wang10]</td>
<td>20.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Lee10]</td>
<td>19.5</td>
<td>20.2</td>
<td>16.2</td>
</tr>
<tr>
<td>Ours (SVM&lt;sub&gt;struct&lt;/sub&gt;)</td>
<td>18.5</td>
<td>17.7</td>
<td>16.4</td>
</tr>
<tr>
<td>Ours (struct-pred)</td>
<td>17.1</td>
<td>14.2</td>
<td>12.8</td>
</tr>
</tbody>
</table>

Table: Pixel classification error in the bedroom data set [Hedau et al. 10].

<table>
<thead>
<tr>
<th></th>
<th>[Luca11]</th>
<th>[Hoiem07]</th>
<th><a href="a">Hedau09</a></th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o box</td>
<td>29.59</td>
<td>23.04</td>
<td>22.94</td>
<td>16.46</td>
</tr>
<tr>
<td>w/ box</td>
<td>26.79</td>
<td>-</td>
<td>22.94</td>
<td>15.19</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

Conclusion:
- Efficient learning and inference tools for structure prediction based on primal-dual methods.
- Inference: No need for application specific moves.
- Learning: can learn large number of parameters using local updates.
- State-of-the-art results.

Future Work:
- More features.
- Better object reasoning.
- Weakly label setting.
- Better inference?