Empirical Risk Minimization

October 29, 2015

Outline

- Empirical risk minimization view
  - Perceptron
  - CRF

Notation for Linear Models

- Training data: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- Testing data: $\{(x_{N+1}, y_{N+1}), \ldots (x_{N+N'}, y_{N+N'})\}$
- Feature function: **g**
- Weights: **w**
- Decoding:

$$\text{decode}(\mathbf{w}, \boldsymbol{x}) = \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})$$

- Learning:

$$\text{learn}\left(\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right) = \arg\max_{\mathbf{w}} \Phi\left(\mathbf{w}, \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right)$$

- Evaluation:

$$\frac{1}{N'} \sum_{i=1}^{N'} \text{cost}\left(\text{decode}\left(\text{learn}\left(\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right), \boldsymbol{x}_{N+i}\right), \boldsymbol{y}_{N+i}\right)$$

Structured Perceptron

- Described as an online algorithm.
- On each iteration, take one example, and update the weights according to:

$$\mathbf{w} \quad \leftarrow \quad \mathbf{w} + \alpha \left(\mathbf{g}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \mathbf{g}(\boldsymbol{x}_t, \mathrm{decode}(\mathbf{w}, \boldsymbol{x}_t)) \right)$$

- Not discussing today: the theoretical guarantees this gives, separability, and the averaged and voted versions.

Empirical Risk Minimization

- A unifying framework for many learning algorithms.

$$\begin{aligned} \mathrm{learn}\left(\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right) &= \arg\max_{\mathbf{w}} \Phi\left(\mathbf{w}, \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right) \\ &= \arg\min_{\mathbf{w}} \underbrace{\frac{1}{N}\sum_{i=1}^N L(\mathbf{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)}_{\approx\ \mathbb{E}[L(\mathbf{w}, \boldsymbol{X}, \boldsymbol{Y})]} + R(\mathbf{w}) \end{aligned}$$

- Many options for the loss function L and the regularization function R.

Solving the Minimization Problem

- In some friendly cases, there is a closed form solution for the minimizer of **w**
  - E.g., the maximum likelihood estimator for HMMs
- *Usually*, we have to use an iterative algorithm which amounts to progressively finding better versions of w
  - involves hard/soft inference with each improved value of **w** on either part or all of the training set

Loss Functions You May Know

Name	Expression of $L(\mathbf{w}, x, y)$
Log loss (joint)	$-\log p(\boldsymbol{x}, \boldsymbol{y} \mid \mathbf{w})$
Log loss (conditional)	$-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$
Zero-one loss	$\mathbf{1}\{\mathrm{decode}(\mathbf{w}, \boldsymbol{x}) \neq \boldsymbol{y}\}$
Expected zero-one loss	$1 - p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$

Loss Functions You May Know

Name	Expression of $L(\mathbf{w}, x, y)$
Log loss (joint)	$-\log p(\boldsymbol{x}, \boldsymbol{y} \mid \mathbf{w})$
Log loss (conditional)	$-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$
Zero-one loss	$\mathbf{1}\{\text{decode}(\mathbf{w}, \boldsymbol{x}) \neq \boldsymbol{y}\}$
Expected zero-one loss	$1 - p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$

Loss Functions You May Know

Name	Expression of $L(\mathbf{w}, \boldsymbol{x}, \boldsymbol{y})$
Log loss (joint)	$-\log p(\boldsymbol{x}, \boldsymbol{y} \mid \mathbf{w})$
Log loss (conditional)	$-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$
Cost	$\mathrm{cost}(\mathrm{decode}(\mathbf{w}, \boldsymbol{x}), \boldsymbol{y})$
Expected cost, a.k.a. "risk"	$\mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x}, \mathbf{w})}[\mathrm{cost}(\boldsymbol{Y}, \boldsymbol{y})]$

CRFs and Loss

- Plugging in the log-linear form (and not worrying at this level about locality of features):

$$
\begin{aligned}
-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w}) &= -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \log \sum_{\boldsymbol{y}'} \exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') \\
\frac{\partial L}{\partial w_j} &= -g_j(\boldsymbol{x}, \boldsymbol{y}) + \mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x}, \mathbf{w})}[g_j(\boldsymbol{x}, \boldsymbol{Y})]
\end{aligned}
$$

CRFs and Loss

- Plugging in the log-linear form (and not worrying at this level about locality of features):

$$
\begin{aligned}
-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w}) &= -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \log \sum_{\boldsymbol{y}'} \exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') \\
\frac{\partial L}{\partial w_j} &= -g_j(\boldsymbol{x}, \boldsymbol{y}) + \mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x}, \mathbf{w})}[g_j(\boldsymbol{x}, \boldsymbol{Y})]
\end{aligned}
$$

$$
\frac{\partial L}{\partial \mathbf{w}} = -\mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x}, \boldsymbol{w})}\left[\mathbf{g}(\boldsymbol{x}, \boldsymbol{Y})\right]
$$

Training CRFs and Other Linear Models

- Early days: iterative scaling (specialized method for log-linear models only)
- ~2002: quasi-Newton methods
  - (using LBFGS which dates from the late 1980s)
- ~2006: stochastic gradient descent
- ~2010: adaptive gradient methods

Perceptron and Loss

- Not clear immediately what L is, but the "gradient" of L should be:

$$-g_j(\boldsymbol{x}, \boldsymbol{y}) + g_j(\boldsymbol{x}, \mathrm{decode}(\mathbf{w}, \boldsymbol{x}))$$

- The vector of above quantities is actually a *subgradient* of:

$$L(\mathbf{w}, \boldsymbol{x}, \boldsymbol{y}) = \max\{0, -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})) + \max_{\hat{\boldsymbol{y}}} \mathbf{w}^\top (\boldsymbol{x}, \hat{\boldsymbol{y}})\}$$

Compare

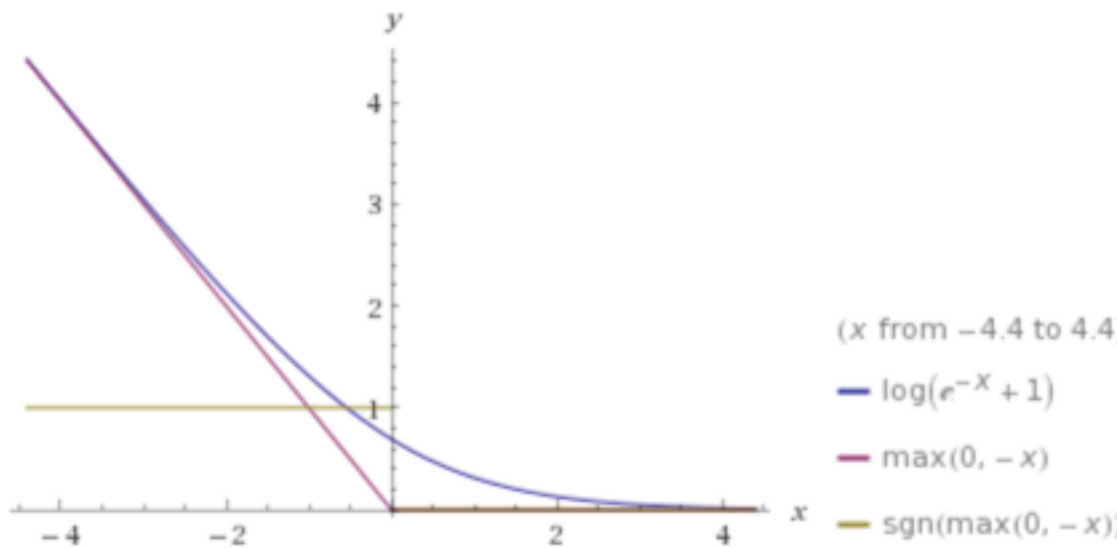
- CRF (log-loss):

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

- Perceptron:

$$L(\mathbf{w}, \boldsymbol{x}, \boldsymbol{y}) = \max\{0, -\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})) + \max_{\hat{\boldsymbol{y}}} \mathbf{w}^\top (\boldsymbol{x}, \hat{\boldsymbol{y}})\}$$

Loss Functions



Loss Functions You Know

Name	Expression of $L(\mathrm{w}, x, y)$	Convex?
Log loss (joint)	$-\log p(\boldsymbol{x}, \boldsymbol{y} \mid \mathbf{w})$	✔
Log loss (conditional)	$-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$	✔
Cost	$\mathrm{cost}(\mathrm{decode}(\mathbf{w}, \boldsymbol{x}), \boldsymbol{y})$	
Expected cost, a.k.a. "risk"	$\mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x}, \mathbf{w})}[\mathrm{cost}(\boldsymbol{Y}, \boldsymbol{y})]$	
Perceptron loss	$-\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}')$	✔

Loss Functions You Know

Name	Expression of $L(\mathbf{w}, \boldsymbol{x}, \boldsymbol{y})$	Cont.?
Log loss (joint)	$-\log p(\boldsymbol{x}, \boldsymbol{y} \mid \mathbf{w})$	✔
Log loss (conditional)	$-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$	✔
Cost	$\mathrm{cost}(\mathrm{decode}(\mathbf{w}, \boldsymbol{x}), \boldsymbol{y})$	
Expected cost, a.k.a. "risk"	$\mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x}, \mathbf{w})}[\mathrm{cost}(\boldsymbol{Y}, \boldsymbol{y})]$	✔
Perceptron loss	$-\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}')$	✔

Loss Functions You Know

Name	Expression of $L(\mathrm{w}, x, y)$	Cost?
Log loss (joint)	$-\log p(\boldsymbol{x}, \boldsymbol{y} \mid \mathbf{w})$	
Log loss (conditional)	$-\log p(\boldsymbol{y} \mid \boldsymbol{x}, \mathbf{w})$	
Cost	$\mathrm{cost}(\mathrm{decode}(\mathbf{w}, \boldsymbol{x}), \boldsymbol{y})$	✔
Expected cost, a.k.a. "risk"	$\mathbb{E}_{p(\boldsymbol{Y} \mid \boldsymbol{x}, \mathbf{w})}[\mathrm{cost}(\boldsymbol{Y}, \boldsymbol{y})]$	✔
Perceptron loss	$-\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}')$	

The Ideal Loss Function

For computational convenience:

- Convex
- Continuous

For good performance:

- Cost-aware
- Theoretically sound

On Regularization

- In principle, this choice is independent from the choice of the loss function.
- Squared L_2 norm is the most common starting place.

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

- L_1 and other sparsity-inducing regularizers as well as structured **regularizers** are of interest

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

Practical Advice

- Features still more important than the loss function.
  - But general, easy-to-implement algorithms are quite useful!
- Perceptron is easiest to implement.
- CRFs and max margin techniques usually do better.
- Tune the regularization constant, λ.
  - Never on the test data.