Inference and Representation

David Sontag

New York University

Lecture 11, Nov. 25, 2014

Conditional random fields (CRFs)

- **Conditional random fields** are undirected graphical models of conditional distributions $p(\mathbf{Y} \mid \mathbf{X})$
  - $\mathbf{Y}$ is a set of **target variables**
  - $\mathbf{X}$ is a set of **observed variables**
- We typically show the graphical model using just the $\mathbf{Y}$ variables
- Potentials are a function of $\mathbf{X}$ and $\mathbf{Y}$

Formal definition

- A CRF is a Markov network on variables $\mathbf{X} \cup \mathbf{Y}$, which specifies the conditional distribution

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$$

  with partition function

$$Z(\mathbf{x}) = \sum_{\hat{\mathbf{y}}} \prod_{c \in C} \phi_c(\mathbf{x}_c, \hat{\mathbf{y}}_c).$$

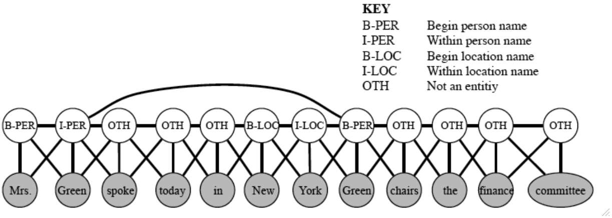
- As before, two variables in the graph are connected with an undirected edge if they appear together in the scope of some factor
- The only difference with a standard Markov network is the normalization term – before marginalized over $\mathbf{X}$ and $\mathbf{Y}$, now only over $\mathbf{Y}$

Application: named-entity recognition

- Given a sentence, determine the people and organizations involved and the relevant locations:
  "Mrs. Green spoke today in New York. Green chairs the finance committee."
- Entities sometimes span multiple words. Entity of a word not obvious without considering its *context*
- CRF has one variable X_i for each word, which encodes the possible labels of that word
- The labels are, for example, "B-person, I-person, B-location, I-location, B-organization, I-organization"
  - Having beginning (B) and within (I) allows the model to segment adjacent entities

Application: named-entity recognition

The graphical model looks like (called a *skip-chain CRF*):



There are three types of potentials:

- $\phi^1(Y_t, Y_{t+1})$ represents dependencies between neighboring target variables [analogous to transition distribution in a HMM]
- $\phi^2(Y_t, Y_{t'})$ for all pairs t, t' such that $x_t = x_{t'}$, because if a word appears twice, it is likely to be the same entity
- $\phi^3(Y_t, X_1, \cdots, X_T)$ for dependencies between an entity and the word sequence [e.g., may have features taking into consideration capitalization]

Notice that the graph structure changes depending on the sentence!

Application: Part-of-speech tagging

United flies some large jet

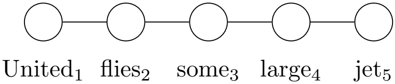
$$\begin{tikzcd}
\mathrm{N} \arrow[r] \arrow[d] & \mathrm{V} \arrow[r] \arrow[d] & \mathrm{D} \arrow[r] \arrow[d] & \mathrm{A} \arrow[r] \arrow[d] & \mathrm{N} \arrow[d] \\
\text{United}_1 & \text{flies}_2 & \text{some}_3 & \text{large}_4 & \text{jet}_5
\end{tikzcd}$$

Graphical model formulation of POS tagging

given:

- a sentence of length n and a tag set $\mathcal{T}$
- one variable for each word, takes values in $\mathcal{T}$
- edge potentials $\theta(i-1, i, t', t)$ for all $i \in n,\ t, t' \in \mathcal{T}$

example:



$$\mathcal{T} = \{A, D, N, V\}$$

Features for POS tagging

- Parameterization as log-linear model:
  - Weights $\mathbf{w} \in \mathbb{R}^d$. Feature vectors $\mathbf{f}_c(\mathbf{x}, \mathbf{y}_c) \in \mathbb{R}^d$.
  - $\phi_c(\mathbf{x}, \mathbf{y}_c; \mathbf{w}) = \exp(\mathbf{w} \cdot \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c))$
- Edge potentials: Fully parameterize ($\mathcal{T} \times \mathcal{T}$ features and weights), i.e.

$$\theta_{i-1,i}(t', t) = w^T_{t',t}$$

  where the superscript "T" denotes that these are the weights for the transitions

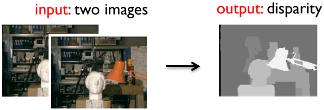
- Node potentials: Introduce features for the presence or absence of certain attributes of each word (e.g., initial letter capitalized, suffix is "ing"), for each possible tag ($\mathcal{T}\times$ #attributes features and weights)

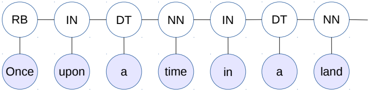
  This part is conditional on the input sentence!

- Edge potential same for all edges. Same for node potentials.

Density estimation for CRFs

- Suppose we want to predict a set of variables **Y** given some others **X**, e.g., stereo vision or part-of-speech tagging:





- We concentrate on predicting $p(\mathbf{Y}|\mathbf{X})$, and use a **conditional** loss function

$$loss(\mathbf{x}, \mathbf{y}, \hat{\mathcal{M}}) = -\log \hat{p}(\mathbf{y} \mid \mathbf{x}).$$

- Since the loss function only depends on $\hat{p}(\mathbf{y} \mid \mathbf{x})$, suffices to estimate the conditional distribution, not the joint

Density estimation for CRFs

$$\text{CRF:} \quad p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \phi_c(\mathbf{x}, \mathbf{y}_c), \quad Z(\mathbf{x}) = \sum_{\hat{\mathbf{y}}} \prod_{c \in C} \phi_c(\mathbf{x}, \hat{\mathbf{y}}_c)$$

- Empirical risk minimization with CRFs, i.e. $\min_{\hat{\mathcal{M}}} \mathbf{E}_{\mathcal{D}} \left[loss(\mathbf{x}, \mathbf{y}, \hat{\mathcal{M}}) \right]$:

$$\begin{aligned}
\mathbf{w}^{ML} &= \arg\min_{\mathbf{w}} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} -\log p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \left(\sum_c \log \phi_c(\mathbf{x}, \mathbf{y}_c; \mathbf{w}) - \log Z(\mathbf{x}; \mathbf{w}) \right) \\
&= \arg\max_{\mathbf{w}} \mathbf{w} \cdot \left(\sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \sum_c \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c) \right) - \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \log Z(\mathbf{x}; \mathbf{w})
\end{aligned}$$

- What if prediction is only done with MAP inference? Then, the partition function is irrelevant. Is there a way to train to take advantage of this?

Goal of learning

- The goal of learning is to return a model $\hat{\mathcal{M}}$ that precisely captures the distribution p^* from which our data was sampled
- This is in general not achievable because of
  - computational reasons
  - limited data only provides a rough approximation of the true underlying distribution
- We need to select $\hat{\mathcal{M}}$ to construct the "best" approximation to $\mathcal{M}^*$
- What is "best"?

What notion of "best" should learning be optimizing?

This depends on what we want to do

1. Density estimation: we are interested in the full distribution (so later we can compute whatever conditional probabilities we want)
2. Specific prediction tasks: we are using the distribution to make a prediction
3. Structure or knowledge discovery: we are interested in the model itself

Structured prediction

- Often we learn a model for the purpose of **structured prediction**, in which given $\mathbf{x}$ we predict $\mathbf{y}$ by finding the MAP assignment:

$$\operatorname*{argmax}_{\mathbf{y}} \hat{p}(\mathbf{y}|\mathbf{x})$$

- Rather than learn using log-loss (density estimation), we use a loss function better suited to the specific task
- One reasonable choice would be the **classification error**:

$$\mathbf{E}_{(\mathbf{x},\mathbf{y})\sim p^*} [\mathbb{1}\{\ \exists \mathbf{y}' \neq \mathbf{y} \text{ s.t. } \hat{p}(\mathbf{y}'|\mathbf{x}) \geq \hat{p}(\mathbf{y}|\mathbf{x})\ \}]$$

  which is the probability over all $(\mathbf{x}, \mathbf{y})$ pairs sampled from p^* that our classifier selects the right labels
- If p^* is in the model family, training with log-loss (density estimation) and classification error would perform similarly (given sufficient data)
- Otherwise, better to directly go for what we care about (classification error)

Structured prediction

- Consider the empirical risk for 0-1 loss (classification error):

$$\frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbb{1}\{\ \exists \mathbf{y}' \neq \mathbf{y} \text{ s.t. } \hat{p}(\mathbf{y}'|\mathbf{x}) \geq \hat{p}(\mathbf{y}|\mathbf{x})\ \}$$

- Each constraint $\hat{p}(\mathbf{y}'|\mathbf{x}) \geq \hat{p}(\mathbf{y}|\mathbf{x})$ is equivalent to

$$\mathbf{w}\cdot\sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}'_c) - \log Z(\mathbf{x};\mathbf{w}) \geq \mathbf{w}\cdot\sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}_c) - \log Z(\mathbf{x};\mathbf{w})$$

- The log-partition function cancels out on both sides. Re-arranging, we have:

$$\mathbf{w}\cdot\left(\sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}'_c) - \sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}_c)\right) \geq 0$$

- Said differently, the empirical risk is **zero** when $\forall(\mathbf{x},\mathbf{y})\in\mathcal{D}$ and $\mathbf{y}'\neq\mathbf{y}$,

$$\mathbf{w}\cdot\left(\sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}_c) - \sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}'_c)\right) > 0.$$

Structured prediction

- Empirical risk is **zero** when $\forall(\mathbf{x},\mathbf{y}) \in \mathcal{D}$ and $\mathbf{y}' \neq \mathbf{y}$,

$$\mathbf{w} \cdot \left(\sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}_c) - \sum_c \mathbf{f}_c(\mathbf{x},\mathbf{y}'_c) \right) > 0.$$

- In the simplest setting, learning corresponds to finding a weight vector **w** that satisfies all of these constraints (when possible)
- This is a linear program (LP)!
- How many constraints does it have? $|\mathcal{D}| * |\mathcal{Y}|$ – exponentially many!
- Thus, we must avoid explicitly representing this LP
- This lecture is about algorithms for solving this LP (or some variant) in a tractable manner

Structured *perceptron* algorithm

- **Input:** Training examples $\mathcal{D} = \{(\mathbf{x}^m, \mathbf{y}^m)\}$
- Let $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_c \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c)$. Then, the constraints that we want to satisfy are

$$\mathbf{w} \cdot \Big(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})\Big) > 0, \quad \forall \mathbf{y} \neq \mathbf{y}^m$$

- The perceptron algorithm uses MAP inference in its inner loop:

$$\mathrm{MAP}(\mathbf{x}^m; \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^m, \mathbf{y})$$

  The maximization can often be performed efficiently by using the structure!

- The perceptron algorithm is then:
  1. Start with $\mathbf{w} = 0$
  2. While the weight vector is still changing:
  3.   For $m = 1, \ldots, |\mathcal{D}|$
  4. $\mathbf{y} \leftarrow \mathrm{MAP}(\mathbf{x}^m; \mathbf{w})$
  5. $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})$

Structured perceptron algorithm

- If the training data is *separable*, the perceptron algorithm is guaranteed to find a weight vector which perfectly classifies all of the data
- When separable with margin γ, number of iterations is at most

$$\left(\frac{2R}{\gamma}\right)^2,$$

  where $R = \max_{m,\mathbf{y}} ||\mathbf{f}(\mathbf{x}^m, \mathbf{y})||_2$
- In practice, one stops after a certain number of outer iterations (called *epochs*), and uses the *average* of all weights
- The averaging can be understood as a type of regularization to prevent overfitting

Allowing slack

- We can equivalently write the constraints as

$$\mathbf{w} \cdot \Big(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})\Big) \geq 1, \quad \forall \mathbf{y} \neq \mathbf{y}^m$$

- Suppose there do not exist weights **w** that satisfy all constraints
- Introduce *slack* variables $\xi_m \geq 0$, one per data point, to allow for constraint violations:

$$\mathbf{w} \cdot \Big(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})\Big) \geq 1 - \xi_m, \quad \forall \mathbf{y} \neq \mathbf{y}^m$$

- Then, minimize the sum of the slack variables, $\min_{\xi \geq 0} \sum_m \xi_m$, subject to the above constraints

Structural SVM (support vector machine)

$$\min_{\mathbf{w},\xi} \quad \sum_m \xi_m + C||\mathbf{w}||^2$$

subject to:

$$\begin{aligned} \mathbf{w} \cdot \Big(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})\Big) &\geq 1 - \xi_m, \quad \forall m, \mathbf{y} \neq \mathbf{y}^m \\ \xi_m &\geq 0, \quad \forall m \end{aligned}$$

This is a quadratic program (QP). Solving for the slack variables in closed form, we obtain

$$\xi_m^* = \max\left(0, \; \max_{\mathbf{y} \in \mathcal{Y}} 1 - \mathbf{w} \cdot \Big(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})\Big)\right)$$

Thus, we can re-write the whole optimization problem as

$$\min_{\mathbf{w}} \quad \sum_m \max\left(0, \; \max_{\mathbf{y} \in \mathcal{Y}} 1 - \mathbf{w} \cdot \Big(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})\Big)\right) + C||\mathbf{w}||^2$$

Hinge loss

- We can view $\max\Big(0,\ \max_{\mathbf{y}\in\mathcal{Y}}\ 1 - \mathbf{w}\cdot\Big(\mathbf{f}(\mathbf{x}^m,\mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m,\mathbf{y})\Big)\Big)$ as a loss function, called *hinge loss*
- When $\mathbf{w}\cdot\mathbf{f}(\mathbf{x}^m,\mathbf{y}^m) \geq \mathbf{w}\cdot\mathbf{f}(\mathbf{x}^m,\mathbf{y})$ for all $\mathbf{y}$ (i.e., correct prediction), this takes a value between 0 and 1
- When $\exists\mathbf{y}$ such that $\mathbf{w}\cdot\mathbf{f}(\mathbf{x}^m,\mathbf{y}) \geq \mathbf{w}\cdot\mathbf{f}(\mathbf{x}^m,\mathbf{y}^m)$ (i.e., incorrect prediction), this takes a value ≥ 1
- Thus, this always *upper bounds* the 0-1 loss!
- Minimizing hinge loss is good because it minimizes an upper bound on the 0-1 loss (prediction error)

Better Metrics

- It doesn't always make sense to penalize all incorrect predictions equally!
- We can change the constraints to

$$\mathbf{w} \cdot \Big(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y})\Big) \geq \Delta(\mathbf{y}, \mathbf{y}^m) - \xi_m, \quad \forall \mathbf{y},$$

  where $\Delta(\mathbf{y}, \mathbf{y}^m) \geq 0$ is a measure of how far the assignment $\mathbf{y}$ is from the true assignment $\mathbf{y}^m$
- This is called **margin scaling** (as opposed to *slack scaling*)
- We assume that $\Delta(\mathbf{y}, \mathbf{y}) = 0$, which allows us to say that the constraint holds for *all* $\mathbf{y}$, rather than just $\mathbf{y} \neq \mathbf{y}^m$
- A frequently used metric for MRFs is **Hamming distance**, where $\Delta(\mathbf{y}, \mathbf{y}^m) = \sum_{i \in V} \mathbb{1}[y_i \neq y_i^m]$

Structural SVM with margin scaling

$$\min_{\mathbf{w}} \sum_m \max_{\mathbf{y}\in\mathcal{Y}} \Big(\Delta(\mathbf{y},\mathbf{y}^m) - \mathbf{w}\cdot\big(\mathbf{f}(\mathbf{x}^m,\mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m,\mathbf{y})\big)\Big) + C||\mathbf{w}||^2$$

How to solve this? Many methods!

1. Cutting-plane algorithm (Tsochantaridis et al., 2005)
2. Stochastic subgradient method (Ratliff et al., 2007)
3. Dual Loss Primal Weights algorithm (Meshi et al., 2010)
4. Frank-Wolfe algorithm (Lacoste-Julien et al., 2013)

Stochastic subgradient method

$$\min_{\mathbf{w}} \sum_m \max_{\mathbf{y} \in \mathcal{Y}} \left(\Delta(\mathbf{y}, \mathbf{y}^m) - \mathbf{w} \cdot \left(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y}) \right) \right) + C||\mathbf{w}||^2$$

- Although this objective is convex, it is not differentiable everywhere
- We can use a *subgradient* method to minimize (instead of gradient descent)
- The *subgradient* of $\max_{\mathbf{y} \in \mathcal{Y}} \ \Delta(\mathbf{y}, \mathbf{y}^m) - \mathbf{w} \cdot \left(\mathbf{f}(\mathbf{x}^m, \mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m, \mathbf{y}) \right)$ at $\mathbf{w}^{(t)}$is

  $$\mathbf{f}(\mathbf{x}^m, \hat{\mathbf{y}}) - \mathbf{f}(\mathbf{x}^m, \mathbf{y}^m),$$

  where $\hat{\mathbf{y}}$ is one of the maximizers with respect to $\mathbf{w}^{(t)}$, i.e.

  $$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} \ \Delta(\mathbf{y}, \mathbf{y}^m) + \mathbf{w}^{(t)} \cdot \mathbf{f}(\mathbf{x}^m, \mathbf{y})$$

- This maximization is called *loss-augmented* MAP inference

Loss-augmented inference

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; \Delta(\mathbf{y}, \mathbf{y}^m) + \mathbf{w}^{(t)} \cdot \mathbf{f}(\mathbf{x}^m, \mathbf{y})$$

- When $\Delta(\mathbf{y}, \mathbf{y}^m) = \sum_{i \in V} \mathbb{1}[y_i \neq y_i^m]$, this corresponds to adding additional single-node potentials

$$\theta_i(y_i) = 1 \text{ if } y_i \neq y_i^m, \text{ and } 0 \text{ otherwise}$$

- If MAP inference was previously exactly solvable by a combinatorial algorithm, loss-augmented MAP inference typically is too
- The Hamming distance pushes the MAP solution *away* from the true assignment $\mathbf{y}^m$

Cutting-plane algorithm

$$\min_{\mathbf{w},\xi} \sum_m \xi_m + C||\mathbf{w}||^2$$

subject to:

$$\begin{aligned} \mathbf{w}\cdot\Big(\mathbf{f}(\mathbf{x}^m,\mathbf{y}^m) - \mathbf{f}(\mathbf{x}^m,\mathbf{y})\Big) &\geq \Delta(\mathbf{y},\mathbf{y}^m) - \xi_m, \quad \forall m, \mathbf{y}\in\mathcal{Y}_m \\ \xi_m &\geq 0, \quad \forall m \end{aligned}$$

- Start with $\mathcal{Y}_m = \{\mathbf{y}^m\}$. Solve for the optimal $\mathbf{w}^*, \xi^*$
- Then, look to see if any of the *unused* constraints are violated
- To find a violated constraint for data point m, simply solve the loss-augmented inference problem:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \Delta(\mathbf{y},\mathbf{y}^m) + \mathbf{w}\cdot\mathbf{f}(\mathbf{x}^m,\mathbf{y})$$

- If $\hat{\mathbf{y}} \in \mathcal{Y}_m$, do nothing. Otherwise, let $\mathcal{Y}_m = \mathcal{Y}_m \cup \{\hat{\mathbf{y}}\}$
- Repeat until no new constraints are added. Then we are optimal!

Cutting-plane algorithm

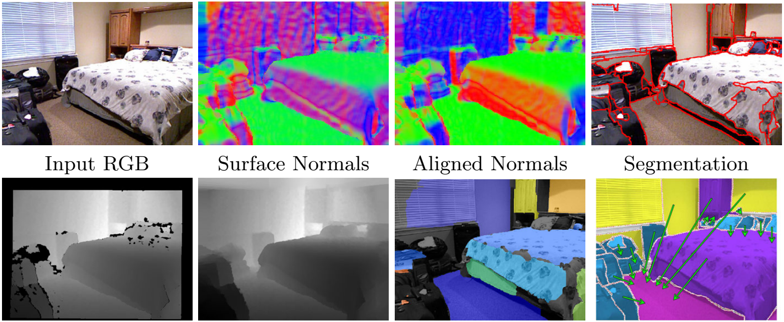
- Can prove that, in order to solve the structural SVM up to ϵ (additive) accuracy, takes a polynomial number of iterations
- In practice, terminates very quickly

Summary of convergence rates

Optimization algorithm	Online	Primal/Dual	Type of guarantee	Oracle type	# Oracle calls
dual extragradient (Taskar et al., 2006)	no	primal-'dual'	saddle point gap	Bregman projection	$O\left(\frac{nR\log\lvert\mathcal{Y}\rvert}{\lambda\varepsilon}\right)$
online exponentiated gradient (Collins et al., 2008)	yes	dual	expected dual error	expectation	$O\left(\frac{(n+\log\lvert\mathcal{Y}\rvert)R^2}{\lambda\varepsilon}\right)$
excessive gap reduction (Zhang et al., 2011)	no	primal-dual	duality gap	expectation	$O\left(nR\sqrt{\frac{\log\lvert\mathcal{Y}\rvert}{\lambda\varepsilon}}\right)$
BMRM (Teo et al., 2010)	no	primal	$\geq$primal error	maximization	$O\left(\frac{nR^2}{\lambda\varepsilon}\right)$
1-slack SVM-Struct (Joachims et al., 2009)	no	primal-dual	duality gap	maximization	$O\left(\frac{nR^2}{\lambda\varepsilon}\right)$
stochastic subgradient (Shalev-Shwartz et al., 2010a)	yes	primal	primal error w.h.p.	maximization	$\tilde{O}\left(\frac{R^2}{\lambda\varepsilon}\right)$
this paper: block-coordinate Frank-Wolfe	yes	primal-dual	expected duality gap	maximization	$O\left(\frac{R^2}{\lambda\varepsilon}\right)$ Thm. 3

R same as before. n=number of training examples. λ is the regularization constant (correpsonding to $2C/n$)

Application to segmentation & support inference



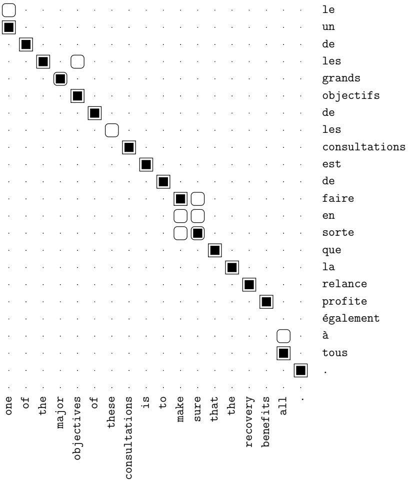
Input RGB | Surface Normals | Aligned Normals | Segmentation

Input Depth | Inpainted Depth | 3D Planes | Support Relations

(Silberman, Sontag, Fergus. ECCV '14)

Application to machine translation

Word alignment between languages:



(Taskar, Lacoste-Julien, Klein. EMNLP '05)