Sequence Labeling

Grzegorz Chrupała and Nicolas Stroppa

Saarland University
Google

META Workshop

Outline

1. Conditional Random Fields
2. Additional Remarks

Outline

1 Conditional Random Fields

2 Additional Remarks

Context

Sequence labeling setting:

- Input sequence: $\mathbf{x} = x_1, x_2, \ldots, x_N$
- Output sequence: $\mathbf{z} = z_1, z_2, \ldots, z_N$

Methods already introduced for sequence labeling: HMM (generative), Maximum Entropy Markov Model (discriminative)

Example

Word	POS	Chunk	NE
West	NNP	B-NP	B-MISC
Indian	NNP	I-NP	I-MISC
all-rounder	I-NP	B-NP	O
Phil	NNP	I-NP	B-PER
Simons	NNP	I-NP	I-PER
took	VBP	B-VP	O
four	CD	B-NP	O
for	IN	B-PP	O
38	CD	B-NP	O
on	IN	B-PP	O
Friday	NNP	B-NP	O
as	IN	B-PP	O
Leicestershire	NNP	B-NP	B-ORG
beat	VBP	B-VP	O

Discriminative models for Sequence Labeling - Local classifier

For each position i, classify each x_i independently (sliding window)

- Criterion: $z_i = \text{argmax}\, P(z_i|x_i)$
- Each example consists of a single position in the sequence (a token/label pair)
- Any classification algorithm can be used

Discriminative models for Sequence Labeling - Local classifier

For each position i, classify each x_i independently (sliding window)

- Criterion: $z_i = \text{argmax}\, P(z_i|x_i)$
- Each example consists of a single position in the sequence (a token/label pair)
- Any classification algorithm can be used
  - Maxent
  - Perceptron
  - k-nn
  - SVM
  - ...

Discriminative models for Sequence Labeling - Local classifier

For each position i, classify each x_i independently (sliding window)

- Criterion: $z_i = \text{argmax}\, P(z_i|x_i)$
- Each example consists of a single position in the sequence (a token/label pair)
- Any classification algorithm can be used
  - Maxent
  - Perceptron
  - k-nn
  - SVM
  - ...

Problem: the sequence nature of the problem is not taken into account at all (e.g. nothing preventing from outputing impossible label bigrams)

Discriminative models for Sequence Labeling - Local classifier++

Local classifier with previous labels: for each position i, classify each x_i sequentially (e.g. from left to right), making use of the previously assiged label

- Criterion: $z_i = \text{argmax}\, P(z_i|x_i, z_{i-1})$
- Training is performed similarly as before, but we now can design features that include the previous label

Discriminative models for Sequence Labeling - Local classifier++

Local classifier with previous labels: for each position i, classify each x_i sequentially (e.g. from left to right), making use of the previously assiged label

- Criterion: $z_i = \text{argmax}\, P(z_i|x_i, z_{i-1})$
- Training is performed similarly as before, but we now can design features that include the previous label

Problem: Errors can be easily propagated

Discriminative models for Sequence Labeling - MEMM

Maximum entropy Markov Model: same as above for training, but label by maximizing on the entire sequence

- $z = \text{argmax}\, P(\mathbf{z}|\mathbf{x}) = \text{argmax}\, \Pi_i P(z_i|x_i, z_{i-1})$

Discriminative models for Sequence Labeling - MEMM

Maximum entropy Markov Model: same as above for training, but label by maximizing on the entire sequence

- $z = \text{argmax}\, P(\mathbf{z}|\mathbf{x}) = \text{argmax}\, \Pi_i P(z_i|x_i, z_{i-1})$

Problem: we're learning a model on a position-basis, so we're still not fully exploiting the sequential nature of the data

Conditional Random Fields

Conditional Random Fields enable to define "true" sequence learning models, i.e. models that learn and operate on sequences.

This means we model directly

$$P(\mathbf{z}|\mathbf{x})$$

without resorting to position-based decompositions of the model definition.

As they are able to deal *natively* with "structured" data, they are referred to as *structure learning* methods.

Conditional Random Fields

Some reasons to focus on CRFs.

- *Sound* and understood principle (maximum entropy)
- Very *flexible* for feature definition
- *Natively'* handling sequences
- *State of the art results* on numerous applications
- Because of this success, now the *baseline* in sequence labeling

Conditional Random Fields

- The principles underlying Conditional Random Fields are identical to the one behind Maximum Entropy classifiers
- You can actually view CRFs as the counterpart of Maxent for sequences

Conditional Random Fields

- The principles underlying Conditional Random Fields are identical to the one behind Maximum Entropy classifiers
- You can actually view CRFs as the counterpart of Maxent for sequences

This yields the following model:

$$P(\mathbf{z}|\mathbf{x}) = \frac{\exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}))}{Z(\mathbf{x})},$$

where $Z(\mathbf{x}) = \sum_z \exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}))$ is the partition function.

Remarks:

- Features now operate on sequences
- The sum in the partition function is big!

Conditional Random Fields as a linear model

CRF model:

$$P(\mathbf{z}|\mathbf{x}) = \frac{\exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}))}{Z(\mathbf{x})}$$

CRF prediction:

$$\begin{aligned}\operatorname*{argmax}_z P(\mathbf{z}|\mathbf{x}) &= \operatorname{argmax} \frac{\exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}))}{Z(\mathbf{x})} \\ &= \operatorname{argmax} \exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z})) \\ &= \operatorname{argmax} \mathbf{w}.\Phi(\mathbf{x},\mathbf{z})\end{aligned}$$

Once the model is learnt, it's a basic linear model, so you can *apply* the model the same way as for MeMM or structured perceptron...

Linear Conditional Random Fields Features

General form: $\Phi_j(\mathbf{x}, \mathbf{z})$

In linear CRFs, we assume a particular structure (close to HMM).

Features are the same as for MeMM, or structured perceptron...

Training Conditional Random Fields

How do we train CRFs?

No closed form to the associated optimization problem. Needs to resort to numeric optimization techniques:

- L-BFGS
- Improved Iterative Scaling
- Stochastic Gradient Descent (SGD)

We'll focus on SGD since it's quite easy, efficient and we already mentionned it

Stochastic Gradient Descent

Expression of the *conditional log-likelihood*:

$$\log P(\mathbf{z}|\mathbf{x};\mathbf{w}) = \log \frac{\exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}))}{\sum_{z'} \exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}'))} = \mathbf{w}.\Phi(\mathbf{x},\mathbf{z}) - \log \sum_{z'} \exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}'))$$

Our goal is to *maximize* this likelihood.

In order to perform gradient descent, we need to compute its gradient.

$$\begin{aligned}
\nabla_{\mathbf{w}} \log P(\mathbf{z}|\mathbf{x};\mathbf{w}) &= \Phi(\mathbf{x},\mathbf{z}) - \sum_{z'} \Phi(\mathbf{x},\mathbf{z}') \frac{\exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}'))}{\sum'_{z'} \exp(\mathbf{w}.\Phi(\mathbf{x},\mathbf{z}''))} \\
&= \Phi(\mathbf{x},\mathbf{z}) - \sum_{z'} \Phi(\mathbf{x},\mathbf{z}') P(\mathbf{z}'|\mathbf{x};\mathbf{w}) \\
&= \Phi(\mathbf{x},\mathbf{z}) - E_{P(\mathbf{z}'|\mathbf{x};\mathbf{w})} \Phi(\mathbf{x},\mathbf{z}')
\end{aligned}$$

Stochastic Gradient Descent

In stochastic gradient descent, we appromixate this conditional log-likelihood with the *empirical* conditional log-likelihood, computed on training examples.

$$\nabla_{\mathbf{w}} \log P(\mathbf{Z}|\mathbf{X};\mathbf{w}) = \sum_i \Phi(\mathbf{x}_i, \mathbf{z}_i) - E_{P(\mathbf{z}'_i|\mathbf{x}_i;\mathbf{w})}\Phi(\mathbf{x}_i, \mathbf{z}'_i)$$

The idea is that we sample the example, and apply an update rule per example:

$$\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x}_i, \mathbf{z}_i) - E_{P(\mathbf{z}'_i|\mathbf{x}_i;\mathbf{w})}\Phi(\mathbf{x}_i, \mathbf{z}'_i)$$

Note: computation of $E_{P(\mathbf{z}'_i|\mathbf{x}_i;\mathbf{w})}\Phi(\mathbf{x}_i, \mathbf{z}'_i)$ can be done by dynamic programming if we use linear CRF (cf. forward-backward algo).

Some notes on SGD

SGD only computes the gradient (first-order partial derivatives), and not the hessian (second-order partial derivatives).

Using the first-order info is less precise but it's much more efficient than using second-order info.

There exists some more or less complex improvements made around this to keep this efficient while making it less approximative.

Outline

1 Conditional Random Fields

2 Additional Remarks

Structured perceptron

CRF SGD update:

$$\mathbf{w} \leftarrow w + \Phi(\mathbf{x}_i, \mathbf{z}_i) - E_{P(\mathbf{z}'_i|\mathbf{x}_i;\mathbf{w})}\Phi(\mathbf{x}_i, \mathbf{z}'_i)$$

Structured perceptron update:

$$\mathbf{w} \leftarrow w + \Phi(\mathbf{x}_i, \mathbf{z}_i) - \Phi(\mathbf{x}_i, \hat{\mathbf{z}}'_i),$$

with:

$$E_{P(\mathbf{z}'|\mathbf{x};\mathbf{w})}\Phi(\mathbf{x}, \mathbf{z}') = \sum_{z'} \Phi(\mathbf{x}, \mathbf{z}') \frac{\exp(\mathbf{w}.\Phi(\mathbf{x}, \mathbf{z}'))}{\sum'_{z'} \exp(\mathbf{w}.\Phi(\mathbf{x}, \mathbf{z}''))}$$

and

$$\Phi(\mathbf{x}_i, \hat{\mathbf{z}}'_i) = \sum_{z'} \Phi(\mathbf{x}, \mathbf{z}') \begin{cases} 1 \text{ if z' is best output} \\ 0 \text{ otherwise} \end{cases}$$

The only difference is that CRF SGD is using softmax and structured perceptron "hard" max!

Comparison with HMM and Maxent

Analogy

If you make sense of the following analogy, we're quite happy!

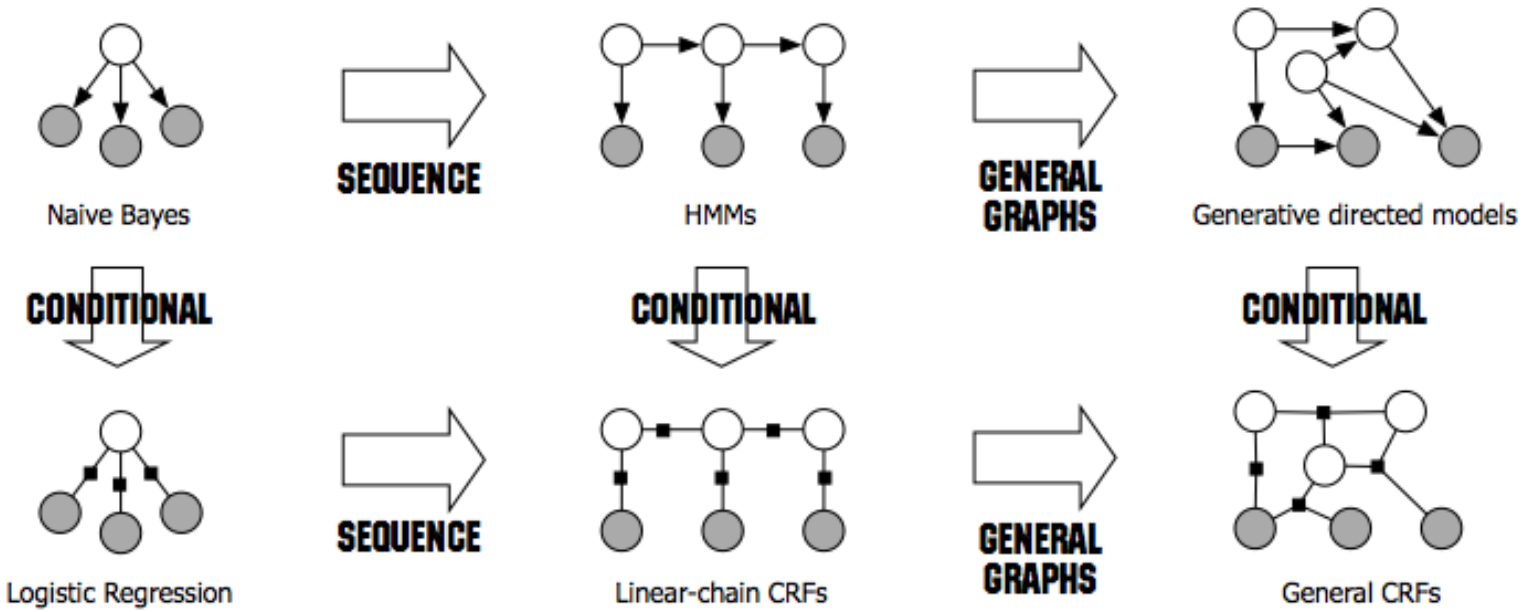
Maxent:Naive Bayes::CRF::HMM

Comparison with HMM and Maxent

Analogy

If you make sense of the following analogy, we're quite happy!

Maxent:Naive Bayes::CRF::HMM



Linear and higher-order CRF

There exists several "flavors" of CRF, with different expressiveness power.

Semi-markov CRF

In semi-markov CRFs, we can define features on several consecutive positions (segmentation as a hidden variable).
Input and Output sequences can have different length.