Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

(John Lafferty et. al.)

Presenter: Devanshu Jain

(devjain@seas.upenn.edu)

Layout

- Problem Statement
- Generative Models: HMM
- Limitations of Generative Models
- Discriminative Models: MEMM
- Limitations of MEMM
- Discriminative Models: CRF
  - Loss Function convexity
  - Inference
  - Parameter Estimation
- Experiments
- CRF spin-offs

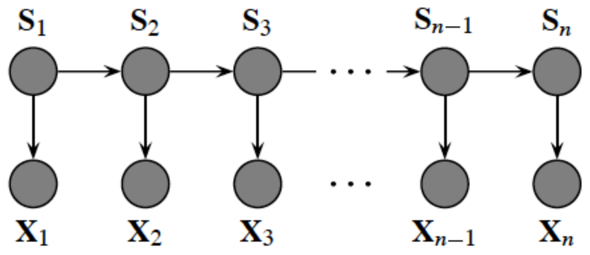
Problem Statement

- The problem, we wish to solve: Labelling Sequence Data
  - POS tagging
  - Named Entity Recognition

- Statement: Given an observation sequence **x**, we want to choose a label sequence **y*** such that the conditional probability P(**y** | **x**) is maximized, that is:

$$\boldsymbol{y}^{*} = \boldsymbol{arg\,max_y\,P(y\,|x)}$$

Generative Models: HMM



- HMMs can be used to solve such problems
- In NER, each observation (x_t) can be the identity of the word at position t and each state (y_t) can be the named-entity label, i.e. one of {*Person*, *Organization*, *Location*, *Other*}.
  - To be precise, named entities can be multi-tokens. So, BIO-method.
  - B: Beginning, I: Intermediate, O: Outside. So each label is prefixed with these letters which indicate whether it's the beginning or continuation of a named entity.
- It makes 2 assumptions:
  1. Each state depends only on its immediate predecessor, that is, $y_t \perp y_i$ given y_{t-1} such that i={1, 2, …, t-2}
  2. Each observation x_t depends only on its current state y_t

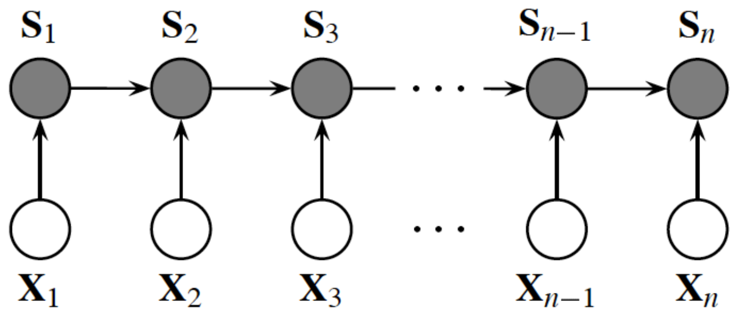
$$p(y, x) = \prod_{t=1}^{T} p(y_t | y_{t-1}) p(x_t | y_t)$$

- The model is generative because it models the distribution P(**y** , **x**)

Generative Models: Limitations

- Modelling p(x) is difficult because it may consist of many interdependent features
  - For example: In NER, word's identity may not be enough evidence, especially in case of '*Person*' category as many proper nouns may not occur in training data. It may be helpful to identify features like capitalization, neighboring words, suffix, etc.
- HMMs make the independence assumption (2) but that is not true above because, suffix and capitalization are highly dependent on the word's identity.
- Generative Models, in general, can be enhanced to model inter-dependencies between such features, but then modelling that becomes highly intractable.
- But, in the end, there is a definite mismatch between the desired learning objective and the prediction objective function.
- Discriminative models such as MEMM, CRF try to address this issue.

Discriminative Models: MEMM



- *Maximum Entropy Markov Models (MEMMs)* are discriminative models, where each state has an exponential model that takes the observation sequence as input and outputs a probability distribution over the next possible states.

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{T} P(y_t|y_{t-1}, x_t)$$

- Each of the $P(y_t|y_{t-1}, x_t)$, is an exponential model of the form:

$$P(y_t|y_{t-1}, x_t) = \frac{1}{Z(x_t, y_{t-1})} \exp(\sum_a \lambda_a f_a(x_t, y_t))$$

where Z is a normalization constant and the summation is over all features

MEMM: Limitations – Label Bias Problem

- MEMM suffers from Label Bias Problem, i.e., the transition probabilities of leaving a given state is normalized for only that state.

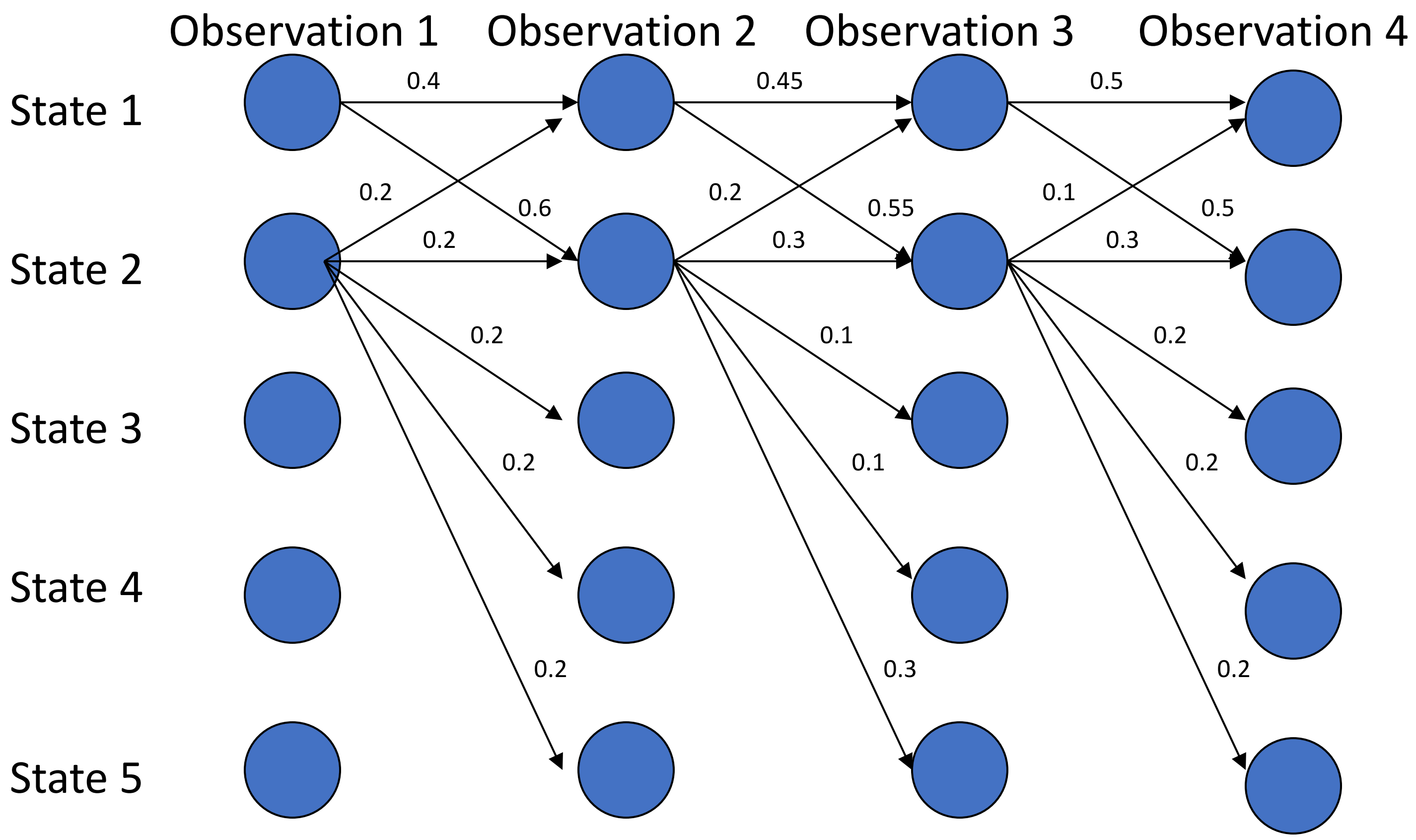
- Imagine that during the training a state *s* only saw state *s'* as the next state when given observation *o,* then according to the eq in previous slide:

$$P(s'|s, o\ ) = 1$$

this is because the normalization is done per state and not globally.

MEMM: Limitations – Label Bias Problem

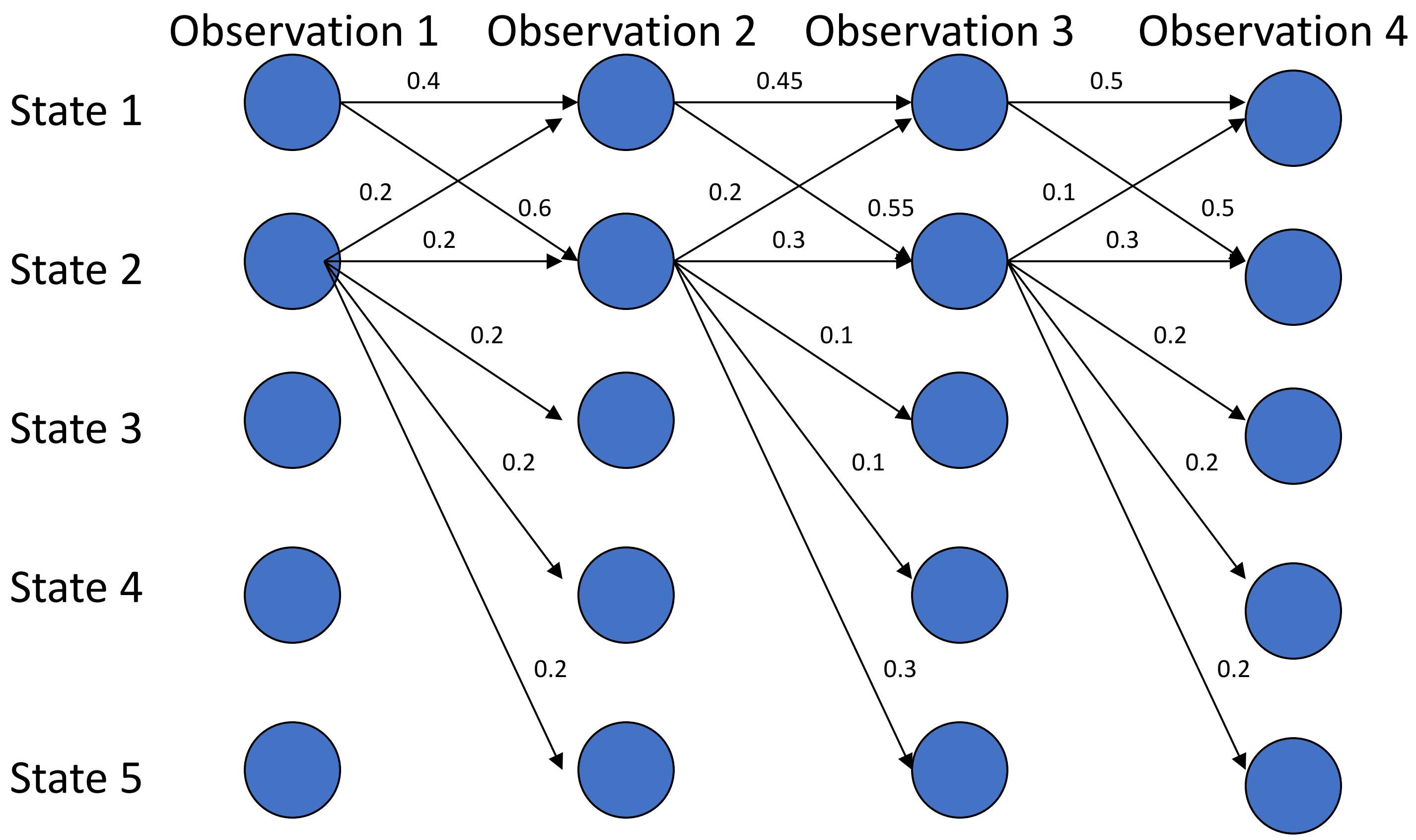
(example borrowed from Dr. Ramesh Nallapati's slides: http://www.cs.stanford.edu/~nmramesh/crf)



- We can observe in the diagram:
  - State 1 almost always intend to transit to State 2
  - State 2 almost always intend to stay in State 2

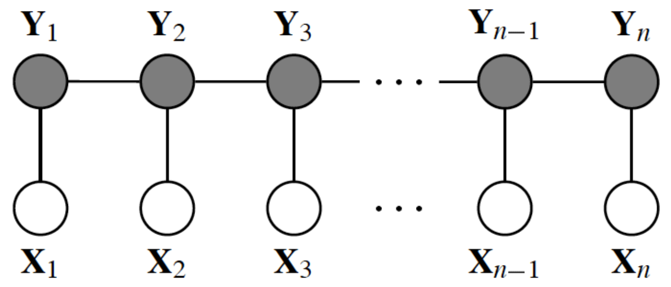
MEMM: Limitations – Label Bias Problem

(example borrowed from Dr. Ramesh Nallapati's slides: http://www.cs.stanford.edu/~nmramesh/crf)



- P(1->1->1->1) = 0.4*0.45*0.5 = 0.090
- P(2->2->2->2) = 0.2*0.3*0.3 = 0.018
- P(1->2->2->2) = 0.6*0.3*0.3 = 0.054
- P(2->1->1->1) = 0.2*0.45*0.5 = 0.450

Discriminative Models: CRF



- Conditional Random Fields (CRFs) overcomes the Label Bias problem
- **X:** random variable over the observation sequence
- **Y:** random variable over the label sequence
- **Def^n :** Let G=(V,E) be a graph such that $\mathbf{Y} = (\mathbf{Y}_{\mathbf{v}})_{v \in V}$ so that **Y** is indexed by vertices of G. Then (**X,Y**) is a conditional random field in case, when conditioned on **X**, the random variables $\mathbf{Y}_{\mathbf{v}}$ obey the Markov property with respect to the graph: $p(\mathbf{Y}_{\mathbf{v}} \mid \mathbf{X}, \mathbf{Y}_{\mathbf{w}}, w \neq v) = p(\mathbf{Y}_{\mathbf{v}} \mid \mathbf{X}, \mathbf{Y}_{\mathbf{w}}, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.
- These are **Linear Chain CRF**

Fundamental Theorem of Random Fields

- Given by **Hammersley and Clifford, 1971**, it states that the probability distribution of **x** satisfies the Markov property with respect to graph G(V,E) if and only if, it can be factored according to G:

$$P(\boldsymbol{x}) = \frac{1}{Z}\prod_{\mathcal{C}}\psi_{\mathcal{C}}$$

- where Z is the normalization constant and $\psi_{\mathcal{C}}$ is the potential function over clique $\mathcal{C}$.

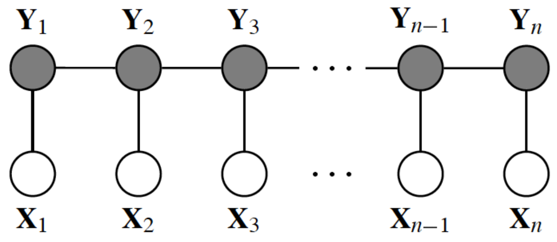
- $\log(\psi_{\mathcal{C}}) \triangleq \boldsymbol{\lambda}_{\mathcal{C}}^{T}\boldsymbol{f}(\mathcal{C})$, where $\boldsymbol{f}(.)$ is the feature vector defined over the clique and $\boldsymbol{\lambda}$ is the corresponding weight vector for those features.

$$P(\boldsymbol{x}) = \frac{1}{Z}\prod_{\mathcal{C}}\exp(\boldsymbol{\lambda}_{\mathcal{C}}^{T}\boldsymbol{f}(\mathcal{C})) = \frac{1}{Z}\exp(\sum_{\mathcal{C}}\boldsymbol{\lambda}_{\mathcal{C}}^{T}\boldsymbol{f}(\mathcal{C}))$$

CRF Notation Legend

- In the few of next slides, we use the following notation:
  - N: Number of training examples
  - Each training example is denoted by (**x**,**y**)
  - **x** = <$x_1, x_2, x_3, \ldots, x_T$>
  - **y** = <$y_1, y_2, y_3, \ldots, y_T$>
  - $\boldsymbol{x}_t^i$: tth term of ith **x**
  - $\boldsymbol{y}_t^i$: label of tth term of ith **x**
  - f (boolean edge feature) and g (boolean vertex feature) are feature functions.
  - l iterates over f features
  - m iterates over g features
  - v is a vertex from vertex set V ; e is an edge from edge set E
  - $y|_e$: components of y along the edge e
  - $y|_v$: components of y along the vertex v

Linear Chain CRF



- P(**Y**) can be factored into the distributions involving the cliques of the graph.
  - Although the graph contains **X** and **Y**, we only define the random field over **Y**
  - **X** are observables
- If the graph G forms a tree or simply a chain, then the cliques are the edges and vertices of the graph (G-X), and the probability distribution of **Y** takes the form:

$$P(\boldsymbol{y}|\boldsymbol{x}) \propto \exp\left(\sum_{e\epsilon E,l} \lambda_l f_l(e, \boldsymbol{y}|_e, \boldsymbol{x}) + \sum_{v\epsilon V,m} \mu_m g_m(v, \boldsymbol{y}|_v, \boldsymbol{x})\right)$$

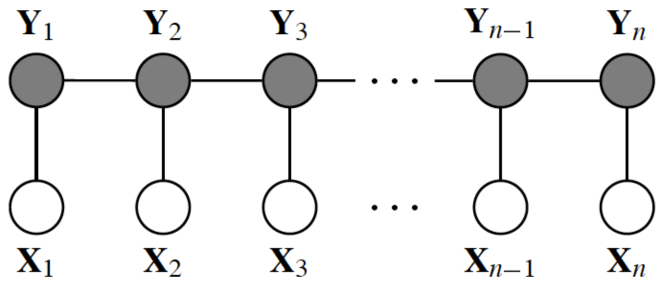
$$\propto \exp(\sum\nolimits_{i=1}^{N} \sum_{t=1}^{T} (\sum_{l} \lambda_l f_l\left(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^i\right) + \sum_{m} \mu_m g_m\left(y_t^{(i)}, \boldsymbol{x}\right)))$$

- We can normalize the above equation by the following constant:

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp(\sum\nolimits_{i=1}^{N} \sum_{t=1}^{T} (\sum_{l} \lambda_l f_l(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^i) + \sum_{m} \mu_m g_m(y_t^{(i)}, \boldsymbol{x})))$$

- So now, the normalization constant is only a function of observation sequence and not the current state. Hence, it solves the label bias problem.

Linear Chain CRF : Loss Function & Inference



- The loss function is the log likelihood of the probability distribution:

$$l(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^i) - \log(Z(\boldsymbol{x}^i))$$

- Here, all the features f and g have been written as f for convenience
- The loss function is concave , which follows from exponential model of the probability distribution
- This means that there is a global optimal point.
- **Inference**: To compute the most likely labelling, we compute y*:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, P(\boldsymbol{y}|\boldsymbol{x})$$

- Viterbi algorithm can be used to solve this!

Parameter Estimation: Gradient Ascent

- Looking at the nature of loss function, we can use gradient ascent algorithm to maximize the likelihood function $l(\theta)$:

$$\begin{aligned}\nabla_k l(\theta) &= \sum_{i=1}^{N} \sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \frac{1}{Z} \sum_{\tilde{y}} \exp(.) \sum_{i=1}^{N} \sum_{t=1}^{T} f_k\left(\tilde{y}_t^{(i)}, \tilde{y}_{t-1}^{(i)}, x_t^{(i)}\right) \\ &= \sum_{i=1}^{N} \sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \sum_{\tilde{y}} P(\tilde{y}|x,\theta) \sum_{i=1}^{N} \sum_{t=1}^{T} f_k\left(\tilde{y}_t^{(i)}, \tilde{y}_{t-1}^{(i)}, x_t^{(i)}\right) \\ &= \ E[f_k] - E_p[f_k]\end{aligned}$$

- Equating the gradient to zero, we note that the optima is reached when empirical expectation of feature f_k is equal to the expectation w.r.t. the model.
- However, finding the closed form solution for θ is not always possible.

Parameter Estimation: Iterative Scaling

- The main idea behind Iterative Scaling is that parameters are updated such that the new values are closer to the optima than before.
- If $\theta = (\lambda_1, \lambda_2, \dots, \lambda_l, \mu_1, \mu_2, \dots \mu_m)$ and $\Delta\theta = (\delta\lambda_1, \delta\lambda_2, \dots, \delta\lambda_l, \delta\mu_1, \delta\mu_2, \dots \delta\mu_m)$, then $\theta + \Delta\theta$ will result in a model with higher log likelihood.
- Authors, (in this paper) provide 2 algorithms to solve this:
  - Generalized Iterative Scaling (Algorithm S)
  - Improved Iterative Scaling (Algorithm T)
- However, the convergence is really slow.
  - In the experiments done by Lafferty et. al. on using CRF for POS tagging, they observed that CRF did not converge to the optima value even after 2000 iterations (starting from a uniform distribution)
  - On the other hand, MEMM was able to converge in ~100 iterations.
  - They, then tried the CRF parameters with the MEMM-optimal parameters as initial values and observed the algorithm to converge in 1000 iterations.

Parameter Estimation: Newton method

- Consider a multivariate function $l(\theta)$, where $\theta = (\lambda_1, \lambda_2, \ldots, \lambda_l)$. We want to choose $\Delta\theta = (\delta\lambda_1, \delta\lambda_2, \ldots, \delta\lambda_l)$ such that :

$$l(\theta + \Delta\theta) < l(\theta)$$

- We can use Taylor expansion to compute the value of l at points nearby θ_n (given that the function l is twice differentiable):

$$l(\boldsymbol{\theta}_n + \Delta\boldsymbol{\theta}) \approx l(\boldsymbol{\theta}_n) + \Delta\boldsymbol{\theta}^T \nabla l(\boldsymbol{\theta}_n) + \frac{1}{2}\Delta\boldsymbol{\theta}^T(\nabla^2 l(\boldsymbol{\theta}_n))\Delta\boldsymbol{\theta}$$

- Here ∇l is the gradient and $\nabla^2 l$ is the hessian of the function l
- We need to choose $\Delta\boldsymbol{\theta}$ to minimize $l(\boldsymbol{\theta}_n + \Delta\boldsymbol{\theta})$
- So, differentiating the eq^n above w.r.t. $\Delta\boldsymbol{\theta}$ and setting it to zero, we get:

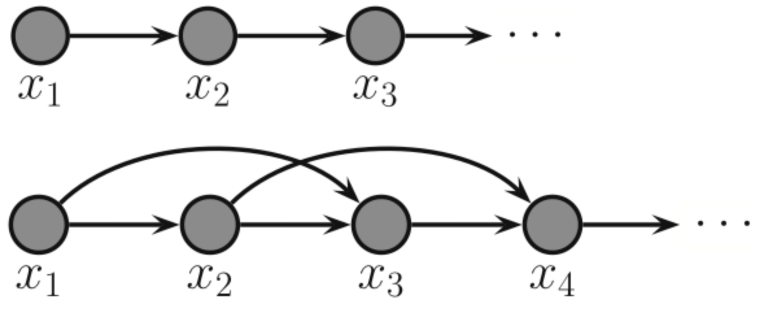
$$\Delta\boldsymbol{\theta} = -\boldsymbol{H}_{\boldsymbol{n}}^{-\boldsymbol{1}}\boldsymbol{g}_{\boldsymbol{n}}$$

where $\boldsymbol{H}_{\boldsymbol{n}}^{-\boldsymbol{1}}$ is the inverse of hessian matrix and $\boldsymbol{g}_{\boldsymbol{n}}$ is the gradient, both evaluated at $\boldsymbol{\theta}_n$

Parameter Estimation: Quasi-Newton method

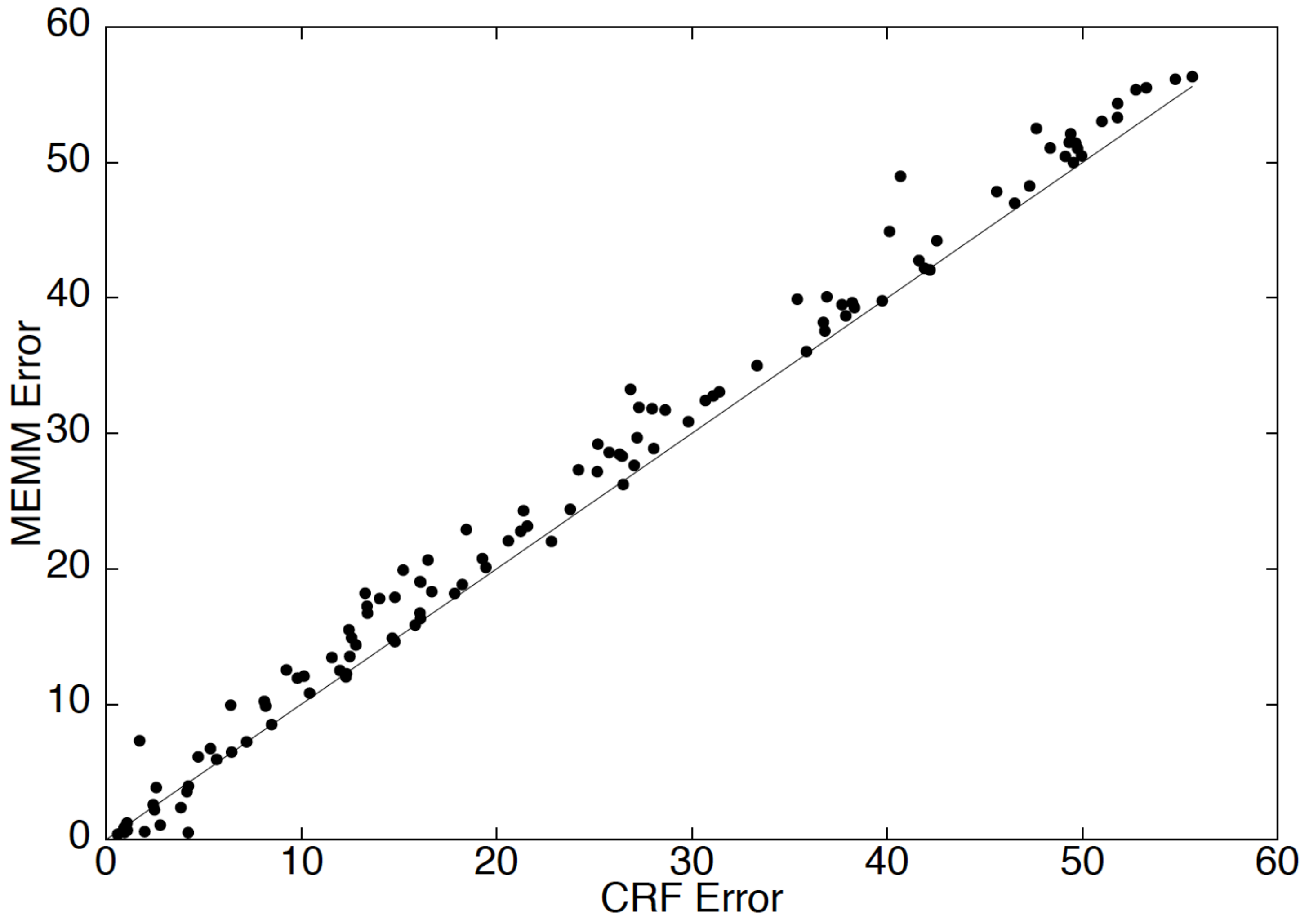
- Newton method requires the computation of inverse of Hessian matrix.
  - Its quadratic in size
  - Many problems use millions of features. Even storing the matrix can be a big issue.
- Idea in Quasi-Newton method is that instead of recalculating the hessian matrix at every point in the iteration, we can approximate the hessian. The approximation needs to qualify certain conditions:
  - Symmetricity: Hessian is a symmetric matrix since the order of differentiation is irrelevant
  - Secant Condition: $H_n(\boldsymbol{\theta_n} - \boldsymbol{\theta_{n-1}}) = (\boldsymbol{g_n} - \boldsymbol{g_{n-1}})$, that is, hessian is the ratio of the change in gradients w.r.t. the change in values, which is quite natural of the hessian
  - Positive semi-definiteness
- L-BFGS updates are applied to do an approximation constrained on the above conditions
- Advantage: The convergence is much faster.

Experiments – Mixed Order Sources

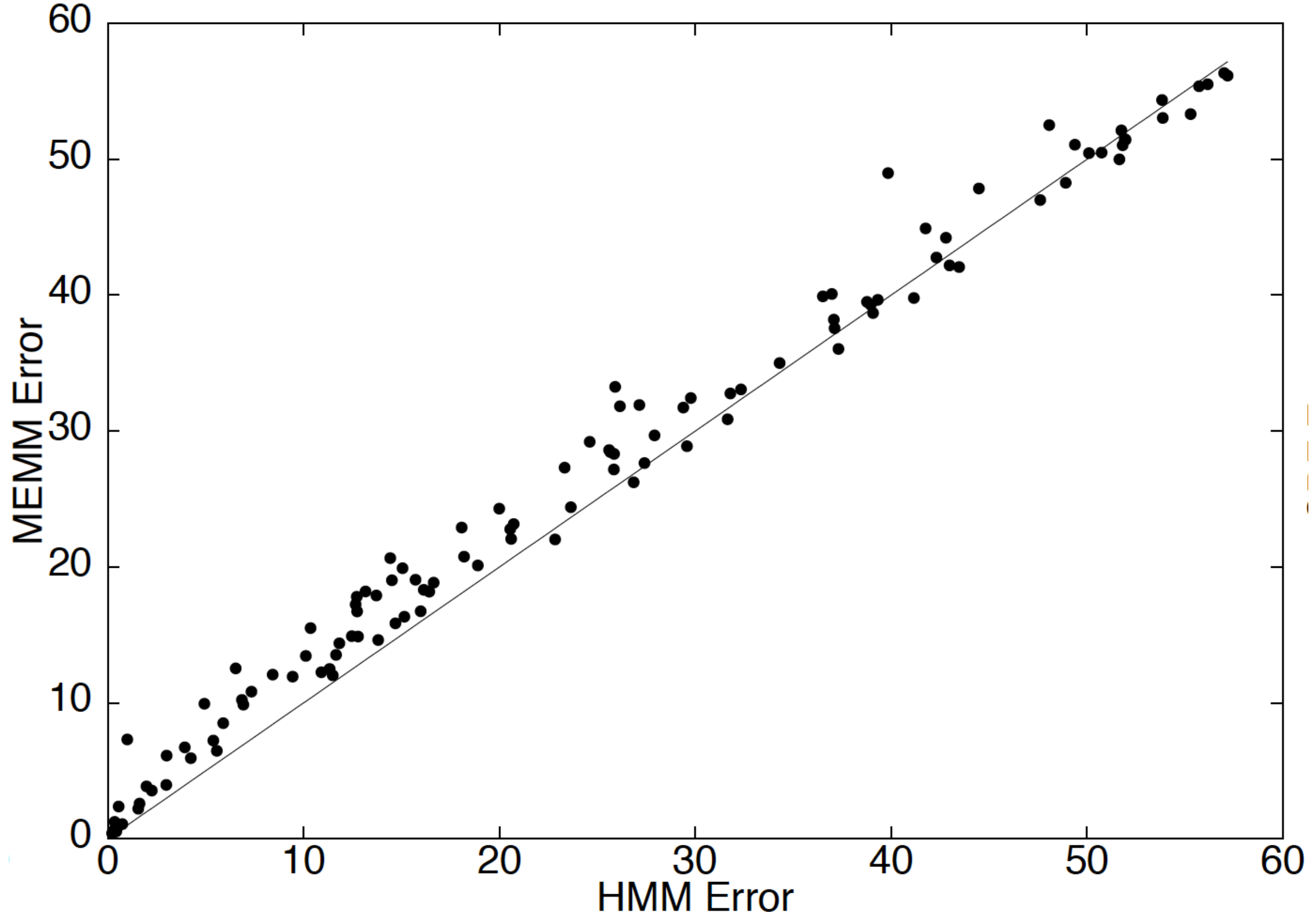


- Objective of experiment was to observe the performance of systems if the data is from a mixed order Markov chain.
  - HMM was used to generate data: set of 5 labels and 26 observation values
  - Transition Probability: $p_\alpha(y_i|y_{i-1}, y_{i-2}) = \alpha p(y_i|y_{i-1}, y_{i-2}) + (1-\alpha)p(y_i|y_{i-1})$
  - Emission Probability: $p_\alpha(x_i|y_i, x_{i-1}) = \alpha p(x_i|y_i, x_{i-1}) + (1-\alpha)p(x_i|y_i)$
  - They don't mention the initial probabilities
  - Many test sets were generated with different values of α
- Linear chain CRF with Generalized Iterative Scaling, MEMM and HMM was used to train the model

Experiments – Mixed Order Sources MEMM vs CRF

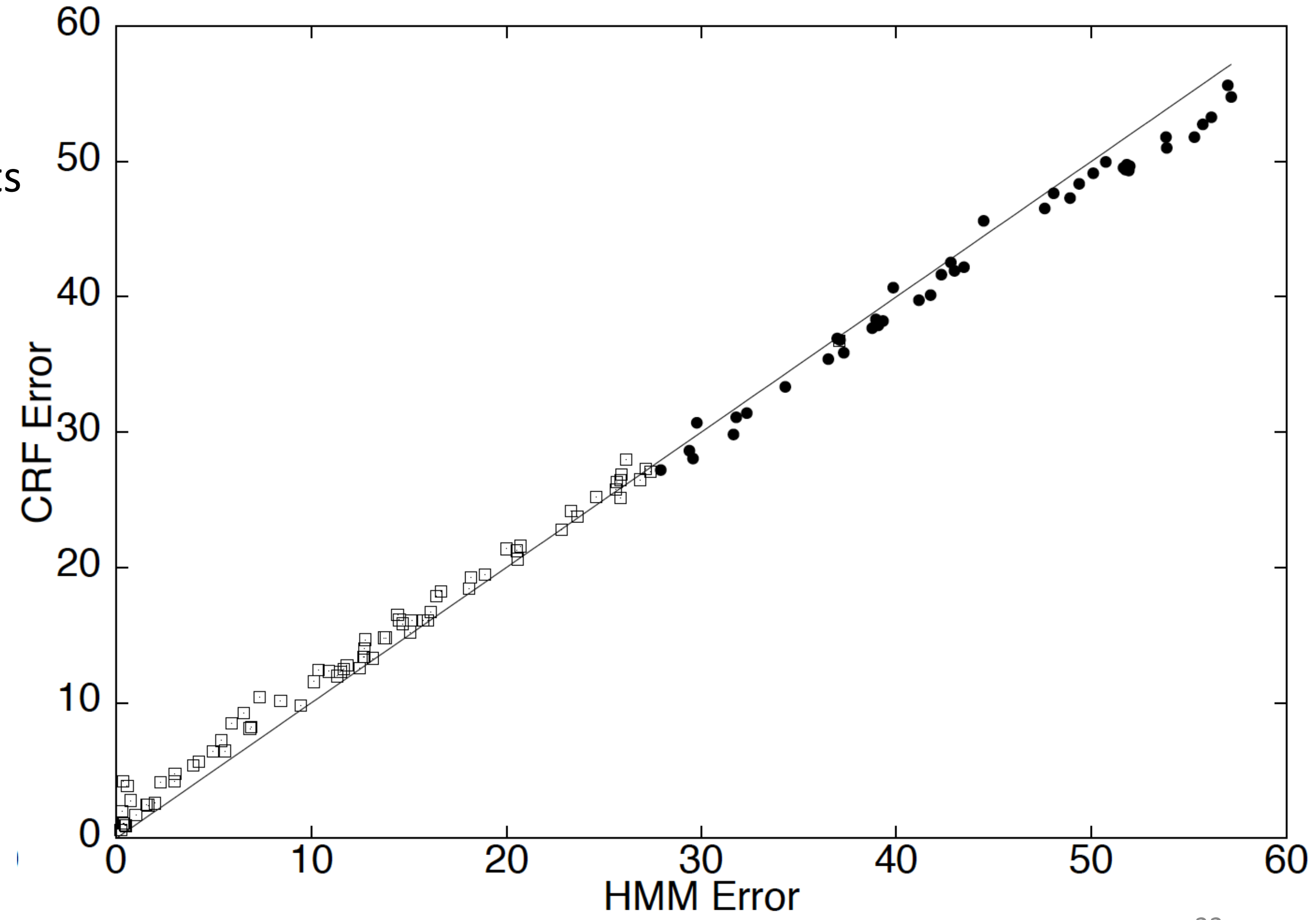


Experiments – Mixed Order Sources MEMM vs HMM



Experiments – Mixed Order Sources CRF vs HMM

- Square points represent test datasets that were generated with $\alpha < 0.5$ and solid circles represents test sets that were generated with $\alpha > 0.5$



Regularization

- A major thing missing in the Lafferty's paper was regularization.
- As we saw in previous slide, the optimal parameters are reached when the model expectation of a feature becomes equal to the empirical expectation of the feature.
- Thus, the model can over-fit to the training data.
- We can add a regularization term to the log likelihood equation to remedy this:

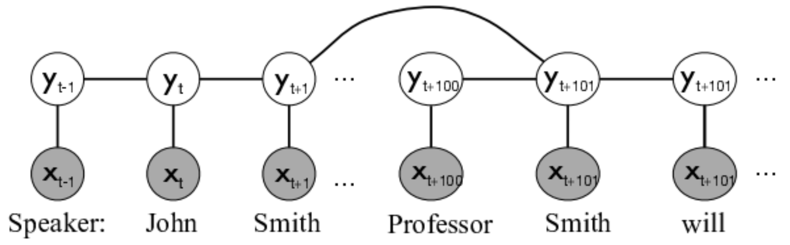
$$l(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_k \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^i) - \log\left(Z\left(\boldsymbol{x}^i\right)\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2}$$

- Here, we have used L2-regularizer. We can also use L1-regularizer or could follow structured sparsity approach by grouping features on a template-basis.

General CRF

- Instead of assuming the graph G to be linear, we can assume a more general graph.
    - Doing that, the definition of cliques would change in the slide 13 and we will take components of **y** corresponding to those cliques
- Parameter Estimation:
    - Gradient Descent and Iterative Scaling methods both require the calculation of P(y|x)
    - Its an NP-hard problem for a general graph
    - Approximation algorithms are required.
- Inference also suffers from the same problem as above.

Skip Chain CRF



- Linear Chain CRF make the assumption that labels follow the Markov property given the observation sequence.
- In tasks such as Information Extraction, it may be important for dependencies among labels for similar observations.
  - For example, the same name is mentioned multiple times in a document (non consecutively). We may want to link the states for these observation symbols.

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z}[\exp(\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{l}\lambda_l f_l\left(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^i\right)) + \exp(\sum_{i=1}^{N}\sum_{(u,v)\epsilon\mathcal{L}}\sum_{k}\mu_k g_k\left(y_u^{(i)}, y_v^{(i)}, \boldsymbol{x}^i\right))]$$

- The second term corresponds to skip-edges (edges between non consecutive states)
- The parameters can be estimated in the same way. However, inference becomes difficult.
  - Viterbi algorithm can no longer be applied.
  - Have to use approximation algorithms

Semi CRF

- In many problems such as NER, observation sequence is segmented such as proper names, locations, etc. and each segment needs to be labelled separately.
  - Night Watchmen stabbed Jon Snow: {(1,2,I), (3,3,O), (4,5,I)}: Each (t,u,y) means t: starting position ; u: ending position ; y: label for the segment
- Consider an observation sequence **x** and its corresponding segmentation **s** = <(t_i,u_i,y_i)>
- Define a vector $\boldsymbol{g} = < g^1, g^2, \ldots, g^K >$ of K feature functions each of which maps a particular segment *j* in **s** to a measurement $g^k(j, \boldsymbol{x}, \boldsymbol{s})$. Then,

$$P(\boldsymbol{s}|\boldsymbol{x},\boldsymbol{\theta}) = \frac{1}{Z}\exp\left(\boldsymbol{\theta}.\boldsymbol{G}(\boldsymbol{x},\boldsymbol{s})\right), \text{ where } \mathbf{G}(\mathbf{x},\mathbf{s}) = \sum_{j=1}^{|\boldsymbol{s}|} \boldsymbol{g}(j,\boldsymbol{x},\boldsymbol{s})$$

- Inference can be done by Viterbi Algorithm
- The parameters $\boldsymbol{\theta}$ can be estimated using Quasi-Newton methods

References

- J. Lafferty, A. McCallum, F. Pereira Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data ICML 2002
- C. Sutton and A. McCallum Introduction to Conditional Random Fields for Relational Learning In Statistical Relational Learning, 2007
- C. Sutton and A. McCallum Collective segmentation and labeling of distant entities in information extraction University of Massachusetts Amherst Dept. Of Computer Science, 2004.
- S. Sarawagi and W. Cohen Semi-Markov Conditional Random Fields for Information Extraction NIPS, 2005

Thank You