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

Parameter learning

- The MAP problem was defined as

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

- Learn parameters $\mathbf{w}$ for more accurate prediction

$$\max_{y_1,\cdots,y_n} \sum_i \mathbf{w}_i \phi_i(y_i) + \sum_\alpha \mathbf{w}_\alpha \phi_\alpha(y_\alpha)$$

Loss functions

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

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

- Different learning frameworks depending on the surrogate loss $\hat{\ell}(\mathbf{w}, x, y)$
  - 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]

Loss functions

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

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

- Different learning frameworks depending on the surrogate loss $\hat{\ell}(\mathbf{w}, x, y)$
  - 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

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \xi_i$$

$$\text{subject to} \;\; y_i(b + \mathbf{w}^T\mathbf{x}_i) - 1 + \xi_i \; \geq \; 0, \quad \forall i = 1, \dots, N.$$

  with $y_i \in \{-1, 1\}$ binary.

- We need to extend this to reason about more complex structures, not just binary variables.

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 **w**, the parameters to learn.

- We will like to minimize the empirical risk

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

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 **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

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 **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 **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 $y = y'$

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 **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

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

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

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

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

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

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

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

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

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

- 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

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

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

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

- What's the problem of this?

Separable case

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

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

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

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

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

- This is a quadratic program, so it's convex

Separable case

- Satisfying the inequalities might have more than one solution.
- Select the **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
- But it involves exponentially many constraints!

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

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

- This is a quadratic program, so it's convex
- 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

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

$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \xi_i \quad \forall i \in \{1, \cdots, n\}, \forall y \neq y_i$$

- Can also be written in terms of kernels

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

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]

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

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

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

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

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

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

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

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

Margin re-scaling

- In this case the minimization problem is formulated as

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

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

- 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^{*}.

Input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n), C, \varepsilon$

$S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$

repeat

  for $i = 1, \ldots, n$ **do**

    / prepare cost function for optimization */*

    set up cost function

$$H(\mathbf{y}) \equiv \begin{cases} 1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (\mathrm{SVM}_0) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \triangle(\mathbf{y}_i, \mathbf{y}) & (\mathrm{SVM}_1^{\triangle s}) \\ \triangle(\mathbf{y}_i, \mathbf{y}) - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (\mathrm{SVM}_1^{\triangle m}) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} & (\mathrm{SVM}_2^{\triangle s}) \\ \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (\mathrm{SVM}_2^{\triangle m}) \end{cases}$$

    where $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{(j\mathbf{y}')} \delta\Psi_j(\mathbf{y}')$.

    / find cutting plane */*

    compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$

    / determine value of current slack variable */*

    compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$

    if $H(\hat{\mathbf{y}}) > \xi_i + \varepsilon$ **then**

      / add constraint to the working set */*

      $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$

      / Variant (a): perform full optimization */*

      $\alpha_S \leftarrow$ optimize the dual of SVM_0, SVM_1^{*} or SVM_2^{*} over S, $S = \cup_i S_i$.

      / Variant (b): perform subspace ascent */*

      $\alpha_{S_i} \leftarrow$ optimize the dual of SVM_0, SVM_1^{*} or SVM_2^{*} over S_i

    end if

  end for

until no S_i has changed during iteration

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling
$$\mathbf{w}^T\phi(x_i, y) + \Delta(y_i, y)$$
and for slack rescaling
$$\{\mathbf{w}^T\phi(x_i, y) + 1 - \mathbf{w}^T\phi(x_i, y_i)\}\Delta(y_i, y)$$

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

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling

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

  and for slack rescaling

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

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$
- Use Graph-cuts or message passing

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling

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

and for slack rescaling

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

- 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]

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling

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

and for slack rescaling

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

- 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]

One Slack Formulation

- Margin rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$

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

- Slack rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$

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

One Slack Formulation

- Margin rescaling

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

- Slack rescaling

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

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

One Slack Formulation

- Margin rescaling

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

- Slack rescaling

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

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

Example: Handwritten Recognition

- Predict text from image of handwritten characters

$$\arg\max_{\mathbf{y}} \ \mathbf{w}^\top \mathbf{f}(\quad, \mathbf{y}) = \text{"brace"}$$

- Equivalently:

$$\mathbf{w}^\top \mathbf{f}(\quad, \text{"brace"}) > \mathbf{w}^\top \mathbf{f}(\quad, \text{"aaaaa"})$$

$$\mathbf{w}^\top \mathbf{f}(\quad, \text{"brace"}) > \mathbf{w}^\top \mathbf{f}(\quad, \text{"aaaab"})$$

$$\dots$$

$$\mathbf{w}^\top \mathbf{f}(\quad, \text{"brace"}) > \mathbf{w}^\top \mathbf{f}(\quad, \text{"zzzzz"})$$

- 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 $(x, y) \in \mathcal{S}$, minimize

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

- CRF loss: The conditional distribution is

$$p_{x,y}(\hat{y}; \mathbf{w}) = \frac{1}{Z(x,y)} \exp\left(\ell(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\ell(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

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

$$\bar{\ell}_{log}(\mathbf{w}, x, y) = \ln \frac{1}{p_{x,y}(y; \mathbf{w})}.$$

Conditional Random Fields

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

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

- CRF loss: The conditional distribution is

$$p_{x,y}(\hat{y}; \mathbf{w}) = \frac{1}{Z(x,y)} \exp\left(\ell(y, \hat{y}) + \mathbf{w}^\top \Phi(x, \hat{y})\right)$$

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\ell(y, \hat{y}) + \mathbf{w}^\top \Phi(x, \hat{y})\right)$$

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

$$\bar{\ell}_{log}(\mathbf{w}, x, y) = \ln \frac{1}{p_{x,y}(y; \mathbf{w})}.$$

CRF learning

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

$$\text{(CRF)} \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  where $(x,y) \in \mathcal{S}$ ranges over the training pairs and

$$\mathbf{d} = \sum_{(x,y)\in\mathcal{S}} \Phi(x,y)$$

  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 η.

CRF learning

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

$$(\text{CRF}) \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  where $(x,y) \in \mathcal{S}$ ranges over the training pairs and

$$\mathbf{d} = \sum_{(x,y)\in\mathcal{S}} \Phi(x,y)$$

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

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

CRF learning

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

$$\text{(CRF)} \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  where $(x,y) \in \mathcal{S}$ ranges over the training pairs and

$$\mathbf{d} = \sum_{(x,y)\in\mathcal{S}} \Phi(x,y)$$

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

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

- Problem: Requires computing the partition function!

CRF learning

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

$$\text{(CRF)} \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  where $(x,y) \in \mathcal{S}$ ranges over the training pairs and

$$\mathbf{d} = \sum_{(x,y)\in\mathcal{S}} \Phi(x,y)$$

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

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

- Problem: Requires computing the partition function!

Loss functions

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

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

- In structure SVMs

$$\bar{\ell}_{hinge}(\mathbf{w}, x, y) = \max_{\hat{y}\in\mathcal{Y}} \left\{\ell(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y}) - \mathbf{w}^\top\Phi(x, y)\right\}$$

Loss functions

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

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

- In structure SVMs

$$\bar{\ell}_{hinge}(\mathbf{w}, x, y) = \max_{\hat{y}\in\mathcal{Y}} \left\{\ell(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y}) - \mathbf{w}^\top\Phi(x, y)\right\}$$

- CRF loss: The conditional distribution is

$$p_{x,y}(\hat{y}; \mathbf{w}) = \frac{1}{Z(x, y)} \exp\left(\ell(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y})\right)$$

$$Z(x, y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\ell(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y})\right)$$

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

$$\bar{\ell}_{log}(\mathbf{w}, x, y) = \ln \frac{1}{p_{x,y}(y; \mathbf{w})}.$$

Loss functions

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

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

- In structure SVMs

$$\bar{\ell}_{hinge}(\mathbf{w}, x, y) = \max_{\hat{y}\in\mathcal{Y}} \left\{\ell(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y}) - \mathbf{w}^\top\Phi(x, y)\right\}$$

- CRF loss: The conditional distribution is

$$p_{x,y}(\hat{y}; \mathbf{w}) = \frac{1}{Z(x, y)} \exp\left(\ell(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y})\right)$$

$$Z(x, y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\ell(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y})\right)$$

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

$$\bar{\ell}_{log}(\mathbf{w}, x, y) = \ln \frac{1}{p_{x,y}(y; \mathbf{w})}.$$

Relation between loss functions

- The CRF program is

$$(\text{CRF}) \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

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

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\ell(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

- In structured SVMs

$$(\text{structured SVM}) \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{ \ell(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y}) \right\} - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

Relation between loss functions

- The CRF program is

$$\text{(CRF)} \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

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

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\ell(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

- In structured SVMs

$$\text{(structured SVM)} \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{\ell(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right\} - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

A family of structure prediction problems

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

- One parameter extension of CRFs and structured SVMs

$$\min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z_\epsilon(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

d is the empirical means, and

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

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

A family of structure prediction problems

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

- One parameter extension of CRFs and structured SVMs

$$\min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z_\epsilon(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  $\mathbf{d}$ is the empirical means, and

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

- 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

$$\min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z_\epsilon(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

d is the empirical means, and

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

- CRF if $\epsilon = 1$, Structured SVM if $\epsilon = 0$ respectively.
- Introduces the notion of loss in CRFs.
- Dual takes the form

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

over the probability simplex over $\mathcal{Y}$.

A family of structure prediction problems

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

- One parameter extension of CRFs and structured SVMs

$$\min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z_\epsilon(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  $\mathbf{d}$ is the empirical means, and

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

- CRF if $\epsilon = 1$, Structured SVM if $\epsilon = 0$ respectively.
- Introduces the notion of loss in CRFs.
- Dual takes the form

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

  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

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

- Using this we can write the approximate program as

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

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

Primal-Dual approximated learning algorithm

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

- 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

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

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

Primal-Dual approximated learning algorithm

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

- 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

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

- 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

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

- 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

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

- 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.

Primal-Dual approximated learning algorithm

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

- 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

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

- 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.

Primal-Dual approximated learning algorithm

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

- 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

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

- 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.

Learning algorithm

Message-Passing algorithm for Approximated Structured Prediction:
Set $\bar{e}_{y,v}(\hat{y}_v) = \exp\left(e_{y,v}(\hat{y}_v)\right)$ and similarly $\bar{\phi}_{r,v}, \bar{\phi}_{r,\alpha}$.

1. For $t = 1, 2, ...$

   (a) For every $v = 1, ...n$, every $(x, y) \in \mathcal{S}$, every $\alpha \in N(v)$, every $\hat{y}_v \in \mathcal{Y}_v$ do:

$$m_{x,y,\alpha\to v}(\hat{y}_v) = \left\| \prod_{r:\alpha\in E_r} \bar{\phi}^{\theta_r}_{r,\alpha}(x, \hat{y}_\alpha) \prod_{u\in N(\alpha)\setminus v} n_{x,y,u\to\alpha}(\hat{y}_u) \right\|_{1/\epsilon c_\alpha}$$

$$n_{x,y,v\to\alpha}(\hat{y}_v) \propto \left(\bar{e}_{y,v}(\hat{y}_v) \prod_{r:v\in V_r} \bar{\phi}^{\theta_r}_{r,v}(x, \hat{y}_r) \prod_{\beta\in N(v)} m_{x,y,\beta\to v}(\hat{y}_v) \right)^{c_\alpha/\hat{c}_v} \Big/ m_{x,y,\alpha\to v}(\hat{y}_v)$$

   (b) For every $r = 1, ..., d$ do:

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

$$b_{x,y,v}(\hat{y}_v) \propto \left(\bar{e}_{y,r}(\hat{y}_v) \prod_{r:v\in V_{r,x}} \bar{\phi}^{\theta_r}_{r,v}(x, \hat{y}_v) \prod_{\alpha\in N(v)} n^{-1}_{x,y,v\to\alpha}(\hat{y}_v) \right)^{1/\epsilon c_v}$$

$$b_{x,y,\alpha}(\hat{y}_\alpha) \propto \left(\prod_{r:\alpha\in E_{r,x}} \bar{\phi}^{\theta_r}_{r,\alpha}(x, \hat{y}_\alpha) \prod_{v\in N(\alpha)} n_{x,y,v\to\alpha}(\hat{y}_v) \right)^{1/\epsilon c_\alpha}$$

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

Examples in computer vision

Examples

- Depth estimation
- Multi-label prediction
- Object detection
- Non-maxima supression
- Segmentation
- Sentence generation
- Holistic scene understanding
- 2D pose estimation
- Non-rigid shape estimation
- 3D scene understanding
- ...

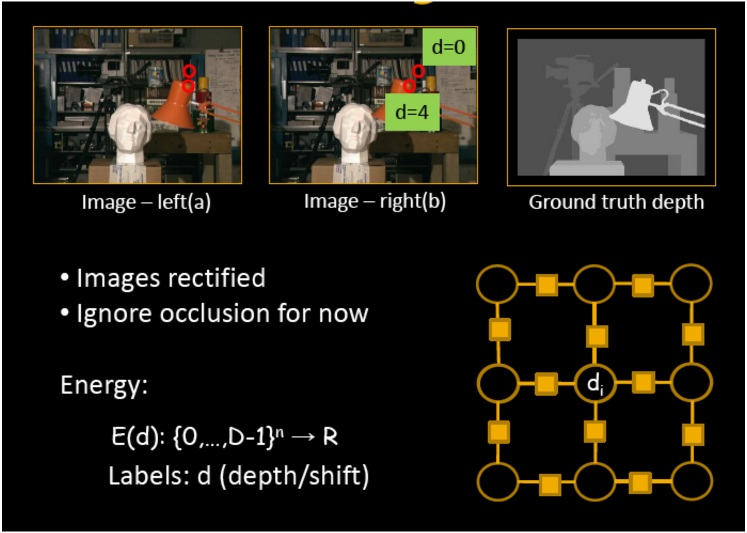
For each application ...

... 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



[Source: P. Kohli]

Stereo matching pairwise

Energy:

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

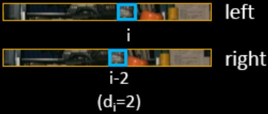
$$E(d) = \sum_i \theta_i(d_i) + \sum_{i,j \in N_4} \theta_{ij}(d_i, d_j)$$

Unary:

$$\theta_i(d_i) = (l_j - r_{i-d_i})$$

"SAD; Sum of absolute differences"

(many others possible, NCC,...)

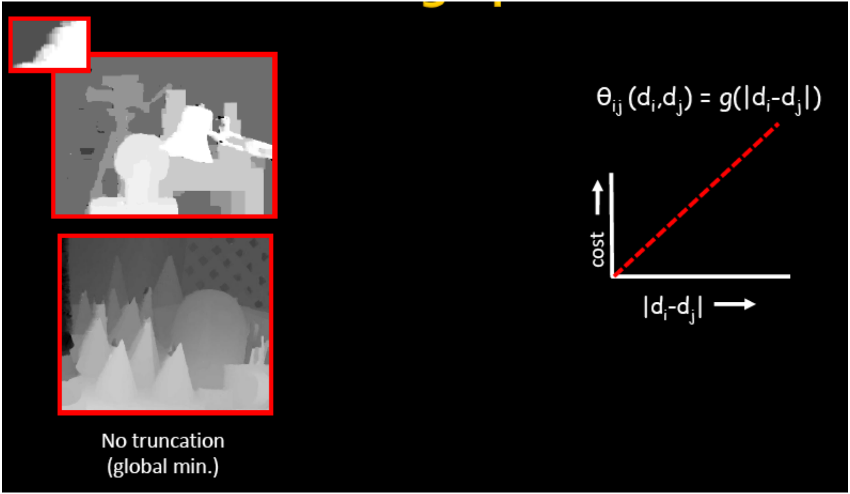


Pairwise:

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

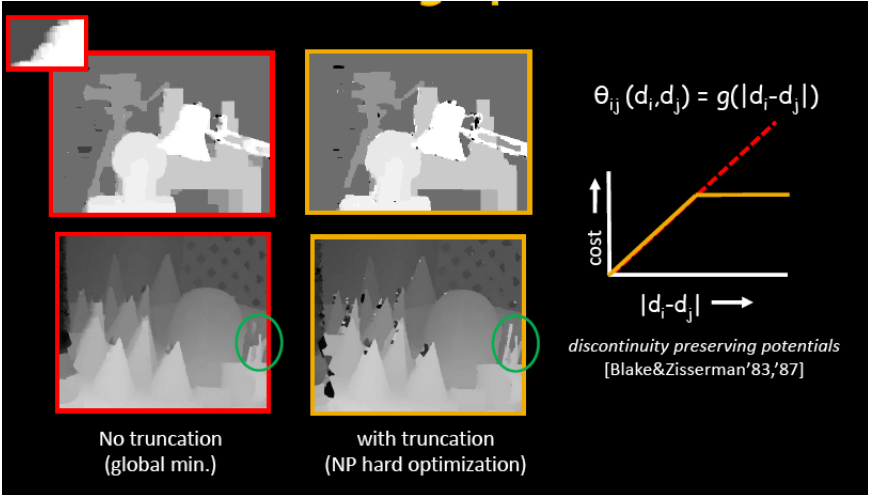
[Source: P. Kohli]

Stereo matching: energy



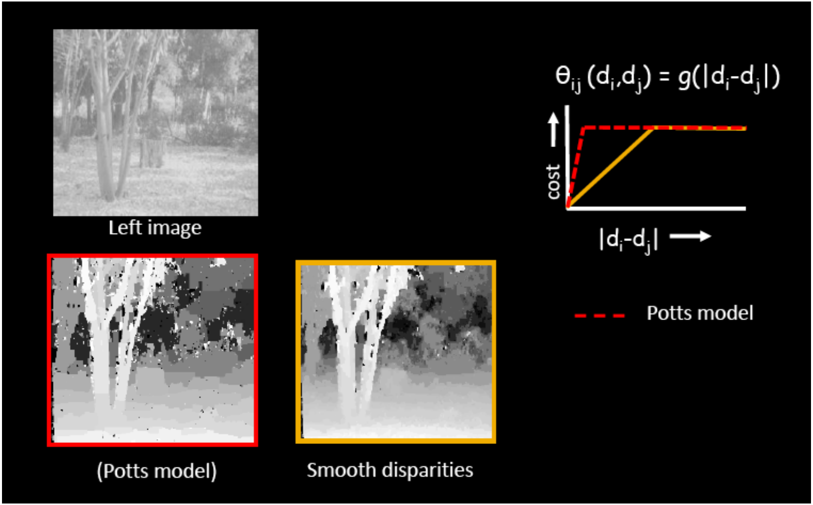
[Source: P. Kohli]

Stereo matching: energy



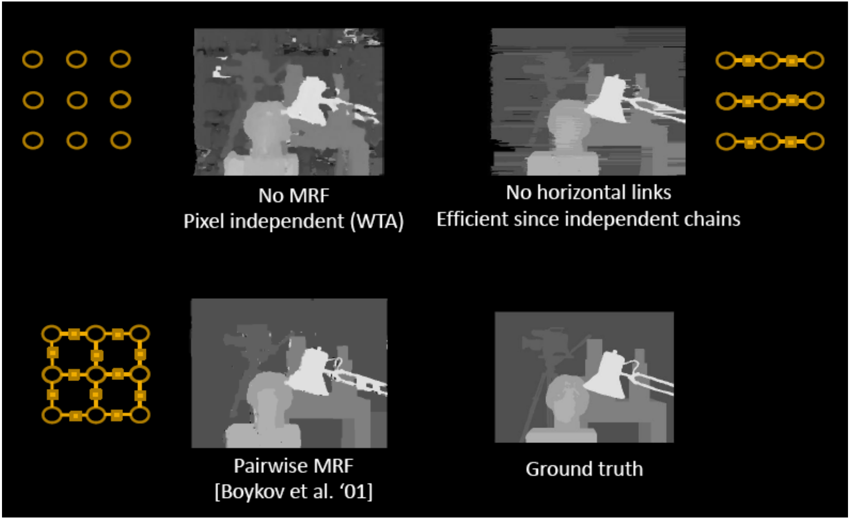
[Source: P. Kohli]

More on pairwise [O. Veksler]



[Source: P. Kohli]

Graph Structure



- see http://vision.middlebury.edu/stereo/

Learning and inference

- 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 ϵ.
- The latter is how typically stereo algorithms are scored
- Which inference method will you choose?
- And for learning?

Example: Object Detection

- 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

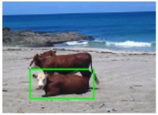
- Note: $x|_y$ (the image restricted to the box region) is again an image.
- Compare two images with boxes by comparing the images within the boxes:

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

- Any common image kernel is applicable:
  - linear on cluster histograms: $k(h,h') = \sum_i h_i h'_i$,
  - χ^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]

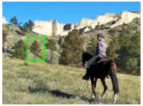
Restriction Kernel example

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$

is large.

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$

is small.

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$

could also be large.

- Note: This behaves differently from the common tensor products

$$k_{joint}(\ (x, y), (x', y')\ ) \neq k(x, x')k(y, y'))\ !$$

[Source: M. Blascko]

Margin Rescaling

$$\langle w, \varphi(x_i, y_i)\rangle - \langle w, \varphi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i, \ \forall i, \forall y \in \mathcal{Y} \setminus y_i$$

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

$$\Delta(y_i, y) = 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)}$$

[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

$$\Delta(y_i, y) = 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)}$$

- Solution: use branch-and-bound over the space of all rectangles in the image (Blaschko & Lampert, 2008)

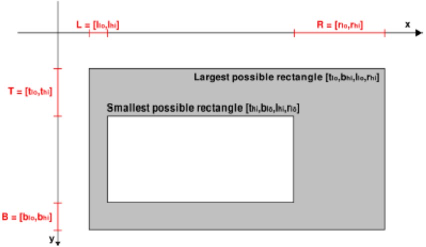
[Source: M. Blascko]

Sets of Rectangles

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]

Optimization

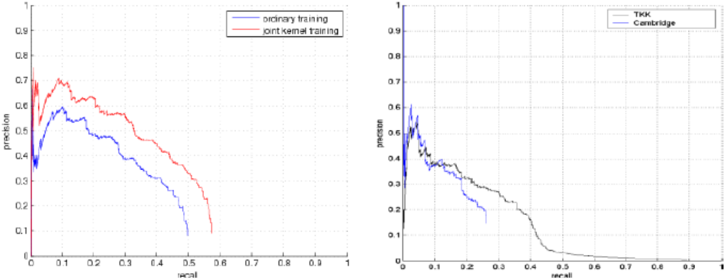
- Train using constraint generation
  - Train an SVM with margin rescaling
  - Identify the most violated constraint with branch and bound and add it to the constraint set

$$\max_{y \in \mathcal{Y} \setminus y_i} \overbrace{\sum_{j=1}^{n} \sum_{\tilde{y} \in \mathcal{Y}} \alpha_{j\tilde{y}} \left(k_x(x_j|_{y_j}, x_i|_y) - k_x(x_j|_{\tilde{y}}, x_i|_y) \right)}^{\text{Lampert et al., PAMI 2009}} + \underbrace{\Delta(y_i, y)}_{\text{upper bound this term}}$$

- iterate until convergence criterion is reached

[Source: M. Blascko]

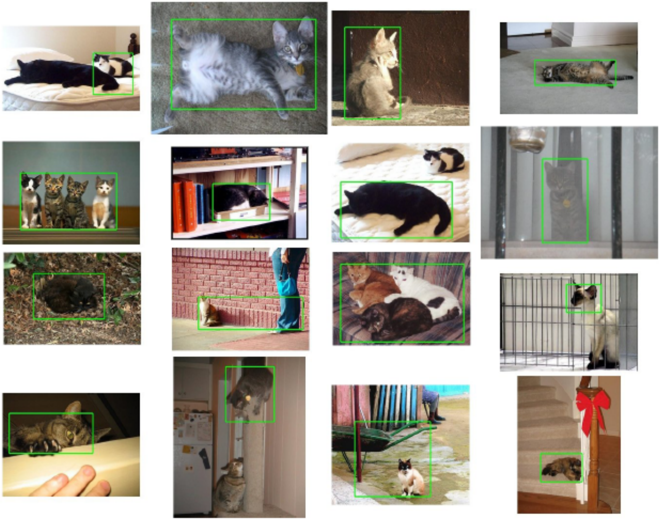
- ≈5,000 images: ≈2,500 train/val, ≈2,500 test
- ≈9,500 objects in 10 predefined classes:
  - `bicycle, bus, car, cat, cow, dog, horse, motorbike, person, sheep`
- Task: predict locations and confidence scores for each class
- Evaluation: Precision-Recall curves



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:

$$k_{\text{global}}(\ (x_i, y_i),\ (x_j, y_j)\ ) = k_I(x_i, x_j)$$

- Local context kernel:

$$k_{\text{local}}(\ (x_i, y_i),\ (x_j, y_j); \theta) = k_I(x_i|_{\Theta(y_i)}, x_j|_{\Theta(y_j)})$$

- Putting it all together:

$$\begin{aligned} k(\ (x_i, y_i),\ (x_j, y_j)\ ) &= \beta_1 k_{\text{restr}}(\ (x_i, y_i),\ (x_j, y_j)\ ) \\ &+ \beta_2 k_{\text{local}}(\ (x_i, y_i),\ (x_j, y_j); \theta) \\ &+ \beta_3 k_{\text{global}}(\ (x_i, y_i),\ (x_j, y_j)\ ) \end{aligned}$$

- β can be learned using multiple kernel learning   Blaschko & Lampert, 2009

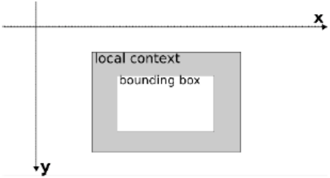
[Source: M. Blascko]

Local Context Kernel

- 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



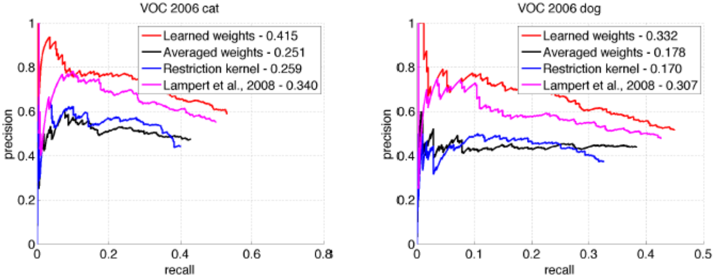
- Model the statistics of an object's neighborhood
- Don't model the statistics of the object itself

[Source: M. Blascko]

Results

Context is a very busy area of research in vision!

	bicycle	*bus*	*car*	*cat*	*dog*	*cow*
learned	0.410	**0.253**	**0.268**	**0.415**	**0.332**	**0.286**
fixed	**0.429**	0.177	0.263	0.251	0.178	0.194
no context	0.396	0.100	0.145	0.259	0.170	0.118



[Source: M. Blascko]

Example: 3D Indoor Scene Understanding

- **Task**: Given an image, predict the 3D parametric cuboid that best describes the layout.

Prediction

Variables are not independent of each other, i.e. structured prediction

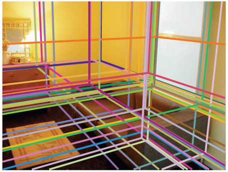
- $\mathbf{x}$: Input image
- $\mathbf{y}$: Room layout
- $\phi(\mathbf{x}, \mathbf{y})$: Multidimensional feature vector
- $\mathbf{w}$: Predictor
- Estimate room layout by solving inference task

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$$

- Learning $\mathbf{w}$ via structured SVMs or CRFs

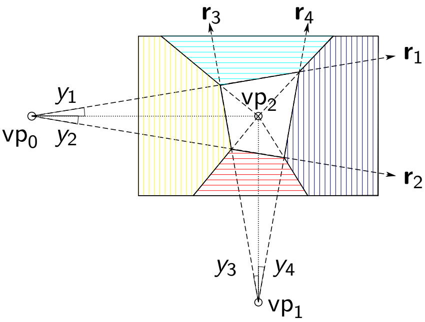
Single Variable Parameterization

- 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

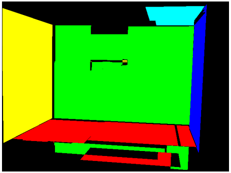
- Approach of [Wang et al. 10].
- 4 variables $y_i \in \mathcal{Y}$, $i \in \{1, \ldots, 4\}$ corresponding to the four degrees of freedom of the problem.
- One state of y_i denotes the angle of ray $\mathbf{r}_i$.
- High order potentials, e.g., 50^4 for fourth-order.



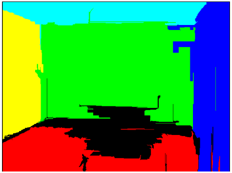
For both parameterizations is even worst when reasoning about objects.

Integral Geometry for Features

- 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.



orientation map



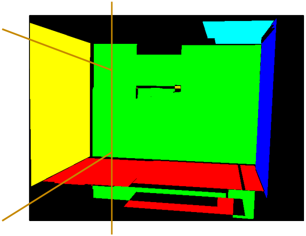
geometric context

Integral Geometry for Features

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

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

- Features count frequencies of image cues



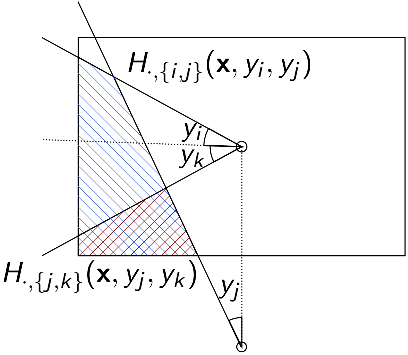
Orientation map and proposed left wall

Integral Geometry for Features

- Using inspiration from integral images, we decompose

$$\begin{aligned}\phi_{\cdot,\alpha}(\mathbf{x}, \mathbf{y}_\alpha) &= \phi_{\cdot,\{i,j,k\}}(\mathbf{x}, y_i, y_j, y_k) = \\ &= H_{\cdot,\{i,j\}}(\mathbf{x}, y_i, y_j) - H_{\cdot,\{j,k\}}(\mathbf{x}, y_j, y_k)\end{aligned}$$

- Integral geometry

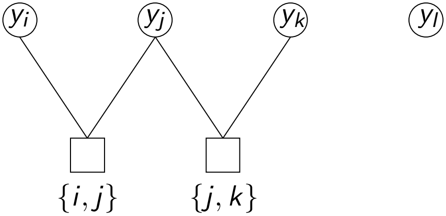


Integral Geometry for Features

- Decomposition:

$$H_{\cdot,\{i,j\}}(\mathbf{x}, y_i, y_j) - H_{\cdot,\{j,k\}}(\mathbf{x}, y_j, y_k)$$

- Corresponding factor graph:



- The front-wall:

$$\phi_{\cdot,\textit{front-wall}} = \phi(\mathbf{x}) - \phi_{\cdot,\textit{left-wall}} - \phi_{\cdot,\textit{right-wall}} - \phi_{\cdot,\textit{ceiling}} - \phi_{\cdot,\textit{floor}}$$

Integral Geometry

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

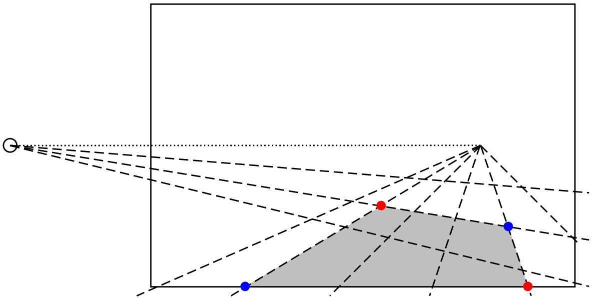


Figure: Concept of integral geometry

Learning and Inference

Learning

- Family of structure prediction problems including CRF and structured-SVMs as especial 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]

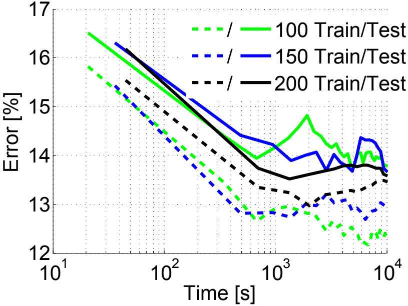
Inference

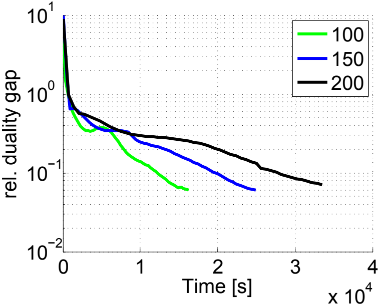
- Inference using parallel convex belief propagation
- Convergence and other theoretical guarantees
- **Code available online**: general potentials, cross-platform, Amazon EC2!

[A. Schwing, T. Hazan, M. Pollefeys and R. Urtasun, CVPR 2011]

Time vs Accuracy

Learning very fast: State-of-the-art after less than a minute!





Inference as little as 10ms per image!

Results

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

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

	OM	GC	OM + GC
[Hoiem07]	-	28.9	-
[Hedau09] (a)	-	26.5	-
[Hedau09] (b)	-	21.2	-
[Wang10]	22.2	-	-
[Lee10]	24.7	22.7	18.6
Ours (SVM$^{\text{struct}}$)	**19.5**	**18.2**	**16.8**
Ours (struct-pred)	**18.6**	**15.4**	**13.6**

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

	[Luca11]	[Hoiem07]	[Hedau09](a)	Ours
w/o box	29.59	23.04	22.94	**16.46**

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].

	OM	GC	OM + GC
[Hoiem07]	-	28.9	-
[Hedau09] (a)	-	26.5	-
[Hedau09] (b)	-	21.2	-
[Wang10]	22.2	-	-
[Lee10]	24.7	22.7	18.6
Ours (SVM$^{\text{struct}}$)	**19.5**	**18.2**	**16.8**
Ours (struct-pred)	**18.6**	**15.4**	**13.6**

Table: WITH object reasoning.

	OM	GC	OM + GC
[Wang10]	20.1	-	-
[Lee10]	19.5	20.2	16.2
Ours (SVM$^{\text{struct}}$)	**18.5**	**17.7**	16.4
Ours (struct-pred)	**17.1**	**14.2**	**12.8**

Results

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

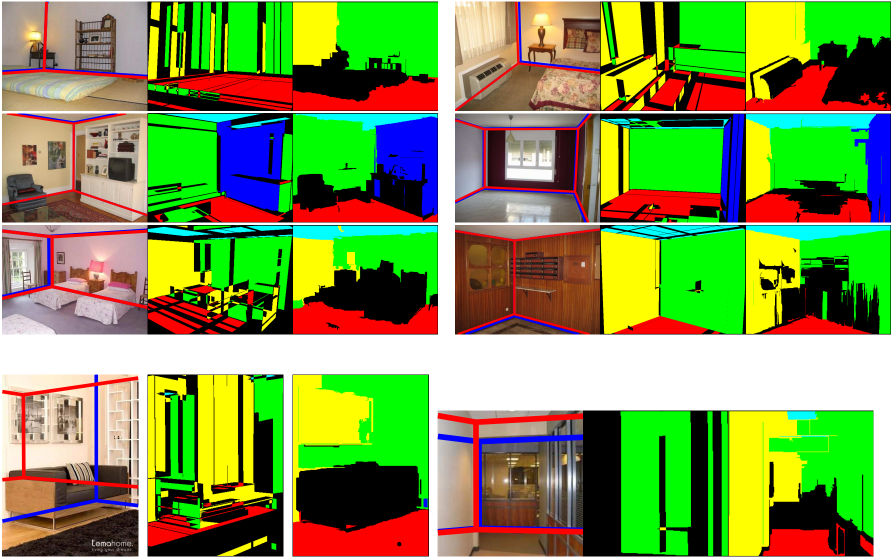
Table: Pixel classification error in the layout dataset of [Hedau et al. 09] with object reasoning.

	OM	GC	OM + GC
[Wang10]	20.1	-	-
[Lee10]	19.5	20.2	16.2
Ours (SVM^{struct})	**18.5**	**17.7**	16.4
Ours (struct-pred)	**17.1**	**14.2**	**12.8**

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

	[Luca11]	[Hoiem07]	[Hedau09](a)	Ours
w/o box	29.59	23.04	22.94	**16.46**
w/ box	26.79	-	22.94	**15.19**

Qualitative Results



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?