Planning and Optimal Control

4. Markov Random Fields

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

Tue. 24.10.	(1)	1. Markov Models
Tue. 31.10.	—	— *Luther Day* —
Tue. 7.11	(2)	2. Hidden Markov Models
Tue. 14.11.	(3)	2b. (ctd.)
Tue. 21.11.	(4)	3. State Space Models
Tue. 28.11.	(5)	3b. (ctd.)
Tue. 5.12.	(6)	4. Markov Random Fields
Tue. 12.12.	(7)	5. Markov Decision Processes
Tue. 19.12.	(8)	6. Partially Observable Markov Decision Processes
Tue. 26.12.	—	— *Christmas Break* —
Tue. 9.1.	(9)	7. Reinforcement Learning
Tue. 16.1.	(10)	
Tue. 23.1.	(11)	
Tue. 30.1.	(12)	
Tue. 6.2.	(13)	

Outline

1. Markov Random Fields
2. Inference in MRFs
3. Learning MRFs
4. Partially Observed Markov Random Fields
5. Conditional Random Fields

Outline

1. Markov Random Fields
2. Inference in MRFs
3. Learning MRFs
4. Partially Observed Markov Random Fields
5. Conditional Random Fields

Motivation

- models for sequential data often naturally can be written using conditional density / probability functions conditioning on the past
  - e.g., Markov models of type $p(x_t \mid x_{t-1})$ or the latent state transition model $p(z_t \mid z_{t-1})$
- for other types of structured data there usually is no such marked direction
  - e.g., for images
- directed graphical models / Bayesian networks such as Markov Models and HMMs can be generalized to multidimensional data
  - multidimensional HMMs
  - require a direction to be marked, e.g., from top left to bottom right.
  - but it "feels" somewhat artificial

$\rightsquigarrow$ use undirected graphical models / Markov random fields

Stochastic Processes & Random Fields

Stochastic process / **random process** / **random function**:

- a collection of random variables X_i indexed by some **index set** I

$$\{X_i \mid i \in I\}$$

  - discrete-time: $I = \{a, a+1, a+2, \ldots, b\}$, $a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{\infty\}$
  - continuous-time: $I = [a, b]$, $a \in \mathbb{R} \cup \{-\infty\}, b \in \mathbb{R} \cup \{\infty\}$
  - **Random field**: $I \subseteq \mathbb{R}^K$ or a grid (spatial) or a graph.
- $=$ a density for structured data, on $\mathcal{X}^I$

Markov Random Fields

A random field p on I is called **Markov** if

- each variable is independent from all others given its neighbors

$$X_i \perp I \setminus N_i \setminus \{X_i\} \mid N_i$$
$$N_i := \{j \in I \mid j \text{ is a neighbor of } i \text{ in } I\}$$

Hammersley-Clifford Theorem

A random field p on I is **Markov** iff

- p factorizes into non-negative functions over maximal cliques in I:

$$\exists (q_c)_{c \in C} : p(x) = \prod_{c \in C} q_c(x_c)$$

$$C := \{c \subseteq I \mid c \text{ is a maximal clique}\}$$

  - q_c are called **potentials**.

Note: A set c of vertices is called a **clique** if all its nodes are linked in I.
A clique c is called **maximal**, if there is no clique d: $d \supsetneq c$.

Pairwise MRF

- potentials can be defined on any subsets of maximal cliques
  - but not on supersets
- most simple non-trivial potentials: on every edge

$$p(x) = \prod_{i,j \in I \text{ linked}} q_{i,j}(x_i, x_j)$$

  - **pairwise MRF**

Parametrizing Potentials I: Tables / Arrays

- potential functions q are parametrized
  - so that parameters θ can be learnt to fit the model to data
- if all variables in a potential q are discrete, the simplest parametrization is a table / a multidimensional array:

$$q(x_1, \ldots, x_K) = \theta_{x_1,\ldots,x_K}, \quad \theta \in (\mathbb{R}_0^+)^{\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \mathcal{X}_K}$$

  - example:

$\theta =$

$x_2 \setminus x_1$	red	green	blue
square	0.2	0.7	2.3
circle	0.5	0.0	0.2

- potentials are not normalized (generally do not sum to 1).
  - for a general graph, there would be no guarantee that the product of however normalized potentials again is normalized.

Example: Image Segmentation

- let $I = \{1, \ldots, N\} \times \{1, \ldots, M\}$ be the coordinates of the pixels of an $N \times M$ image
- let's define the graph on I to have an edge for neighboring pixels, i.e.,

$$(i,j) :\sim (i-1,j), (i+1,j), (i,j-1), (i,j+1)$$

- the state space $\mathcal{X} := \{\text{road}, \text{offroad}, \text{obstacle}\}$ are labels of the pixels denoting the type of object they belong to.
- here, the maximal cliques are just single edges
- an MRF could define its pairwise potentials via a table:

$q_{1,2}(x_1, x_2) =$

$x_1 \setminus x_2$	road	offroad	obstacle
road	0.9	0.1	0.2
offroad	0.1	0.9	0.01
obstacle	0.2	0..01	0.9

The Partition Function

- potentials usually are not normalized / sum to 1.
  - even if they would, for general graphs it would not guarantee that their product is normalized.

- an MRF with parametrized potentials therefore is represented via

$$p(x \mid \theta) = \frac{1}{Z(\theta)} \prod_{c \in C} q_c(x_c \mid \theta_c)$$

  - $Z(\theta)$ is called **partition function**

$$Z(\theta) := \sum_{x \in \mathcal{X}} \prod_{c \in C} q_c(x_c \mid \theta_c)$$

    - Z makes the MRF p a proper probability function / sum to 1.
    - Z in general depends on all parameters.
    - ... but on none of the x_i.

Parametrizing Potentials II: Features & Log-linear Models

- often array potentials do not work
  - e.g., because they have too many parameters
    if cliques are large or include nominal variables with many levels
  - cliques contain continuous variables
- alternative approach:
  1. define **features** $\phi(x_1, \ldots, x_K)$ for the variables of a potential q
  2. define the potential as a **log-linear model** in the features:

$$
\begin{aligned}
q(x_1, \ldots, x_K \mid \theta) &:= e^{\theta^T \phi(x_1,\ldots,x_K)} \\
&= e^{\sum_{\ell=1}^{L} \theta_\ell \phi_\ell(x_1,\ldots,x_K)}
\end{aligned}
$$

  - aka **maximum entropy model**, **maxent model**

$$
\log p(x \mid \theta) = \sum_c \theta_c^T \phi_c(x_c) + \log Z(\theta)
$$

Example: Image Segmentation (ctd.)

- let's define the graph on I to have an edge for pixels up to L1-distance 2, i.e.,

$$
(i,j) :\sim \quad (i-2,j) \quad \begin{matrix} (i-1,j-1) \\ (i-1,j) \\ (i-1,j+1) \end{matrix} \quad \begin{matrix} (i,j-2) \\ (i,j-1) \\ \\ (i,j+1) \\ (i,j+2) \end{matrix} \quad \begin{matrix} (i+1,j-1) \\ (i+1,j) \\ (i+1,j+1) \end{matrix} \quad (i+2,j)
$$

- now maximal cliques are a pixel (i,j) and its four distance 1 neighbors

- instead we could define features, e.g.,
  the frequency of each label in the neighborhood:

$$
\begin{aligned}
\phi(x_c)_1 &:= \text{frequency of road in } x_c \\
\phi(x_c)_2 &:= \text{frequency of offroad in } x_c \\
\phi(x_c)_3 &:= \text{frequency of obstacle in } x_c
\end{aligned}
$$

- and potentials as log-linear model in these features:

$$
q_c(x_c \mid \theta) := e^{\theta_1 \phi(x_c)_1 + \theta_2 \phi(x_c)_2 + \theta_3 \phi(x_c)_3}
$$

Tables as Special Case of Log-Linear Models

- if we define a binary indicator feature for each joint variable value:

$$\phi(x_1, \ldots, x_K) = (\mathbb{I}((x_1, \ldots, x_K) = (x'_1, \ldots, x'_K)))_{(x'_1, \ldots, x'_K) \in \mathcal{X}^K}$$

then the log-linear model is just the array potential.

Parametrizing Potentials III: Parameter Sharing

- often different potentials describe the same relation, just between different sets of variables
  - e.g., $q_{1,2}$ and $q_{5,17}$ describe the relation between a pixel and its neighbors, but for different image patches
    - one centered at (1,2), the other at (5,17)

- such potentials (and their parameters) often can be shared

$$q_c(x_c \mid \theta_c) = q(x_c \mid \theta)$$

- example: image segmentation
  - usually potentials will not depend on the reference pixel, but all be shared.

- parameter sharing allows to roll-out a MRF to graphs of different size
  - e.g., images of different width and height
  - MRF with shared parameters define MRF templates

Outline

MRF Inference

Inference in MRF (and generally graphical models) requires work:

- exact inference:
  - joint tree algorithm
  - simpler (less efficient) algorithm:
    - variable elimination / bucket elimination
- approximate inference:
  - variational inference
  - inference via sampling / Monte Carlo inference

Variable eliminiation

- idea:
  - marginalize out one non-target variable X_i at a time
  - collect all potentials containing this variable
  - ... and replace them by their product
    - summing over all possible values for X_i
    - materializing the product as array

Variable eliminiation / Algorithm

infer-mrf-varelim$(v, (q_c)_{c \in C})$:
  while $\bigcup_{c \in C} c \setminus v \neq \emptyset$:
    choose $i \in \bigcup_{c \in C} c \setminus v$ arbitrarily
    $(q, C) :=$ eliminate-variable(i, q, C)
  $p := \prod_{c \in C} q_c$
  $p :=$ normalize(p)
  return p
eliminate-variable$(i, (q_c)_{c \in C}, C)$:
  $C' := \{c \in C \mid i \in c\}$
  $C' := C' \cup \{c \in C \mid c \subseteq C'\}$
  $c' := \bigcup_{c \in C'} c \setminus \{x\}$
  $q_{c'} := (\sum_{x_i \in \mathcal{X}_i} \prod_{c \in C'} q_c(x_i, (x_{c'})_{c \cap c'}))_{x_{c'} \in \mathcal{X}_{c'}}$
  return $((q_{C \setminus C'}, q_{c'}), C \setminus C' \cup \{c'\})$

where

- $v \subseteq I$ target variables to infer marginal of
- $(q_c)_{c \in C}$ MRF defined by a set of potentials on $c \subseteq I$

yields $(p_{x_v})_{x_v \in \mathcal{X}_v}$ marginal of variables v

Inference / Variable eliminiation / Example

- $I := \{A, B, C, D, E, F\}$
- $C := \{\{A\}, \{A, B\}, \{A, C\}, \{B, D\}, \{B, C, E\}, \{C, F\}, \{F\}\}$
- $v := \{D\}$

- elimination sequence: F, E, C, A, B

Inference / Variable eliminiation / Example

- $I := \{A, B, C, D, E, F\}$
- $C := \{\{A\}, \{A, B\}, \{A, C\}, \{B, D\}, \{B, C, E\}, \{C, F\}, \{F\}\}$
- $v := \{D\}$

- elimination sequence: F, E, C, A, B

- compute:

$$q(C) := \sum_F q(C, F)\, q(F)$$

$$q(B, C) := \sum_E q(B, C, E)\, q(C)$$

$$q(A, B) := \sum_C q(B, C, E)\, q(C)\, q(A, B)\, q(A)$$

$$q(B, D) := \sum_A q(A, B)\, q(B, D)$$

$$q(D) := \sum_B q(B, D)$$

Infering Conditional Probabilities $p(A \mid B = b)$

- in general, A and B could denote sets/vectors of variables:

$$p(X_{i_1}, X_{i_2}, \ldots, X_{i_N} \mid X_{j_1} = b_1, X_{j_2} = b_2, \ldots, X_{j_M} = b_M\}$$
$$A = (X_{i_1}, X_{i_2}, \ldots, X_{i_N})$$
$$B = (X_{j_1}, X_{j_2}, \ldots, X_{j_M})$$
$$b = (b_1, \ldots, b_M)$$

- for each conditioning variable / value pair $(B_m, b_m) = (X_{j_m}, b_m)$ add an **evidence potential** epd_{j_m, b_m}:

$$\text{epd}_{i,b} : \mathcal{X}_i \rightarrow \mathbb{R}_0^+$$
$$x \mapsto \mathbb{I}(x = b)$$

- infer marginal of A for the potentials

$$p' := p \cup \{\text{epd}_{i,b} \mid (i, b) \in \text{zip}(B, b)\}$$

Note: $zip(A, B) := \{(A_i, B_i) \mid i = 1, \ldots, |A|\}$ for two sequences $A \in \mathcal{X}^*, B \in \mathcal{Y}^*$ of equal length.

Infering Conditional Probabilities / Example

- let us model the following rules:
  - if there is precipitation, roads are three times more likely to be slippery.
  - if there is frost, roads are two times more likely to be slippery.

- A: There is heavy precipitation.
  B: There is frost.
  C: Roads are slippery.

$$q(A, C) = \begin{pmatrix} 0.5 & 0.5 \\ 0.25 & 0.75 \end{pmatrix}, \quad q(B, C) = \begin{pmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \end{pmatrix}$$

- What are the chances of the road to be slippery if there is precipitation, but no frost?

$$p(C \mid A = 1, B = 0)$$

Infering Conditional Probabilities / Example

- initial potentials:

$$q(A,C) = \begin{pmatrix} 0.5 & 0.5 \\ 0.25 & 0.75 \end{pmatrix}, \quad q(B,C) = \begin{pmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \end{pmatrix},$$

$$q(A) = \text{epd}_{A,1}(A) = \begin{pmatrix} 0 & 1 \end{pmatrix}, \quad q(B) = \text{epd}_{B,0}(B) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

- eliminate A:

$$q(C) = \sum_A q(A,C)q(A) = \begin{pmatrix} 0.5 & 0.75 \end{pmatrix}$$

- eliminate B:

$$q'(C) = (\sum_A q(B,C)q(B))q(C)) = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix} \odot \begin{pmatrix} 0.5 & 0.75 \end{pmatrix}$$

$$= \begin{pmatrix} 0.25 & 0.375 \end{pmatrix}$$

$$\text{normalization}(q')(C) = \begin{pmatrix} 0.4 & 0.6 \end{pmatrix}$$

Outline

Learning Maxent Models via Gradient Descent

- gradients for maxent models are straight-forward to derive:

$$\ell(\theta; x) := \log p(x \mid \theta) = \sum_c \theta_c^T \phi_c(x_c) - \log Z(\theta)$$

$$\nabla_{\theta_c} \ell(\theta; x) = \phi_c(x_c) - \nabla_{\theta_c} \log Z(\theta)$$

$$Z(\theta) := \sum_{x \in \mathcal{X}} \prod_{c \in C} e^{\theta_c^T \phi_c(x_c)}$$

$$\nabla_{\theta_c} \log Z(\theta) = \frac{1}{Z(\theta} \sum_{x \in \mathcal{X}} \prod_{c \in C} e^{\theta_c^T \phi_c(x_c)} \phi_c(x_c)$$

$$= \sum_{x \in \mathcal{X}} p(x \mid \theta)\, \phi_c(x_c) = \mathbb{E}(\phi_c(X_c))$$

$$\rightsquigarrow \quad \nabla_{\theta_c} \ell(\theta; x) = \phi_c(x_c) - \mathbb{E}(\phi_c(X_c))$$

- but it requires inference in the model to compute $\mathbb{E}(\phi_c(X_c))$!

Infering $\mathbb{E}(\phi_c(X_c))$

- infer marginal $p(X_c)$
- compute array $(\phi_c(X_c))_{x_c \in \mathcal{X}_c}$
- sum all cells of the elementwise tensor product $p(X_c)\phi_c(X_c)$

$$\mathbb{E}(\phi_c(x_c)) = \sum_{x_c \in \mathcal{X}_c} p(x_c)\phi_c(x_c)$$

Learning Maxent Models via Gradient Descent

```
learn-mrf-gd(x, (q_c)_{c∈C}, η, K, ε):
  for c ∈ C: θ_c := 1_{Θ_c}
  for k := 1 : K:
    for c ∈ C: f_c := 0
    for n = 1 : N:
      for c ∈ C:
        f_c += φ(x_{n,c})/N
    for c ∈ C:
      p_c := infer-mrf(c, (q_c(θ_c))_{c∈C})
      g_c := 0
      for v ∈ 𝒳^C:
        g_c += p_c(v) · φ(v)
      Δθ_c := f_c − g_c
    if Σ_c ||Δθ_c||_2 < ε:
      return (θ_c)_{c∈C}
    for c ∈ C:
      θ_c := θ_c − ηΔθ_c
  return ''non converged in K steps''
```

where

- $x \in (\mathcal{X}^I)^*$ data
- $(q_c)_{c\in C}$ potentials of cliques, having parameters $\theta_c \in \Theta_c$
- $C \subseteq 2^I$ variables of the potentials / maximal cliques of graph I
- η steplength
- K maximal number of iterations
- ϵ minimum gradient norm

yields $(\theta_c)_{c\in C}$ parameters of the potentials

Optimality Criterion: Matching Moments

$$\ell(\theta; x_{1:N}) := \frac{1}{N}\sum_{n=1}^{N} \log p(x_n \mid \theta)$$

$$\nabla_{\theta_c}\ell(\theta; x) = \frac{1}{N}\sum_{n=1}^{N} \phi_c(x_{n,c}) - \nabla_{\theta_c} \log Z(\theta)$$

$$= \mathbb{E}_{p_{\text{emp}}}(\phi_c(x_c)) - \mathbb{E}_p(\phi_c(x_c))$$

thus at $\nabla_{\theta_c}\ell(\theta; x) = 0$:

$$\mathbb{E}_{p_{\text{emp}}}(\phi_c(x_c)) = \mathbb{E}_p(\phi_c(x_c))$$

- **moment matching**

Learning Maxent Models via Iterative Proportional Fitting

- for array potentials

$$\mathbb{E}_p(\phi_c(x_c)) = \mathbb{E}_p(\mathbb{I}(x_c = x'))_{x' \in \mathcal{X}_c} = p(x_c \mid \theta) \propto \theta_{c,x_c}$$

$$\mathbb{E}_{p_{\text{emp}}}(\phi_c(x_c)) = \mathbb{E}_{p_{\text{emp}}}(\mathbb{I}(x_c = x'))_{x' \in \mathcal{X}_c} = p_{\text{emp}}(x_c) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}(x_{n,c} = x_c$$

- fixpoint iteration:

$$\theta_{c,x_c}^{(t+1)} = \theta_{c,x_c}^{(t)} \frac{p_{\text{emp}}(x_c)}{p(x_c \mid \theta^{(t)})}$$

- approximate inference

Learning Maxent Models via Iterative Proportional Fitting

learn-mrf-ipf$(x, (q_c)_{c \in C})$:
  for $c \in C$:
    $\theta_c := 1_{\Theta_c}$
    $p_{\text{emp},c} := \left(\frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(x_{n,c} = x'_c)\right)_{x'_c \in \mathcal{X}_c}$
  repeat
    for $c \in C$:
      $p :=$ infer-mrf$(c, (q_c(\theta_c))_{c \in C})$
      for $x_c \in \mathcal{X}_c$:
        $\theta_{c,x_c} := \theta_{c,x_c} \frac{(p_{\text{emp},c})_{x_c}}{p_{x_c}}$
  until convergence
  return $(\theta_c)_{c \in C}$

where

- $x \in (\mathcal{X}^I)^*$ data
- $(q_c)_{c \in C}$ potentials of cliques, having parameters $\theta_c \in \Theta_c$
- $C \subseteq 2^I$ variables of the potentials / maximal cliques of graph I

yields $(\theta_c)_{c \in C}$ parameters of the potentials

Outline

Learning via EM Algorithm

Learning from complete data we just discussed in the last section.

For incomplete data use EM:

- E-step: complete the data using inference
  - inference for every instance individually
  - joint marginals for variables cooccurring in the same clique/potential
  - every instance is split into possible completions
    - the probability of the completion figures as caseweight for the M-step
    - possibly different splittings for every clique
- M-step: update parameters θ using a method for learning from complete data.
  - e.g., gradient descent

Case weight for joint completions $\mathcal{X}_{c\cap Z}$ of instance x:

$$w_{c,x} := p(c \cap Z \mid X = x_c), \quad c \in C, x \in \mathcal{X}$$

where

$$X := (X_1, \ldots, X_M) \qquad \text{predictors}$$

$$Z := (Z_1, \ldots, Z_K) \qquad \text{latent variables}$$

$$\nabla_{\theta_c} \ell(\theta; x) = \phi_c(x_c) - \mathbb{E}(\phi_c(X_c))$$

$$\rightsquigarrow \quad \nabla_{\theta_c} \ell(\theta; x, z) = \sum_{z_c \in \mathcal{X}_{c\cap Z}} w_{c,x} \left(\phi_c(x_c, z_c) - \mathbb{E}(\phi_c(X_c, z_c))\right)$$

or?

$$\rightsquigarrow \quad \nabla_{\theta_c} \ell(\theta; x, z) = \left(\sum_{z_c \in \mathcal{X}_{c\cap Z}} w_{c,x} \phi_c(x_c, z_c) \right) - \mathbb{E}(\phi_c(X_c, Z_c))$$

Outline

The Sequence Labeling Problem

Given data $\mathcal{D}^{\text{train}}$ of N pairs (x_n, y_n) of sequences $x_n \in \mathcal{X}^*, y_n \in \mathcal{Y}^*$ of same length,

- x_n called **predictor sequence**,
- y_n called **target sequence**

and a loss function $\ell : \mathcal{Y}^* \times \mathcal{Y}^* \to \mathbb{R}$, learn the parameters θ of a model

$$p(y \mid x, \theta)$$

s.t. for yet unseen data $\mathcal{D}^{\text{test}}$ the loss

$$\ell(\hat{y}; \mathcal{D}^{\text{test}}) = \frac{1}{|\mathcal{D}^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}^{\text{test}}} \ell(y, \hat{y}(x))$$

is minimal.

The Sequence Labeling Problem / Example

Part of speech tagging:

- predictor sequence x: words of a sentence.
  - e.g., *At the banks Jim is catching a big fish.*
- target sequence y: part of speech classes of each word.
  - e.g., *pre art N N V V art adj N*
  - a label for each element of the sequence:

At	*the*	*banks*	*Jim*	*is*	*catching*	*a*	*big*	*fish.*
|	|	|	|	|	|	|	|	|
pre	*art*	*N*	*N*	*V*	*V*	*art*	*adj*	*N*

- usually 9 different POS classes/tags/labels for English:

noun:	*car*	pronoun:	*she*	adjective:	*yellow*
verb:	*to drive*	adverb:	*gracefully*	preposition:	*under*
conjunction:	*and*	interjection:	*hurray*	article:	*the*

Label Sequencing Models 1: HMMs

- model targets y_t by hidden states z_t,
  predictors x_t by observations x_t.

$$p(x_{1:T}, y_{1:T} \mid \theta) = p(y_1 \mid \theta) \prod_{t=2}^{T} p(y_t \mid y_{t-1}, \theta) \prod_{t=1}^{T} p(x_t \mid y_t, \theta)$$

- learning:
  - simple, from fully observed data.
- prediction:
  - compute MAP $p(z_{1:T} \mid x_{1:T})$ (decoding)
- but HMMs are generative models
  - spend data to learn generative models of the predictors x_t
  - like Linear Discriminant Analysis vs. Logistic Regression

Label Sequencing Models 2: MEMMs

- Maximum entropy markov model (MEMM)

$$p(y_{1:T} \mid x_{1:T}, \theta) = p(y_1 \mid x_{1:T}, \theta) \prod_{t=2}^{T} p(y_t \mid y_{t-1}, x_{1:T}, \theta)$$

  - $=$ Markov chain with state transition conditionend on all predictors
- but y_t does not depend on future predictors $x_{t+1:T}$
  - y_t and x_{t+1} are d-separated by v-connection at y_{t+1}.
  - in the POS example, $x_9 =$*fish* would not allow to recognize $x_3 =$*banks* as noun (riverbank) instead of as verb (to bank in the financial sense).
  - called “label bias problem”

Label Sequencing Models 3: CRFs

- Conditional Random Fields (CRFs)

$$p(y_{1:T} \mid x_{1:T}, \theta) = \frac{1}{Z(x_{1:T}, \theta)} \prod_{t=1}^{T} q(y_t \mid x_{1:T}, \theta) \prod_{t=2}^{T} q(y_t, y_{t-1} \mid x_{1:T}, \theta$$

  often with log-linear potentials

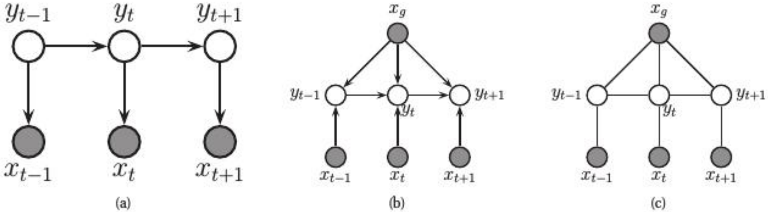
$$q(y_t \mid x_{1:T}, \theta) = e^{\theta_{(t)}^T \phi(x_{1:T}, y_t, y_{t-1})}$$

$$q(y_t, y_{t-1} \mid x_{1:T}, \theta) = e^{\theta_{(t,t-1)}^T \phi(x_{1:T}, y_t, y_{t-1})}$$

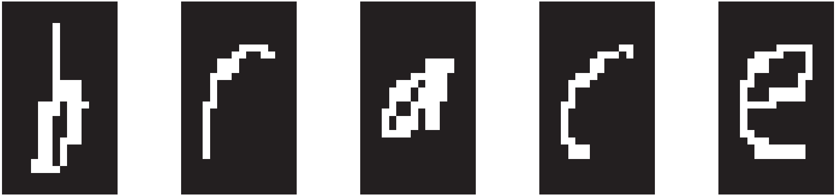
  - = MRF with potentials depending on all predictors

- in CRFs, y_t does depend on $x_{t+1:T}$ (through y_{t+1})
  - because $q(y_{t+1}, y_t)$ is not conditioned on y_t as $p(y_{t+1} \mid y_t)$ is.

HMM vs MEMM vs CRF



Example: Handwriting Recognition



(a) (b) (c) (d) (e)

[source: ?, p.686]

- recognize handwritten texts

$$q(y_t \mid x_{1:T}, \theta) := q(y_t \mid x_t, \theta_1) := \text{deep neural network for letters}$$
$$q(y_t, y_{t-1} \mid x_{1:T}, \theta) := q(y_t, y_{t-1} \mid \theta_2) := \text{language bigram model}$$

Conditional Random Fields

- many CRFs are chain-structured as the ones discussed
- CRFs can be defined more generally on arbitrary targets y structured by a graph I:

$$p((y_i)_{i\in I} \mid x, \theta) = \frac{1}{Z(x,\theta)} \prod_{c\in C} q(y_c \mid x, \theta)$$

  often with log-linear potentials

$$q(y_c \mid x, \theta) = e^{\theta_c^T \phi(x, y_c)}$$

  - $=$ MRF with potentials depending on all predictors

Learning CRFs via Gradient Descent

- gradients for CRFs are straight-forward to derive:

$$\ell(\theta; y, x) := \log p(y \mid x, \theta) = \sum_c \theta_c^T \phi_c(y_c, x) - \log Z(x, \theta)$$

$$\nabla_{\theta_c} \ell(\theta; y, x) = \phi_c(y_c, x) - \nabla_{\theta_c} \log Z(x, \theta)$$

$$Z(x, \theta) := \sum_{y \in \mathcal{Y}} \prod_{c \in C} e^{\theta_c^T \phi_c(y_c, x)}$$

$$\nabla_{\theta_c} \log Z(x, \theta) = \frac{1}{Z(\theta} \sum_{y \in \mathcal{Y}} \prod_{c \in C} e^{\theta_c^T \phi_c(y_c, x)} \phi_c(y_c, x)$$

$$= \sum_{y \in \mathcal{Y}} p(y \mid x, \theta) \phi_c(y_c, x) = \mathbb{E}(\phi_c(y_c, x))$$

$$\rightsquigarrow \quad \nabla_{\theta_c} \ell(\theta; y, x) = \phi_c(y_c, x) - \mathbb{E}(\phi_c(y_c, x))$$

- requires N inferences in the model to compute $\mathbb{E}(\phi_c(y_{n,c}, x_n))$!

Summary (1/3)

- **Random fields** / **stochastic processes** are **densities for structured data**
  - represented by a set of random variables indexed by a **(undirected) graph**.
- **Markov random fields**
  - each variable is **independent from all others given its neighbors**
    or equivalently
  - decompose in a product over the **maximal cliques**.
    - clique factors are called **potentials**.
- Potentials usually are parametrized:
  - **parametrized as arrays**:
    - an array with a value for every combination of values of the variables.
  - **parametrized by features and a log-linear model**:

$$q(x_{1:K} \mid \theta) = e^{\theta^T \phi(x_{1:K})}$$

  - **parameter sharing** for potentials describing the same relation between different instances / sets of variables

Summary (2/3)

- The **partition function** enforces the marginal of the product of potentials to be 1.
  - depending on all parameters
  - it usually is given only **implicitly** as sum over all possible instances and thus cannot be computed but for very simple models.
- A simple method for **inference** in MRFs is **variable eliminiation**.
  - marginalize out one non-target variable at a time
  - multiplying all potentials containing this variable
  - observed variables are represented by **evidence potentials**.
- MRFs can be **learned** by **gradient descent**.
  - due to the partition function requires inference of the expected features
  - one inference per gradient step (and clique/potential)

Summary (3/3)

- Partially observed MRFs may contain **latent variables**.
  - can be learned by EM.
  - M-step: gradient descent as for fully observed MRFs.
  - E-step: infer distribution of latent variables
    - for each clique containing a latent variable
    - joint distribution per clique
    - requires N inferences per EM step (and affected clique/potential)
- **Conditional random fields** make **potentials depend on the predictors**.
  - to ensure that a target can depend on future observations (for the sequence labeling problem; "label bias problem").
  - also can be learned by gradient descent as well.
  - also require N inferences per gradient step (and clique/potential)

Further Readings

- Markov random fields:
  - **?**, chapter 19.

References