CRF Feature Induction

Andrew McCallum “Efficiently Inducing Features of Conditional Random Fields”

Kuzman Ganchev
1 Introduction
   Basic Idea
   Aside: Transformation Based Learning
   Notation/CRF Review

2 Arbitrary CRFs

3 Linear CRFs

4 Results
Basic Idea: Greedy Search

- Assume we have a large set of candidate features.
- Repeat until < something >:
  - Train a CRF using current set of features
  - Add “best” new feature(s)
- “best” means “greatest increase in log likelihood
Aside: One-slide TBL

- **Input**
  - labeled structured data (e.g. POS tagged text)
  - large set of rule candidates (type: if X and Y then Z)
- **Learning method: greedy search**
  - Repeat until < something >:
    - Add “best” new feature(s)
    - where “best” means greatest error reduction.
- **Output of learning:**
  - Ordered set of rules
Notation

- $\bar{x}$ - an observed instance (input to classifier)
- $\bar{y}$ - a label instance (output of classifier)
- $C(\bar{x})$ - the set of cliques of the Markov graph for a particular observed instance.
- $n_i$ - set of neighbors of node $y_i$.
- Classifier is:

$$
\arg\max_y p(\bar{y}|\bar{x})
$$

$$
p(\bar{y}|\bar{x}) = \frac{1}{Z(\bar{x})} \exp \left( \sum_{c \in C} \lambda_c f_c(\bar{y}_c | \bar{x}) \right)
$$
Linear CRFs

- Graph is just a chain of nodes
- Can represent all clique features as features of edges.

\[
p(y|x) = \frac{1}{Z(x)} exp \left( \sum_i \lambda_i f_i(y_i, y_{i-1}|x) \right)
\]
Basic Idea: Greedy Search (repeat)

- Assume we have a large set of candidate features.
- Repeat until < something >:
  - Train a CRF using current set of features
  - Add “best” new feature(s)
- “best” means “greatest increase in log likelihood
**General CRFs: Log Likelihood**

- **Objective function is log likelihood of training data (i is a training instance):**

\[
L = \sum_i \log(p(y_i|\bar{x}_i))
\]

\[
= \sum_i \left( \sum_{c \in C} \lambda_c f_c(y_{i,c}|\bar{x}_i) - \log Z(\bar{x}_i) \right)
\]

- Defined for a set of features and their weights \( \Lambda \).
- Convex in feature weights: we can take \( \max_\Lambda L_\Lambda \).
General CRFs: Maximizing Gain

- We want features that give us best “improvement in log likelihood”
- define $G(g)$, gain for feature $g$:

$$G(g) = \max_{\mu} G^\Lambda(g, \mu)$$

$$= \max_{\mu} L^\Lambda_{g, \mu} - L^\Lambda - \frac{\mu^2}{2\sigma^2}$$

- Last term is regularization.
General CRFs: Assume $\Lambda$ doesn’t change

- We begin by assuming weights for old features remain unchanged.
- This gives us (sum over instances omitted):

$$G(g) = \max_{\mu} L_{\Lambda, g, \mu} - L_\Lambda - \frac{\mu^2}{2\sigma^2}$$

$$= \max_{\mu} \sum_{c \in C} \mu g(y_c|x) - \log Z_{\Lambda, g, \mu}(x) + \log Z_{\Lambda}(x) - \frac{\mu^2}{2\sigma^2}$$

- No need to retrain each feature considered
- Still retrain for features actually added
General CRFs: Mean Field Approximation

- $p(\vec{y}|\vec{x}) \approx \prod_{v \in V} p(y_v|\vec{x})$
- (avoids joint inference)
- calculate max likelihood distributions for all output values $p(y_i = y|\vec{x})$.
- To infer distribution at node $y$, assume distributions at all other nodes are fixed.

\[
p_{\Lambda,g,\mu}(y_i = y|\vec{x}) \approx \sum_{\vec{y}'} p_{\Lambda,g,\mu}(y_i = y|n_i = \vec{y}', \vec{x})p_{\Lambda}(n_i = \vec{y}'|\vec{x})
\]
**General CRFs: Gain only for some nodes**

- With mean field approximation:

\[
G(g) = \max_{\mu} \sum_{v \in V} \mu_g(y_v|n_v, \bar{x}) - \log Z_{\Lambda,g,\mu}(\bar{x}) + \log Z_{\Lambda}(\bar{x}) - \frac{\mu^2}{2\sigma^2}
\]

- Restrict calculation to “wrong” or “borderline” label nodes
  - Use only nodes that are mislabeled by the current model or nodes that are within some margin.
- Quoth McCallum: do exact inference for a while at start.
Linear CRFs: Agglomerated Features

- transition features less important than observation features
- define agglomerative features $g(y_i|n_{i-1}, x) \simeq g(y_i|x)$
- This gives us:

$$p_{\Lambda,g,\mu}(y_i = y|x) = \frac{p_{\Lambda}(y_i = y|x)e^{\mu g(y_i,x)}}{Z_i(\Lambda, g, \mu)}$$

- $Z_i(\Lambda, g, \mu)$ is easy to compute.
- Exact calculation of $p_{\Lambda}(y_i = y|x)$
Linear CRFs: Agglomerated Features (contd)

- Still use only “wrong” $y_i$ for gain.

$$G_{\Lambda, g, \mu} = \sum_{i \in \text{wrong}} \log \left( \frac{e^{\mu g(y_i, \bar{x})}}{Z_i(\Lambda, g, \mu)} \right) - \frac{\mu^2}{2\sigma^2}$$

$$= |\text{wrong}| \mu E[g] - \sum_{i \in \text{wrong}} \log(E_{\Lambda}(e^{\mu, g|\bar{x}})) - \frac{\mu^2}{2\sigma^2}$$

- Claim: This converges in just 10 iterations of Newton’s method
Linear CRFs: Other simplifications

- Add many features at a time
- After adding features train for only a few iterations of BFGS
  - Claim: this helps reduce overfitting
  - Does it really?
**English named entity extraction.**

<table>
<thead>
<tr>
<th></th>
<th>Without induction</th>
<th></th>
<th>With induction</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Recall</td>
<td>F1</td>
<td>Prec</td>
</tr>
<tr>
<td><strong>PER</strong></td>
<td>91.8</td>
<td>46.7</td>
<td>61.9</td>
<td>93.2</td>
</tr>
<tr>
<td><strong>LOC</strong></td>
<td>94.1</td>
<td>80.5</td>
<td>86.7</td>
<td>93.0</td>
</tr>
<tr>
<td><strong>ORG</strong></td>
<td>92.0</td>
<td>48.5</td>
<td>63.5</td>
<td>84.9</td>
</tr>
<tr>
<td><strong>MISC</strong></td>
<td>91.7</td>
<td>66.7</td>
<td>77.2</td>
<td>83.1</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>92.7</td>
<td>60.7</td>
<td>73.3</td>
<td>89.8</td>
</tr>
</tbody>
</table>

CRF Feature Induction (Ganchev)