Topics in Natural Language Processing

Shay Cohen

Institute for Language, Cognition and Computation

University of Edinburgh

Lecture 8
Administrativia

- The schedule for presentations and brief responses was sent to everybody
- One slot with three presenters
- Please plan to come at 1pm to this lecture
- Neural networks were quite popular
- Presentations: coordinate with me
- Regarding the essay: start thinking about it
Log-linear models. \( p(x, y \mid w) = \frac{\exp \left( \sum_{i=1}^{d} w_i \cdot g_i(x, y) \right)}{Z(w)} \)

\[ Z(w) = \sum_{x, y} \exp \left( \sum_{i=1}^{d} w_i \cdot g_i(x, y) \right) \]

Gradient:

\[ \frac{\partial L}{\partial w_j} = E_{\tilde{p}} \left[ \theta_j \right] - E_{\tilde{p}(1w)} \left[ g_j \right] \]

Log-likelihood maximisation tries to have the model feature expectations and the empirical distribution feature expectations “agree”
Overfitting

The advantage of log-linear models: can have arbitrary features

The problem: too many features lead to overfitting
Regularisation

What is overfitting?
$L_2$ Regularisation

New objective:

$$G(w|x_1, y_1, \ldots, x_n, y_n) = L(w|x_1, y_1, \ldots, x_n, y_n) - \frac{\lambda}{2} ||w||^2_2$$

where $||w||^2_2 = \sum_{i=1}^{d} w_i^2$

Partial derivatives:

$$\frac{\partial G}{\partial w_j} = \frac{\partial L}{\partial w_j} - \frac{\lambda}{2} \cdot 2 \cdot w_j =$$

$$= \frac{\partial L}{\partial w_j} - \lambda \cdot w_j$$
\textbf{L}_1 \text{ Regularisation}

New objective:

\[ G(w|x_1,y_1,\ldots,x_n,y_n) = L(w|x_1,y_1,\ldots,x_n,y_n) - \lambda ||w||_1^2 \]

where \[ ||w||_1^2 = \sum_{i=1}^{d} |w_i| \]

Encourages sparse solutions, such that most of \( w_i \) are exactly 0

"feature selection"
Bayesian interpretation to regularisation

\[ G(w | x_1, y_1, \ldots, x_n, y_n) = L(w | x_1, y_1, \ldots, x_n, y_n) - \frac{\lambda}{2} ||w||^2 \]

Could the answer be a MAP estimate for some prior?

\[ G(w | x_1, y_1, \ldots, x_n, y_n) \propto \log p(x_1, y_1, \ldots, x_n, y_n | w) + \log p(w) \]

\[ p(w) \propto e^{\exp \left( -\frac{\lambda}{2} \sum_{i=1}^{d} w_i^2 \right)} = e^{\exp \left( -\frac{\sum_{i=1}^{d} w_i^2}{2 \times \frac{1}{\lambda}} \right)} \]
Bayesian interpretation to regularisation

\[ G(w|x_1, y_1, \ldots, x_n, y_n) = L(w|x_1, y_1, \ldots, x_n, y_n) - \frac{\lambda}{2} ||w||^2_2 \]

Could the answer be a MAP estimate for some prior?

\[ G(w|x_1, y_1, \ldots, x_n, y_n) \propto \log p(x_1, y_1, \ldots, x_n, y_n|w) + \log p(w) \]

\[ p(w) \propto \text{likelihood} \quad \text{Gaussian} \]

This means that \( p(w) \) is a Gaussian distribution with mean 0 and variance \( 1/\lambda \)

**MLE with \( L_2 \)-regularisation is MAP estimate with Gaussian prior**
Today’s class

Learning from incomplete data
Learning from Incomplete Data

- Semi-supervised learning
  
  Small amounts of labeled data and "large" amounts of unlabeled data.

- Latent variable learning
  
  Add extra information to the model.

- Unsupervised learning
  
  Hand: Given input examples, learn a decoder.
How to estimate a PCFG?

We learned how to estimate a PCFG from treebank

Reminder: count and normalize
Unsupervised learning: PCFGs

How to estimate a PCFG from strings?

(Assume we have grammar)
General case: Viterbi (or “hard”) EM

Model: \[ p(x, y \mid \theta) \] (this means the grammar structure/rules are known)

Observed Data: \[ x_1, \ldots, x_n \]

Step 0:

\[ \text{Guess some} \ \theta \]

Step 1:

\[ \text{Parse} \ x_1, \ldots, x_n \ \text{using} \ \theta \Rightarrow y_1, \ldots, y_n \]

Step 2:

\[ \text{Estimate} \ \theta \ \text{using} \ y_1, \ldots, y_n \]

Repeat step 1
Maximum likelihood estimation

General principle: write down the likelihood of whatever you observe, and then maximise with respect to parameters

Model: $p(x, y | \theta)$

Observed: $x_1, \ldots, x_n$

Likelihood:

$L(x_1, \ldots, x_n | \theta) = \prod_{i=1}^{n} p(x_i | \theta) = \prod_{i=1}^{n} \left( \sum_{y} p(x_i, y | \theta) \right)$

$\max_{\theta} \sum_{i=1}^{n} \log \left( \sum_{y} p(x_i, y | \theta) \right)$
The EM Algorithm

- A softer version of hard EM

- Instead of identifying a single tree per sentence, identify a distribution over trees (E-step)

- Then re-estimate the parameters, with each tree for each sentence “voting” according to its probability (M-step)

- Semiring parsing: instead of CKY use the inside algorithm
EM: Main Disadvantage

Sensitivity to initialisation (finds local maximum)

Global log-likelihood optimisation in general is “hard”
Latent-variable learning

“Structure” is present

Some information is missing from model

Model: $p(x, y, h | \theta)$

Observed: $(x_1, y_1), \ldots, (x_n, y_n)$

Log-likelihood:

$$L(x_1, \ldots, x_n, y_1, \ldots, y_n | \theta) = \prod_{i=1}^{n} p(x_i, y_i | \theta) = \prod_{i=1}^{n} \sum_{h} p(x_i, y_i, h | \theta)$$
Example of Latent-Variable Use in PCFGs

“Context-freeness” can lead to over-generalization:

**Seen in data:**

```
S
  NP   VP
    D   N   V   NP
      the dog saw P
```

**Unseen in data (ungrammatical):**

```
S
  NP   VP
    N   V   NP
      him saw D   N
```

The latent states for each node are never observed.
How to learn with latent variables?

- Expectation-Maximisation (EM)

- Current surging interest: method of moments and spectral learning

- Revival of old methods: Neural networks

- Other methods