Probability and Structure in Natural Language Processing

Noah Smith, Carnegie Mellon University

2012 International Summer School in Language and Speech Technologies

Where We Left Off

- Graphical models … inference ... "max" inference and decoding with linear models (five views).
- Supervised learning:
  - Log loss
  - Error as loss

Comparison

	Generative (Log Loss)	Error as Loss
Computation	Convex optimization.	Optimizing a piecewise constant function.
Error-awareness	None	☺
Guarantees	Consistency.	None.

Discriminative Learning

- Various loss functions between log loss and error.
- Three commonly used in NLP:
  - Conditional log loss (“max ent,” CRFs)
  - Hinge loss (structural SVMs)
  - Perceptron’s loss
- We’ll discuss each, compare, and unify.

Log Loss (Second Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{y} \mid \boldsymbol{x})$$

- Can be understood as a generative model over **Y**, but does not model **X**.
  - "Conditional" model

Log Loss (Second Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{y} \mid \boldsymbol{x})$$

Examples:

- Logistic regression (for classification)
- MEMMs
- CRFs

Log Loss (Second Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{y} \mid \boldsymbol{x})$$

Computationally …

- Convex and differentiable.
- Requires posterior inference, which can be expensive depending on the model's structure.
- Linear decoding (for some parameterizations).

Log Loss (Second Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{y} \mid \boldsymbol{x})$$

Error …

- No notion of error.
- Learner wins by moving as much probability mass as possible to training examples' correct outputs.

Log Loss (Second Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{y} \mid \boldsymbol{x})$$

Guarantees…

- Consistency: if the true conditional model is in the right family, enough data will lead you to it.

Log Loss (Second Version)

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{y} \mid \boldsymbol{x})$$

Different parameterizations …

- Global log-linear (CRF):

$$-\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \log \sum_{\boldsymbol{y}'} \exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}', \boldsymbol{y}')$$

- Locally normalized log-linear (MEMM):

$$-\sum_{\boldsymbol{e}} freq(\boldsymbol{e}; \boldsymbol{x}, \boldsymbol{y}) \left(\mathbf{w}^\top \mathbf{g}(\boldsymbol{e}) - \log \sum_{\boldsymbol{e}' \in \mathcal{C}(\boldsymbol{e})} \exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{e}') \right)$$

Comparing the Two Log Losses

	$-\log p_w(\mathbf{x}, \mathbf{y})$	$-\log p_w(\mathbf{y} \mid \mathbf{x})$
Parameterization	Usually multinomials (BN-like).	Almost always log-linear (MN-like).
Under the usual parameterization …		
Computation	Count and normalize.	Convex optimization.
Error-awareness	None.	Aware of the **Y**-prediction task, (approximates zero-one).
Guarantees	Consistency of joint.	Consistency of cond.

Hinge Loss

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') + error(\boldsymbol{y}', \boldsymbol{y})$$

- Penalizes the model for letting competitors get close (in terms of score) to the correct answer **y**.
  - Can penalize them in proportion to their error.

Hinge Loss

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') + error(\boldsymbol{y}', \boldsymbol{y})$$

Examples …

- Perceptron (including Collins' structured version)
  - Classic version ignores *error* term
- SVM and some structured variants:
  - Max-margin Markov networks (Taskar et al.)
  - MIRA (1-best, k-best)

Hinge Loss

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') + error(\boldsymbol{y}', \boldsymbol{y})$$

Computationally …

- Convex, not everywhere differentiable.
  - Many specialized techniques now available.
- Requires MAP or "cost-augmented" MAP inference.
- Linear decoding.

Hinge Loss

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') + error(\boldsymbol{y}', \boldsymbol{y})$$

Error …

- Built in.
- Most convenient when error function factors similarly to features **g**.

Hinge Loss

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') + error(\boldsymbol{y}', \boldsymbol{y})$$

Guarantees …

- Generalization bounds.
  - Not clear how seriously to take these in NLP; may not be tight enough to be meaningful.
- Often you will find *convergence* guarantees for optimization techniques.

They Are All Related

$$\frac{1}{\beta} \log \sum_{\boldsymbol{y}'} \exp \left[\beta \left(\mathbf{w}^\top \left(\mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') - \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \right) + \gamma\, error(\boldsymbol{y}', \boldsymbol{y}) \right) \right]$$

	β	γ
Conditional log loss	1	0
Perceptron's hinge loss	∞	0
Structural SVM's hinge loss	∞	> 0
Softmax-margin (Gimpel and Smith, 2010)	1	1

CRFs, Max Margin, or Perceptron?

- For supervised problems, we do not expect large differences.
- Perceptron is easiest to implement.
  - With cost-augmented inference, it should get better and begins to approach MIRA and M^3Ns.
- CRFs are best for probability fetishists.
  - Probably most appropriate if you are extending with latent variables; the jury is out.
- Not yet "plug and play.."

R(**w**)

- Regularization term – avoid overfitting
  - Usually means "avoid large magnitudes in **w**"

- (Log) Prior – respect background beliefs about the predictor $h_{\mathbf{w}}$

R(**w**)

- Usual starting point: squared L_2 norm
  - Computationally convenient (it's strongly convex, it is its own Fenchel conjugate, …)
  - Probabilistic view: Gaussian prior on weights (Chen and Rosenfeld, 2000)
  - Geometric view: Euclidean distance (original regularization method in SVMs)
  - Only one hyperparameter

$$R(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2 = \lambda \sum_j w_j^2$$

R(**w**)

- Another option: L_1-norm
  - Computationally less convenient (not everywhere differentiable)
  - Probabilistic view: Laplacian prior on weights ("lasso" in regression)
  - Sparsity inducing ("free" feature selection)

$$R(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_j |w_j|$$

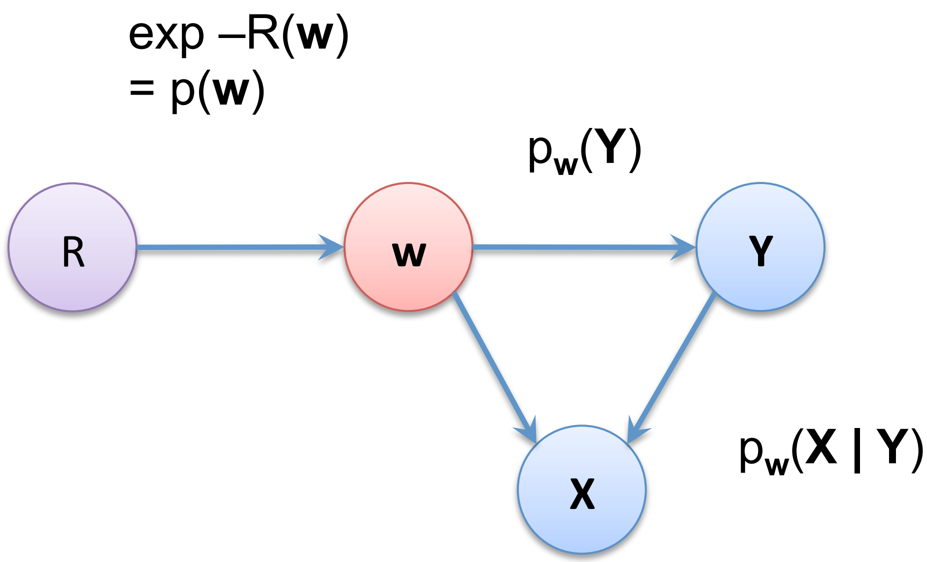
R(**w**)

- Lots of attention to this in machine learning.
- "Structured sparsity"
  - Want groups of features to go to zero, or group-internal sparsity, or …
- Interpolation between L_1 and L_2 – "elastic net"
  - Sparsity but maybe better behaved
- This is not yet "plug and play."
  - Optimization algorithm is heavily affected.

MAP Learning is Inference

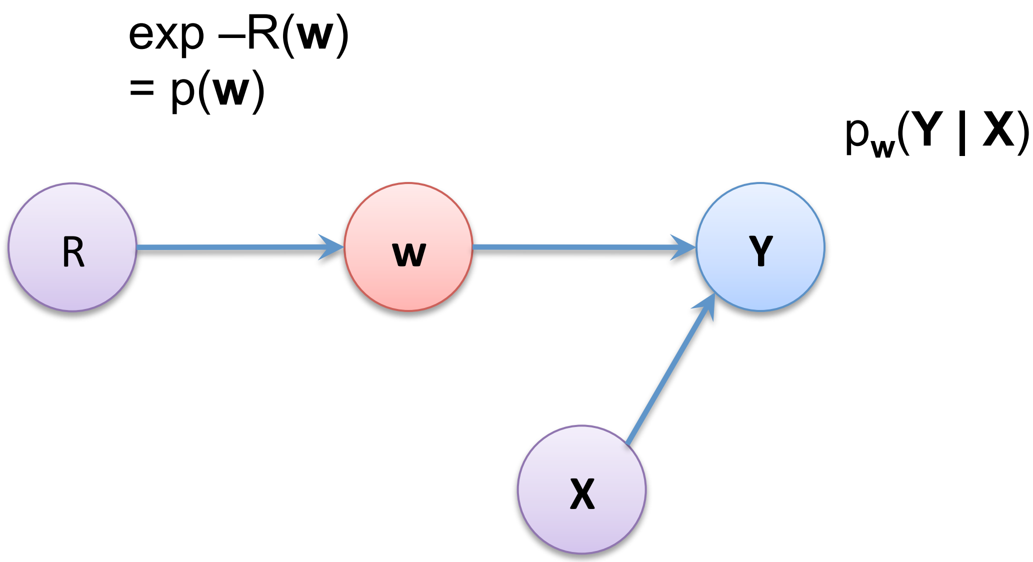
- Seeking “most probable explanation” of the data, in terms of **w**.
  - Explain the data: p(**x**, **y** | **w**)
  - Not too surprising: p(**w**)
- If we think of “**W**” as another random variable, MAP learning is MAP inference.
  - Looks very different from decoding!
  - But at a high level of abstraction, it is the same.

MAP Learning as a Graphical Model



- This is a view of learning a "noisy channel" model.

MAP Learning as a Graphical Model



- This is a view of learning in a CRF.

MAP Estimation for CRFs

$\max_{\mathbf{w}} p(\mathbf{w} \mid \mathbf{x}, \mathbf{y})$, which is MAP inference

iterate to obtain gradient:

sufficient statistics from $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$, obtained by *posterior* inference

How To Think About Optimization

- Depending on your choice of loss and R, different approaches become available.
  - Learning algorithms can interact with inference/ decoding algorithms, too.
- In NLP today, it is probably more important to focus on the features, error function, and prior knowledge.
  - Decide what you want, and then use the best available optimization technique.

Key Techniques

- Quasi-Newton – batch method for differentiable loss functions
  - LBFGS, OWLQN when using L_1 regularization
- Stochastic subgradient ascent – online
  - Generalizes perceptron, MIRA, stochastic gradient ascent
  - Sometimes sensitive to step size
  - Can often use "mini-batches" to speed up convergence
- For error minimization: randomization

Pitfalls

- Engineering online learning procedures is tempting and *may* help you get better performance.
  - Without at least some analysis in terms of loss, error, and regularization, it's unlikely to be useful outside your problem/dataset.
- When randomization is involved, look at variance across runs (Clark et al., 2011)
- Always tune hyperparameters (e.g., regularization strength) on *development* data!

Major Topics in Current Work

- Coping with approximate inference
- Exploiting incomplete data
  - Semisupervised learning
  - Creating features from raw text
  - Latent variable models (discussed tomorrow)
- Feature management
  - Structured sparsity (R)