

Language and Statistics II

Lecture 16: Going Discriminative

Noah Smith

October 19, 2009

Overview

- Maximizing the margin: nonseparable data
- Loss-augmented decoding
- Training large margin models

Quick review

- Motivation: only model/discriminate what is necessary *for the task*
- Perceptron: find *some* linear separator between $\langle \mathbf{x}, \mathbf{y} \rangle$ and all $\{\langle \mathbf{x}, \mathbf{y}' \rangle : \mathbf{y}' \in \mathcal{Y}\}$
- Exp-loss and boosting: bound ranking loss
- Log-loss: MLE of conditional log-linear model $p(\mathbf{y} \mid \mathbf{x})$
- Large margin, arbitrary error function: QP with too many constraints

Multiclass SVMs

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{\mathbf{w}^\top \mathbf{w}}{2} \\ \text{s.t.} \quad & \forall i, \forall \mathbf{y} \in \mathcal{Y}, \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \geq e(\mathbf{y}; \tilde{\mathbf{y}}_i) \end{aligned}$$

Intuition: find weights \mathbf{w} to make each wrong \mathbf{y} as far away (margin) as it is bad (error).

Note that the objective is quadratic, and the constraints are linear.

For structured models, \mathcal{Y} can be exponentially large.

Slack for Nonseparability

“Cut the constraints some slack.”

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{\mathbf{w}^\top \mathbf{w}}{2} + \boxed{\frac{1}{C} \sum_i \xi_i} \\ \text{s.t.} \quad & \forall i, \forall \mathbf{y} \in \mathcal{Y}, \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \geq e(\mathbf{y}; \tilde{\mathbf{y}}_i) - \boxed{\xi_i} \end{aligned}$$

It's okay to decrease $e(\mathbf{y}; \tilde{\mathbf{y}}_i)$ to solve the problem, but you pay for doing so. C controls how much you pay.

This is called “margin rescaling,” and it is due to Taskar, Guestrin, and Koller (2003).

Alternative: “Slack Rescaling”

Tsochantaridis, Joachims, Hofmann, Altun (2005).

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

$$\text{s.t.} \quad \forall i, \forall \mathbf{y} \in \mathcal{Y}, \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \geq 1 - \frac{\xi_i}{e(\mathbf{y}; \tilde{\mathbf{y}}_i)}$$

(Computationally more difficult, usually.)

Alternative: No Error, No Slack

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{\mathbf{w}^\top \mathbf{w}}{2} \\ \text{s.t.} \quad & \forall i, \forall \mathbf{y} \in \mathcal{Y}, \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \geq 0 \end{aligned}$$

Similar to perceptron (except for the objective function).

Solving for ξ_i

$\forall i, \forall \mathbf{y} \in \mathcal{Y},$

$$\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \geq e(\mathbf{y}; \tilde{\mathbf{y}}_i) - \xi_i$$

$$\xi_i \geq e(\mathbf{y}; \tilde{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) + \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y})$$

$$\forall i, \quad \xi_i = \max_{\mathbf{y} \in \mathcal{Y}} \left(e(\mathbf{y}; \tilde{\mathbf{y}}_i) + \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \right) - \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)$$

Min-Max Formulation

$$\min_{\mathbf{w}} C \frac{\mathbf{w}^\top \mathbf{w}}{2} - \left(\sum_i \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \max_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) + e(\mathbf{y}; \tilde{\mathbf{y}}_i) \right) \right)$$

Comparison

Min-max formulation of the structural SVM with margin rescaling:

$$\min_{\mathbf{w}} C \frac{\mathbf{w}^\top \mathbf{w}}{2} - \left(\sum_i \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \max_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) + e(\mathbf{y}; \tilde{\mathbf{y}}_i) \right) \right)$$

Maximum *a posteriori* estimation of the CRF:

$$\min_{\mathbf{w}} C \frac{\mathbf{w}^\top \mathbf{w}}{2} - \left(\sum_i \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \right)$$

Loss-Augmented Decoding

$$\max_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) + e(\mathbf{y}; \tilde{\mathbf{y}}_i) \right)$$

May be counter-intuitive: find a \mathbf{y} that's both high-scoring and high-error. If $\tilde{\mathbf{y}}_i$ wins, we know it's at least $e(\mathbf{y}', \tilde{\mathbf{y}}_i)$ better than competing \mathbf{y}' . If not, punish the \mathbf{y}' that does win!

Loss-Augmented Decoding

$$\max_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) + e(\mathbf{y}; \tilde{\mathbf{y}}_i) \right)$$

Efficiency depends on (i) efficiency of decoding and (ii) whether error function factors the same way as \mathbf{g} . Hamming loss, PARSEVAL scores are easy.

Slack rescaling version, usually intractable:

$$\max_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) \boxed{\times} e(\mathbf{y}; \tilde{\mathbf{y}}_i) \right)$$

Optimization

$$\min_{\mathbf{w}} C \frac{\mathbf{w}^\top \mathbf{w}}{2} - \left(\sum_i \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \max_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}) + e(\mathbf{y}; \tilde{\mathbf{y}}_i) \right) \right)$$

Continuous? Convex? Differentiable?

Optimization

- Subgradient methods (Ratliff, Bagnell, and Zinkevich, 2006), including an SGD-like version, seem to balance simplicity and effectiveness. Probably easier to work with, understand, and faster than structured SMO (Taskar et al., 2003) and dual extragradient (Taskar, Lacoste-Julien, and Jordan 2005).
- Cutting planes (Tsochantaridis et al., 2005)
- Exponentiated gradient (Bartlett, Collins, Taskar, and McAllester, 2004) is a promising alternative but requires working in the dual.

Dual and Factored Dual

Exploited by Taskar et al., and in exponentiated gradient.

Dual form:

$$\max_{\alpha \geq \mathbf{0}} -\frac{1}{2} \left(\sum_i \sum_{\mathbf{y} \in \mathcal{Y}} \alpha_{i,\mathbf{y}} ((\mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y}))) \right)^2 + \sum_i \sum_{\mathbf{y} \in \mathcal{Y}} \alpha_{i,\mathbf{y}} e(\mathbf{y}; \tilde{\mathbf{y}}_i)$$

Factored dual transforms this into a function of dual variables for “parts” rather than structures $\mathbf{y} \in \mathcal{Y}$. Often makes use of a linear program formulation of loss-augmented decoding.

Exponentiated Gradient

Can be used online or in batch mode. Here's the non-factored dual version for minimizing \mathcal{F} with respect to α :

$$\alpha_{i,y} \leftarrow \frac{\alpha_{i,y} \exp\left(-\eta \frac{\partial \mathcal{F}}{\partial \alpha_{i,y}}\right)}{\sum_{y' \in \mathcal{Y}} \alpha_{i,y'} \exp\left(-\eta \frac{\partial \mathcal{F}}{\partial \alpha_{i,y'}}\right)}$$

Can also be used for training CRFs!

See Collins, Globerson, Koo, Carreras, and Bartlett (2008) for a leisurely discussion of the factored dual version and application to parsing.

Passive Aggressive Learners: Perceptron Redux

Recall the perceptron online update:

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{g}\left(\tilde{\mathbf{x}}_i, \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{y})\right)$$

Crammer, Dekel, Keshet, Shalv-Schwartz, and Singer (2006) describe a set of algorithms that start here (rather than with a large margin objective) and try to endow the perceptron update with notions of error, regularization, and/or margin.

Margin-Infused Relaxation Algorithm

Similar to perceptron, but update is calculated as follows:

- Find k best-scoring structures $\langle \hat{\mathbf{y}}_{i,1}, \dots, \hat{\mathbf{y}}_{i,k} \rangle$.
- Generate k constraints, for $j \in \{1, \dots, k\}$:

$$\mathbf{g}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) - \mathbf{g}(\tilde{\mathbf{x}}_i, \hat{\mathbf{y}}_{i,j}) \geq e(\hat{\mathbf{y}}_{i,j}; \tilde{\mathbf{y}}_i)$$

- Calculate update, subject to constraints:

$$\mathbf{w} \leftarrow \underset{\mathbf{w}'}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}'\|^2$$

Often $k = 1$.