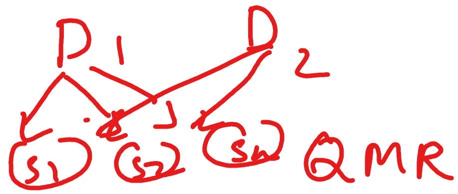
Learning Graphical Models from Data

- Learning Graph structure
- Learning Potentials given fixed graph

Graph Structure



1. Manual: Designed by domain expert
   - Used in applications where dependency structure is well-understood
   - Example: QMR systems, Kalman filters, Vision (Grids), HMM for speech recognition and IE.
2. Learned from examples
   - NP hard to find the optimal structure.
   - Widely researched, mostly posed as a branch and bound search problem.
   - Useful in dynamic situations

Parameters in Potentials

1. Manual: Provided by domain expert
   - Used in infrequently constructed graphs, example QMR systems
   - Also where potentials are an easy function of the attributes of connected graphs, example: vision networks.
2. Learned: from examples
   - More popular since difficult for humans to assign numeric values
   - Many variants of parameterizing potentials.
     1. Table potentials: each entry a parameter, example, HMMs
     2. Potentials: combination of shared parameters and data attributes: example, CRFs.

Learning potentials

Given sample of data D generated from a distribution represented by a graphical model with known structure G, learn potentials.

Two settings:

1. All variables observed or not.
   1. Fully observed: each training sample has all n variables observed.
   2. Partially observed: a subset of the variables are observed.
2. Potentials coupled with a log-partition function or not.
   1. No: Closed form solutions
   2. Yes: Potentials attached to arbitrary overlapping subset of variables in a UDGM. Example = edge potentials in a grid graph. Solve using gradient descent kind of iterative algorithms.

log Z

DOUBT

Learning potentials: Two settings

1. Generative:
   1. $P(\mathbf{x}) = P(x_1, \ldots, x_n)$ represented as a graphical model G.
   2. Samples are $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ $x^i = \{x^i_1, x^i_2, \ldots, x^i_n\}$
   3. Potentials to be learned are $\psi_C(\mathbf{x}_C)$
2. Conditional:
   1. $P(\mathbf{y}|\mathbf{x}) = P(y_1, \ldots, y_n|\mathbf{x})$ represented as a graphical model G over the $\mathbf{y}$ variables.
   2. Training samples are $D = \{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \ldots, (\mathbf{x}^N, \mathbf{y}^N)\}$
   3. Potentials to be learned are $\psi_C(\mathbf{y}_c, \mathbf{x}_C)$ $\psi_c(y_c, \cancel{x})$.

Most of the topics under learning we will discuss apply equally well to the two settings.

General framework for Parameter learning in graphical models

- Conditional distribution $\Pr(\mathbf{y}|\mathbf{x}, \theta)$, potentials are function of $\mathbf{x}$ and parameters θ to be learned.
- $\mathbf{y} = y_1, \ldots, y_n$ forms a graphical model: directed or undirected.
- Undirected:

$$\Pr(y_1, \ldots, y_n|\mathbf{x}, \theta) = \frac{\prod_C \psi_c(\mathbf{y}_c, \mathbf{x}, \theta)}{Z_\theta(\mathbf{x})}$$

$F_\theta(y_c, \theta, x) = \log \psi_c(y_c, x, \theta) \longrightarrow$

$$= \frac{1}{Z_\theta(\mathbf{x})} \exp(\sum_c F_\theta(\mathbf{y}_c, c, \mathbf{x}))$$

where $Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}'} \exp(\sum_c F_\theta(\mathbf{y}'_c, c, \mathbf{x}))$
clique potential $\psi_c(\mathbf{y}_c, \mathbf{x}) = \exp(F_\theta(\mathbf{y}_c, c, \mathbf{x}))$

Forms of $F_\theta(\mathbf{y}_c, c, \mathbf{x})$

- Log-linear model over user-defined features. E.g. CRFs, Maxent models, etc.
  Let K be number of features. Denote a feature as $f_k(\mathbf{y}_c, c, \mathbf{x})$. Then,

$$F_\theta(\mathbf{y}_c, c, \mathbf{x}) = \sum_{k=1}^{K} \theta_k f_k(\mathbf{y}_c, c, \mathbf{x})$$

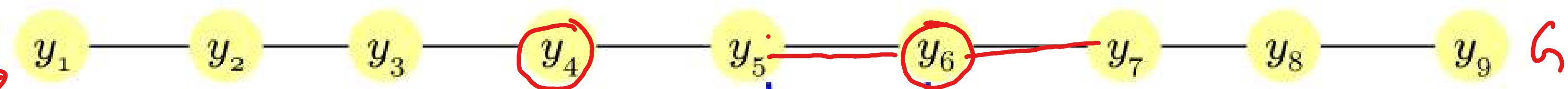
- Arbitrary function, e.g. a neural network that takes as input $\mathbf{y}_c, c, \mathbf{x}$ and transforms them possibly non-linearly into a real value. θ are the parameters of the network.

Example: Named Entity Recognition

My review of Fermat's last theorem by S. Singh

$y_i \in \{\text{Title}, \text{Author}, \text{Other}\}$.

t	1	2	3	4	5	6	7	8	9
x	My	review	of	Fermat's	last	theorem	by	S.	Singh
y	**Other**	**Other**	**Other**	**Title**	**Title**	**Title**	**other**	**Author**	**Author**



$$\mathbf{f}(y_i, y_{i-1}, i, \mathbf{x})$$

$f(\vec{y}_c, c, x)$

(y_i, y_{i-1})

$c = (i-1, i)$

Features decompose over adjacent labels.

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} \mathbf{f}(y_i, y_{i-1}, i, \mathbf{x})$$

Named Entity Recognition: Features

- Feature vector for each position

$$\mathbf{f}(y_i, \mathbf{x}, i, y_{i-1})$$

User provided

i-th label

Word i & neighbors

previous label

- Examples

$f_2(y_i, \mathbf{x}, i, y_{i-1}) = 1$ if y_i is Person & x_i is Douglas

$f_3(y_i, \mathbf{x}, i, y_{i-1}) = 1$ if y_i is Person & y_{i-1} is Other

$\rightarrow f_{K+1}(y_i, y_{i-1}, i, x) = 1$ if $y_i = y_{i-1}$ and 0 otherwise.

$\boxed{e_i} \in \mathbb{R}$

$\vec{f}_K(y_i, y_{i-1}, i, x) =$ embedding of the i^{th} word from BERT.

BER

$x_1 \quad x_2 \quad \ldots \quad x_n$

Training

Given

- N input output pairs $D = \{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \ldots, (\mathbf{x}^N, \mathbf{y}^N)\}$
- Form of F_θ
- Learn parameters θ by maximum likelihood.

$$\max_\theta LL(\theta, D) = \max_\theta \sum_{i=1}^{N} \log \Pr(\mathbf{y}^i | \mathbf{x}^i, \theta)$$

Training undirected graphical model

$$LL(\theta, D) = \sum_{i=1}^{N} \log \Pr(\mathbf{y}^i | \mathbf{x}^i, \theta)$$

$$= \sum_{i=1}^{N} \log \frac{1}{Z_\theta(\mathbf{x}^i)} \exp(\sum_c F_\theta(\mathbf{y}^i_c, c, \mathbf{x}^i))$$

$$= \sum_i [\sum_c F_\theta(\mathbf{y}^i_c, c, \mathbf{x}^i) - \log Z_\theta(\mathbf{x}^i)$$

The first part is easy to compute but the second term requires to invoke an inference algorithm to compute $Z_\theta(\mathbf{x}^i)$ for each i. Computing the gradient of the above objective with respect to θ also requires inference.

Training via gradient descent

Assume log-linear models like in CRFs where
$F_\theta(\mathbf{y}^i_c, c, \mathbf{x}^i) = \theta \cdot \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i_c, c)$ $= \sum_{k=1}^{K} \theta_k f_k(y^i_c, c, x^i)$ Also, for brevity write
$\mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) = \sum_c \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i_c, c)$

$$LL(\theta) = \sum_i \log \Pr(\mathbf{y}^i|\mathbf{x}^i, \theta) = \sum_i (\theta \cdot \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \log Z_\theta(\mathbf{x}^i))$$

Add a regularizer to prevent over-fitting.

$$\max_\theta \sum_i (\theta \cdot \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \log Z_\theta(\mathbf{x}^i)) - \|\theta\|^2/C$$

$\log Z_\theta(x^i) = \log \sum_{y'} \exp(\theta \cdot f(x^i, y'))$

Concave in $\theta \implies$ gradient descent methods will work.

$\nabla_\theta \log Z_\theta(x^i) = \dfrac{\sum_{y'} \exp(\theta \cdot f(x^i, y')) f(x^i, y')}{Z_\theta(x^i)}$

Gradient of the training objective

$\#(y') = 3^9$

$\to \exp(\sum_c \theta \cdot f(x^i, y'_c, c))$

$$\nabla L(\theta) = \sum_i \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \frac{\sum_{\mathbf{y}'} \mathbf{f}(\mathbf{y}', \mathbf{x}^i) \exp(\theta \cdot \mathbf{f}(\mathbf{x}^i, \mathbf{y}'))}{Z_\theta(\mathbf{x}^i)} - 2\theta/C$$

$$= \sum_i \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \sum_{\mathbf{y}'} \mathbf{f}(\mathbf{x}^i, \mathbf{y}') \Pr(\mathbf{y}'|\theta, \mathbf{x}^i) - 2\theta/C$$

$$= \sum_i \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - E_{\Pr(\mathbf{y}'|\theta, \mathbf{x}^i)} \mathbf{f}(\mathbf{x}^i, \mathbf{y}') - 2\theta/C$$

$$E_{\Pr(\mathbf{y}'|\theta, \mathbf{x}^i)} f_k(\mathbf{x}^i, \mathbf{y}') = \sum_{\mathbf{y}'} f_k(\mathbf{x}^i, \mathbf{y}') \Pr(\mathbf{y}'|\theta, \mathbf{x}^i)$$

$$= \sum_{\mathbf{y}'} \sum_c f_k(\mathbf{x}^i, \mathbf{y}'_c, c) \Pr(\mathbf{y}'|\theta, \mathbf{x}^i)$$

$$= \sum_c \sum_{\mathbf{y}'_c} f_k(\mathbf{x}^i, \mathbf{y}'_c, c) \Pr(\mathbf{y}'_c|\theta, \mathbf{x}^i)$$

$$= \sum_c \sum_{y'_c} f_k(x^i, y'_c, c) \sum_{y'_{\neg c}} P_r(y'|\theta, x^i)$$