Undirected graphical models (Markov random fields)

Liyuan Hu, Wenxiang Luo

School of Mathematics
Sun Yat-sen University

December 24, 2020

Table of Contents

Table of Contents

Markov Graphs and Their Properties

Undirected graphical model (UGM), also called a Markov random field (MRF) or Markov network: the edges have no errows.

A graph G consists of a pair (V, E), where V is a set of vertices and E the set of edges (defined by pairs of vertices).

- Two vertices X and Y are called adjacent if there is a edge joining them; this is denoted by $X \sim Y$.
- A subgraph $U \in V$ is a subset of vertices together with their edges.
- A clique is a complete subgraph — a set of vertices that are all adjacent to one another; it is called maximal if it is a clique and no other vertices can be added to it and still yield a clique.

Markov Graphs and Their Properties

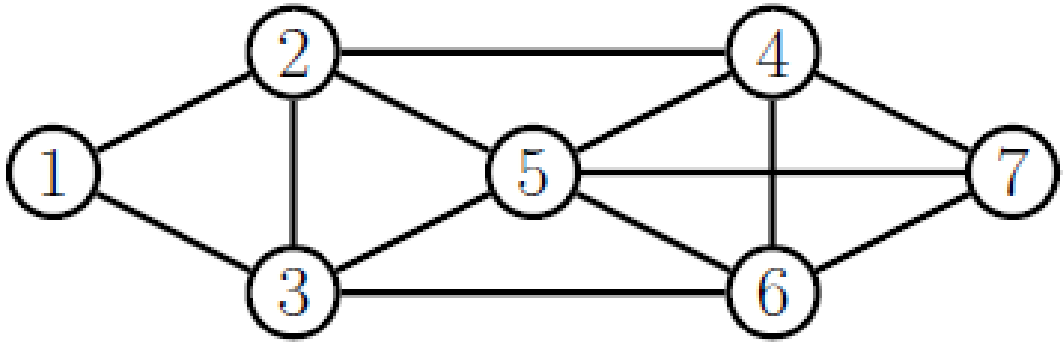


Figure: UGM

Markov Graphs and Their Properties

UGMs define conditional independent (CI) relationships via simple graph separation as follows:

for sets of nodes A, B, and C, we say $\mathrm{x}_A \perp_G \mathrm{x}_B \mid \mathrm{x}_C$ iff C separates A from B in the graph G.

- The global Markov property for UGMs: when we remove all the nodes in C, if there are no paths connecting any node in A to any node in B, then the CI property holds.

Markov Graphs and Their Properties

The smallest set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t 's Markov blanket ($\mathrm{mb}(t)$).

The Markov blanket satisfies the following property:

$$t \perp \mathcal{V} \setminus \mathrm{cl}(t) \mid \mathrm{mb}(t)$$

where $\mathrm{cl}(t) \triangleq \mathrm{mb}(t) \cup \{t\}$ is the closure of node t.

- The undirected local Markov property: a node's Markov blanket is its set of immediate neighbors.

Markov Graphs and Their Properties

From the local Markov property, we can easily see that two nodes are conditionally independent given the rest if there is no direct edge between them.

- The pairwise Markov property:

$$s \perp t \mid \mathcal{V} \backslash \{s, t\} \Longleftrightarrow G_{st} = 0$$

This is called the pairwise Markov property.

Markov Graphs and Their Properties

Theorem 19.3.1 (Hammersley-Clifford). A positive distribution $p(\mathrm{y}) > 0$ satisfies the CI properties of an undirected graph G iff p can be represented as a product of factors, one per maximal clique, i.e.

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c \mid \boldsymbol{\theta}_c)$$

where $\mathcal{C}$ is the set of all the (maximal) cliques of G, and $Z(\theta)$ is the partition function given by

$$Z(\theta) \triangleq \sum_{\mathbf{x}} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c \mid \theta_c)$$

Markov Graphs and Their Properties

pairwise Markov graphs: restrict the parameterization to the edges of the graph, rather than the maximal cliques:

$$p(\mathbf{y} \mid \boldsymbol{\theta}) \propto \psi_{12}(y_1, y_2)\,\psi_{13}(y_1, y_3)\,\psi_{23}(y_2, y_3)\,\psi_{24}(y_2, y_4)\,\psi_{34}(y_3, y_4)\,\psi_{35}(y_3, y$$
$$\propto \prod_{s \sim t} \psi_{st}(y_s, y_t)$$

Representing Potential Function

- If the variables are discrete, we can represent the potential or factors or energy functions as tables of (non-negative) numbers.
- maximum entropy or a log-linear model: defined as a linear function of the parameters.

$$\log \psi_c(\mathbf{y}_c) \triangleq \phi_c(\mathbf{y}_c)^T \theta_c$$

where $\phi_c(\mathbf{x}_c)$ is a feature vector derived from the values of the variables $\mathbf{y}_c$. The resulting log probability has the form

$$\log p(\mathbf{y} \mid \boldsymbol{\theta}) = \sum_c \phi_c(\mathbf{y}_c)^T \boldsymbol{\theta}_c - \log Z(\boldsymbol{\theta})$$

For example, consider a pairwise MRF, we set

$$\phi_{st}(y_s, y_t) = [\ldots, \mathbb{I}(y_s = j, y_t = k), \ldots]$$

Give each feature a weight, we have $K \times K$ potential function:

$$\psi_{st}(y_s = j, y_t = k) = \exp\left(\left[\boldsymbol{\theta}_{st}^T \boldsymbol{\phi}_{st}\right]_{jk}\right) = \exp(\theta_{st}(j, k))$$

.

Example to Motivate Undirected Graphs

- Four students study in pairs to work on homework.
  Alice and Bob are friends
  Bob and Charles study together
  Charles and Debbie argue with each other
  Debbie and Alice study together
  Alice and Charles can't stand each other
  Bob and Debbie had relationship ended badly
- Professor may have mis-spoken.
- Students may have figured out the problem.
- Students transmit this understanding to his/her study partner.

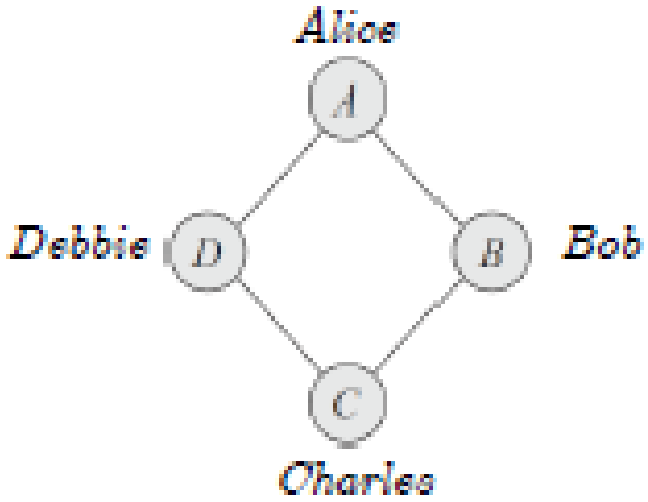
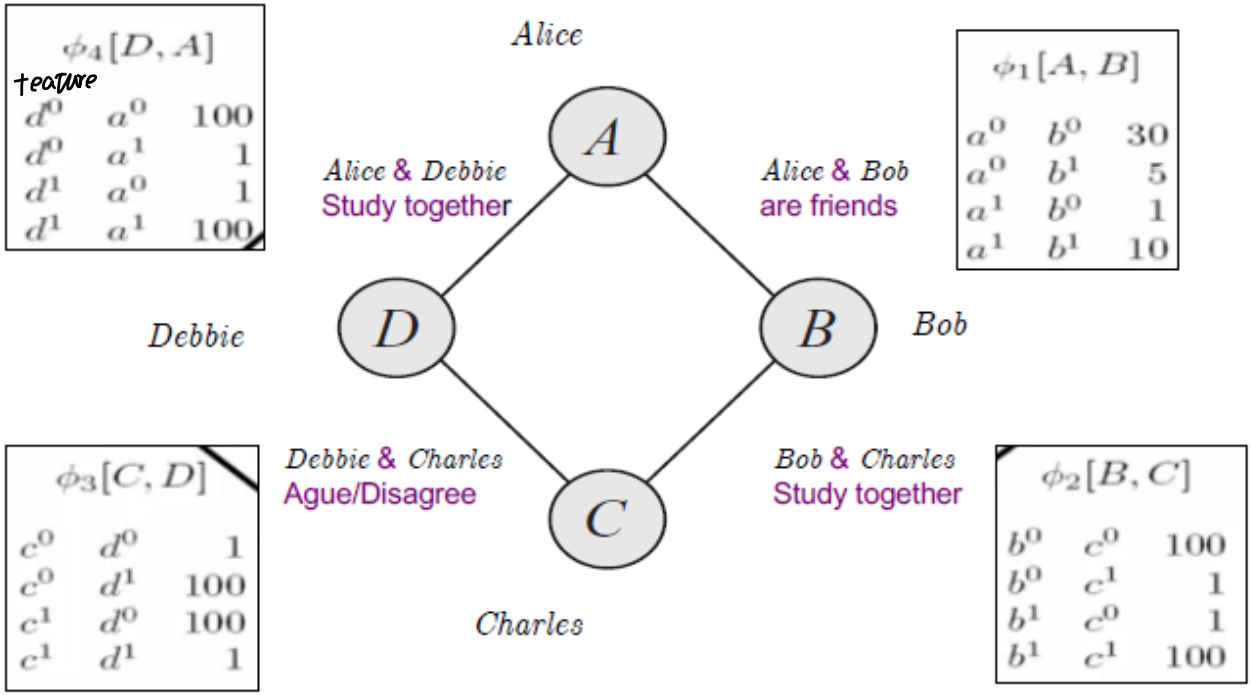


Figure: A Social Network with $(A \perp C | \{B, D\})$ and $(B \perp D | \{A, C\})$

Example to Motivate Undirected Graphs

Factors for Misconception Example

- Factors Captures affinities between variables.
- Let D be a set of random variables, i.e. ϕ.
- A factor ψ is a function from $Val(D)$ to R.



Example to Motivate Undirected Graphs

Assignment				Unnormalized	Normalized
a^0	b^0	c^0	d^0	300000	0.04
a^0	b^0	c^0	d^1	300000	0.04
a^0	b^0	c^1	d^0	300000	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	5000000	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	1000000	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	100000	0.014
a^1	b^1	c^1	d^0	100000	0.014
a^1	b^1	c^1	d^1	100000	0.014

$$P(a,b,c,d) = \frac{1}{Z}\phi_1(a,b) \cdot \phi_2(b,c) \cdot \phi_3(c,d) \cdot \phi_4(d,a)$$

$$Z = \sum_{a,b,c,d} \phi_1(a,b) \cdot \phi_2(b,c) \cdot \phi_3(c,d) \cdot \phi_4(d,a) = 7201840$$

Table of Contents

Gaussian MRF

Markov networks where all the variables are continues.

The Gaussian distribution is almost always used for such graphical models, because of its convenient analytical properties.

We assume that the observations have a multivariate Gaussian distribution with mean μ and covariance matrix Σ.

The Precision Matrix

The inverse covariance matrix Σ^{-1} is called the precision matrix Θ, which contains information about the partial covariances between the variables;

- $\rho_{X_i X_j, \mathbf{V} \setminus \{X_i, X_j\}} = -\frac{\Theta_{ij}}{\sqrt{\Theta_{ii}\Theta_{jj}}}$.
- If $\Theta_{ij} = 0$, then variables i and j are conditionally independent, given the other variables.

The Precision Matrix

proof: Let $X \in \mathbb{R}^p$ be distributed as $\mathcal{N}(\mu, \Sigma)$ and partition the random vector into two components $X_A \in \mathbb{R}^a$ and $X_B \in \mathbb{R}^b$ such that $a + b = p$. Let μ and Σ be partitioned accordingly, i.e.,

$$\mu = \begin{pmatrix} \mu_A \\ \mu_B \end{pmatrix} \quad \textit{and} \quad \Sigma = \begin{pmatrix} \Sigma_{A,A} & \Sigma_{A,B} \\ \Sigma_{B.A} & \Sigma_{B,B} \end{pmatrix}.$$

The conditional density $f(x_A|x_B)$ is proportional to:

$$\begin{aligned}
& f(x_A \mid x_B) \\
& \propto \exp\left\{-\tfrac{1}{2}\left(x_A-\mu_A\right)^T \Theta_{A,A}\left(x_A-\mu_A\right) - \left(x_A-\mu_A\right)^T \Theta_{A,B}\left(x_B-\mu_B\right)\right\} \\
& \propto \exp\left\{-\tfrac{1}{2}\left(x_A-\mu_A-\Theta_{A,A}^{-1}\Theta_{A,B}\left(x_B-\mu_B\right)\right)^T \Theta_{A,A}\right. \\
& \left.\times\left(x_A-\mu_A-\Theta_{A,A}^{-1}\Theta_{A,B}\left(x_B-\mu_B\right)\right)\right\}
\end{aligned}$$

where we used the same partitioning for Θ as for Σ.

The Precision Matrix

Let $A = \{i, j\}$, and $B = [p] \backslash \{i, j\}$, then $\Sigma_{\{i,j\}([p]\backslash\{i,j\})} = (\Theta_{\{i,j\},\{i,j\}})^{-1}$.

And $X_i \perp X_j | X_{[p]\backslash\{i,j\}}$ if and only if the 2×2 conditional covariance matrix $(\Theta_{\{i,j\},\{i,j\}})^{-1}$ is diagonal. This is the case if and only if $\Theta_{\{i,j\},\{i,j\}}$ is diagonal, or equivalently, $\Theta_{i,j} = 0$.

Multiple linear regression

Suppose we partition $X = (Z, Y)$ where $Z = (X_1, \ldots, X_{p-1})$ consists of the first $p-1$ variables and $Y = X_p$ is the last. Then we have the conditional distribution of Y give Z

$$Y \mid Z = z \sim N\left(\mu_Y + (z - \mu_Z)^T \Sigma_{ZZ}^{-1} \sigma_{ZY}, \sigma_{YY} - \sigma_{ZY}^T \Sigma_{ZZ}^{-1} \sigma_{ZY}\right)$$

where we have partitioned Σ as

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{ZZ} & \sigma_{ZY} \\ \sigma_{ZY}^T & \sigma_{YY} \end{pmatrix}$$

The conditional mean has exactly the same form as the population multiple linear regression of Y on Z, with regression coefficient $\beta = \Sigma_{ZZ}^{-1} \sigma_{ZY}$.

Multiple linear regression

If we partition Θ in the same way since $\Sigma\Theta = \mathrm{I}$ standard formulas for partitioned inverses give

$$\theta_{ZY} = -\theta_{YY} \cdot \Sigma_{ZZ}^{-1}\sigma_{ZY}$$

where $1/\theta_{YY} = \sigma_{YY} - \sigma_{ZY}^T\Sigma_{ZZ}^{-1}\sigma_{ZY} > 0$. Hence

$$\begin{aligned}\beta &= \Sigma_{ZZ}^{-1}\sigma_{ZY} \\ &= -\theta_{ZY}/\theta_{YY}\end{aligned}$$

- The dependence of Y on Z is in the mean term alone. Zero elements in β and hence θ_{ZY} mean that the corresponding elements of Z are conditionally independent of Y, given the rest.
- We can learn about this dependence structure through multiple linear regression.

A Modified Regression Algorithm

- Consider a standard LS regression of the outcome $\mathrm{y} = X_j$ on a predictor matrix $Z = X_{-j}$. The estimates for this fit can written as $\left(\mathbf{Z}^T Z\right)^{-1}\left(Z^T \mathrm{y}\right)$.
- Now $Z^T Z$ is the sample covariance of the predictors. The algorithm for exact maximization of $\ell(\Theta)$ replaces $\mathbf{Z}^T Z$ with $\hat{\Sigma}_{-j,-j}$, the model's current estimate of the covariance among the X_{-j}; a modified linear regression,
- We apply this modified linear regression to each node, leaving out variables that represent missing edges; and cycling through the nodes until convergence.

A Modified Regression Algorithm

Algorithm 17.1 *A Modified Regression Algorithm for Estimation of an Undirected Gaussian Graphical Model with Known Structure.*

1. Initialize $\mathbf{W} = \mathbf{S}$.
2. Repeat for $j = 1, 2, \ldots, p, 1, 2, \ldots, p, \ldots$ until convergence:
   (a) Partition the matrix $\mathbf{W}$ into part 1: all but the jth row and column, and part 2: the jth row and column.
   (b) Solve $\mathbf{W}_{11}^{*}\beta^{*} - s_{12}^{*} = 0$ for the unconstrained edge parameters β^{*}, using the reduced system of equations as in (17.19). Obtain $\hat{\beta}$ by padding $\hat{\beta}^{*}$ with zeros in the appropriate positions.
   (c) Update $w_{12} = \mathbf{W}_{11}\hat{\beta}$
3. In the final cycle (for each j) solve for $\hat{\theta}_{12} = -\hat{\beta} \cdot \hat{\theta}_{22}$, with $1/\hat{\theta}_{22} = s_{22} - w_{12}^{T}\hat{\beta}$. ($I = W_{12}\theta_{12} + W_{22}\theta_{22}$, $\theta_{22}' = W_{22} + W_{22}\frac{\theta_{12}}{\theta_{22}}$, $\frac{1}{\theta_{22}} = W_{22} - W_{12}\beta$)

S denotes the empirical covariance matrix, and **W** is the inverse of $\hat{\Theta}$. We remove the zero elements in β and reduced it to β^{*}.

Step 2.(b) is simply replace the inverse of the corresponding part in **S** with that in **W**. Hence we estimate $\hat{\beta}^{*} = \mathbf{W}_{11}^{*-1}s_{12}^{*}$ instead of $\mathbf{S}_{11}^{*-1}s_{12}^{*}$.

A Modified Regression Algorithm

Where does update $w_{12} = \mathbf{W}_{11}\hat{\beta}$ come from?

$$\begin{pmatrix} \mathbf{W}_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix} \begin{pmatrix} \Theta_{11} & \theta_{12} \\ \theta_{12}^T & \theta_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & 0 \\ 0^T & 1 \end{pmatrix}$$

This implies

$$\begin{aligned} w_{12} &= -\mathrm{W}_{11}\theta_{12}/\theta_{22} \\ &= \mathrm{W}_{11}\beta \end{aligned} \tag{2.1}$$

This as part of the formula for the inverse of a partitioned inverse of a matrix.

Naive procedure for constructing graphs

(Meinshausen & Buhlmann 2006).

- The coefficient of the jth predictor in a linear regression is proportional to the partial correlation between the response variable and the jth predictor.
- Idea: apply the lasso to the graph problem by treating each node in turn as the response variable.
- include an edge $i \leftrightarrow j$ in the graph if either the coefficient of jth predictor in regression for x_i is non-zero, or the ith predictor in regression for x_j is non-zero.

The naive procedure from above ignores the symmetric relationship $\theta_{ij} = \theta_{ji}$.

Graphical Lasso

- Consider maximizing the penalized log-likelihood

$$\log\det\Theta - \mathrm{tr}(S\Theta) - \lambda\|\Theta\|_1$$

- The gradient equation is

$$\Theta^{-1} - \mathrm{S} - \lambda \cdot \mathrm{Sign}(\Theta) = 0$$

  Here we use sub-gradient notation, with $\mathrm{Sign}\left(\theta_{jk}\right) = \mathrm{sign}\left(\theta_{jk}\right)$ if $\theta_{jk} \neq 0$, else $\mathrm{Sign}\left(\theta_{jk}\right) \in [-1, 1]$ if $\theta_{jk} = 0$.
- Refer to Equation 2.1:(upper right block : $W_{12} - S_{12} - \lambda \mathrm{sign}(\beta) = 0$, $W_{12} = W_{11}\beta$.)

$$\mathbf{W}_{11}\beta - s_{12} + \lambda \cdot \mathrm{Sign}(\beta) = 0$$

- This is the gradient for a lasso regression:

$$\min_{\beta}\left\{\frac{1}{2}||W_{11}^{1/2}\beta - b\,\|^2 + \rho\|\,\beta\|_1\right\}$$

  where $b = W_{11}^{-1/2}s_{12}$.

Graphical Lasso

Friedman et al. (2008b) use the pathwise coordinate descent method to solve the modified lasso problem. Here are the details of pathwise coordinate descent for the graphical lasso algorithm. Letting $\mathbf{V} = \mathbf{W}_{11}$, the update has the form

$$\hat{\beta}_j \leftarrow S\left(s_{12j} - \sum_{k \neq j} V_{kj}\hat{\beta}_k, \lambda\right) / V_{jj}$$

for $j = 1, 2, \ldots, p-1, 1, 2, \ldots, p-1, \ldots$, where S is the soft-threshold operator:

$$S(x, t) = \text{sign}(x)(|x| - t)_+$$

The procedure cycles through the predictors until convergence.

Graphical Lasso

Algorithm 17.2 *Graphical Lasso.*

1. Initialize $\mathbf{W} = \mathbf{S} + \lambda\mathbf{I}$. The diagonal of $\mathbf{W}$ remains unchanged in what follows.

2. Repeat for $j = 1, 2, \ldots p, 1, 2, \ldots p, \ldots$ until convergence:

   (a) Partition the matrix $\mathbf{W}$ into part 1: all but the jth row and column, and part 2: the jth row and column.

   (b) Solve the estimating equations $\mathbf{W}_{11}\beta - s_{12} + \lambda \cdot \text{Sign}(\beta) = 0$ using the cyclical coordinate-descent algorithm (17.26) for the modified lasso.

   (c) Update $w_{12} = \mathbf{W}_{11}\hat{\beta}$

3. In the final cycle (for each j) solve for $\hat{\theta}_{12} = -\hat{\beta} \cdot \hat{\theta}_{22}$, with $1/\hat{\theta}_{22} = w_{22} - w_{12}^T\hat{\beta}$.

Table of Contents

Ising Model

- Observations at nodes are binary valued: 0 or 1 .
- Standard model is the Ising model

$$p(X, \Theta) = \frac{1}{Z} \exp \left[\sum_{(j,k) \in E} \theta_{jk} X_j X_k - \sum_i \theta_i X_i \right]$$

  where E is the set of edges in the graph
- It implies a logistic form for each node conditional on the others:

$$\Pr\left(X_j = 1 \mid X_{-j} = x_{-j}\right) = \frac{1}{1 + \exp\left(-\theta_j - \sum_{(j,k) \in E} \theta_{jk} x_k\right)}$$

Training maxent models using gradient methods

Consider an MRF in log-linear form:

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_c \boldsymbol{\theta}_c^T \boldsymbol{\phi}_c(\mathbf{y})\right)$$

where c indexes the cliques. The scaled log-likelihood is given by

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_i \log p\left(\mathbf{y}_i \mid \boldsymbol{\theta}\right) = \frac{1}{N} \sum_i \left[\sum_c \boldsymbol{\theta}_c^T \boldsymbol{\phi}_c\left(\mathbf{y}_i\right) - \log Z(\boldsymbol{\theta})\right]$$

since MRFs are in the exponential family, we know that this function is convex in θ, so it has a unique global maximum which we can find using gradient-based optimizers.
the derivative for the weights of a particular clique, c, is

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N} \sum_i \left[\boldsymbol{\phi}_c\left(\mathbf{y}_i\right) - \frac{\partial}{\partial \boldsymbol{\theta}_c} \log Z(\boldsymbol{\theta})\right]$$

Training maxent models using gradient methods

the derivative of the log partition function wrt θ_c is the expectation of the c' th feature under the model, i.e.,

$$\frac{\partial \log Z(\theta)}{\partial \theta_c} = \mathbb{E}\left[\phi_c(\mathbf{y}) \mid \theta\right] = \sum_y \phi_c(\mathbf{y}) p(\mathbf{y} \mid \theta)$$

Hence the gradient of the log likelihood is

$$\begin{aligned} \frac{\partial \ell}{\partial \boldsymbol{\theta}_c} &= \left[\frac{1}{N}\sum_i \phi_c\left(\mathbf{y}_i\right)\right] - \mathbb{E}\left[\phi_c(\mathbf{y})\right] \\ &= \mathbb{E}_{p_{\text{emp}}}\left[\phi_c(\mathbf{y})\right] - \mathbb{E}_{p(\cdot|\theta)}\left[\phi_c(\mathbf{y})\right] \end{aligned}$$

At the optimum, the gradient will be zero, so the empirical distribution of the features will match the model's predictions:

$$\mathbb{E}_{p_{\text{emp}}}\left[\phi_c(\mathbf{y})\right] = \mathbb{E}_{p(\cdot|\theta)}\left[\phi_c(\mathbf{y})\right]$$

Pseudo likelihood

- To find the maximum likelihood estimates, we can use gradient search or Newton methods. However, The complexity of calculating $p(\mathbf{y}, \boldsymbol{\theta})$ will be $O(k^n)$ if there are n nodes in the graph and each node could choose from k different values.
- One alternative to MLE is to maximize the pseudo likelihood (Besag 1975), defined as follows:

$$\ell_{PL}(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \log p\left(y_{id} \mid \mathbf{y}_{i,-d}, \boldsymbol{\theta}\right)$$

  That is, we optimize the product of the full conditionals (composite likelihood, Lindsay 1988).
- The complexity of calculating the pseudo likelihood $p(\mathbf{y}, \boldsymbol{\theta})$ will be $O(kn)$.

Stochastic Maximum Likelihood

The gradient of the log-likelihood for a fully observed MRF is given by

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \left[\frac{1}{N}\sum_i \phi_c(\mathbf{y}_i)\right] - \mathbb{E}\left[\phi_c(\mathbf{y})\right]$$

We can approximate the model expectations using Monte Carlo sampling. We can combine this with stochastic gradient descent, which takes samples from the empirical distribution.

Stochastic Maximum Likelihood

Algorithm 19.1: Stochastic maximum likelihood for fitting an MRF

Initialize weights θ randomly;
$k = 0$, $\eta = 1$;
for *each epoch* **do**
  for *each minibatch of size* B **do**
    for *each sample* $s = 1 : S$ **do**
      Sample $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\theta_k)$;
    $\hat{E}(\phi(\mathbf{y})) = \frac{1}{S}\sum_{s=1}^{S} \phi(\mathbf{y}^{s,k})$;
    for *each training case* i *in minibatch* **do**
      $\mathbf{g}_{ik} = \phi(\mathbf{y}_i) - \hat{E}(\phi(\mathbf{y}))$;
    $\mathbf{g}_k = \frac{1}{B}\sum_{i \in B} \mathbf{g}_{ik}$;
    $\theta_{k+1} = \theta_k - \eta \mathbf{g}_k$;
    $k = k + 1$;
    Decrease step size η;

Table of Contents

Conditional random fields (CRFs)

Just a version of an MRF where all the clique potentials are conditioned on input features:

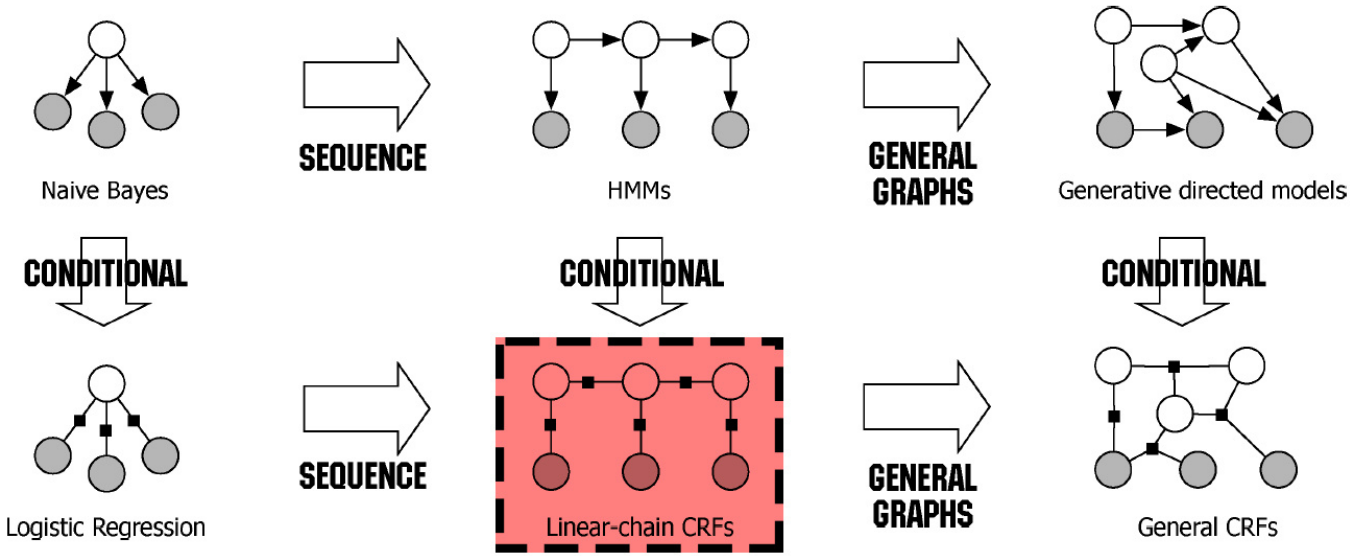
$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \prod_c \psi_c(\mathbf{y}_c \mid \mathbf{x}, \mathbf{w}) .$$

Also called **discriminative random field**, so we can focus our attention on modeling what we care about, namely the distribution of labels given the data.

Another important advantage of CRFs is that we can make the potentials (or factors) of the model be data-dependent.

The disadvantage of CRFs over MRFs is that they require labeled training data, and they are slower to train.

HMM, Chain-structured CRFs, MEMMs and the label-bias problem



The most widely used kind of CRF uses **a chain-structured graph** to model correlation amongst neighboring labels.

Such models are useful for a variety of **sequence labeling tasks**.

part-of-speech (POS) tagging

The input to the problem is a sentence. The output is a tagged sentence, where each word in the sentence is annotated with its part of speech.

INPUT:
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:
Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

KEY:

N = Noun
V = Verb
P = Preposition
Adv = Adverb
Adj = Adjective
...

part-of-speech (POS) tagging

Our task is to learn a function from inputs x to labels y, using our training examples $(x^{(i)}, y^{(i)})$ for $i = 1, \ldots, n$ as evidence.

One way to define the function $f(x)$ is through a *conditional model*, $p(y \mid x)$.

An alternative approach is to define a *generative model*, $p(x, y)$. In many cases we further decompose the probability $p(x, y)$ as follows:

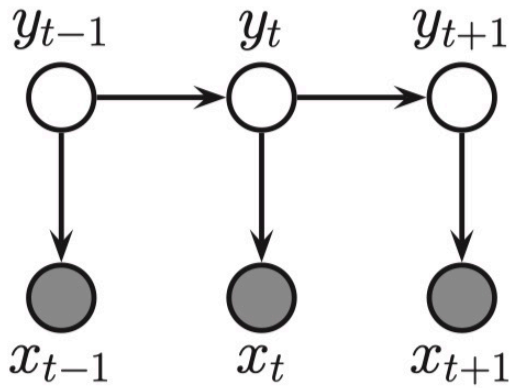
$$p(x, y) = p(y)p(x \mid y),$$

and then estimate the models for $p(y)$ and $p(x|y)$ separately.

We use Bayes rule directly in applying the joint model to a new test example.

$$f(x) = \arg\max_y p(y \mid x) = \arg\max_y \frac{p(y)p(x \mid y)}{p(x)} = \arg\max_y p(y)p(x \mid y)$$

HMM



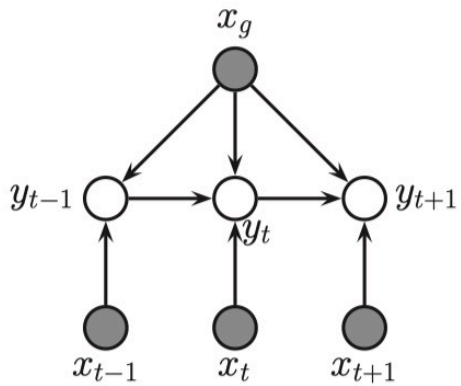
(a)

Two assumptions: I.Markov property. II.Emissions are conditionally independent.

$$p(\mathbf{x}, \mathbf{y} \mid \mathbf{w}) = \prod_{t=1}^{T} p\left(y_t \mid y_{t-1}, \mathbf{w}\right) p\left(\mathbf{x}_t \mid y_t, \mathbf{w}\right). \tag{4.2}$$

Each x_t is required to be local, since it is hard to define a generative model for the whole stream of observations, $x = x_{1:T}$.

Maximum Entropy Markov model(MEMM)



(b)

An obvious way to make a discriminative version of an HMM is to "reverse the arrows" from y_t to x_t:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \prod_t p\left(y_t \mid y_{t-1}, \mathbf{x}, \mathbf{w}\right), \tag{4.3}$$

where $\mathbf{x} = \left(\mathbf{x}_{1:T}, \mathbf{x}_g\right), \mathbf{x}_g$ are global features, and $\mathbf{x}_t$ are features specific to node t.

Observations are no longer conditionally independent. And a key advantage of MEMMs is that they allow highly flexible representations, allowing **features** to be easily integrated in the model.

Log-Linear Models

Definition 1 (Log-linear Models) A log-linear model consists of the following components:

- A set $\mathcal{X}$ of possible inputs.
- A set $\mathcal{Y}$ of possible labels. The set $\mathcal{Y}$ is assumed to be finite.
- A positive integer d specifying the number of features and parameters in the model.
- A function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ that maps any (x, y) pair to *a* feature-vector $f(x, y)$.
- A parameter vector $w \in \mathbb{R}^d$.
- For any $x \in \mathcal{X}, y \in \mathcal{Y}$, the model defines a condtional probability

$$p(y \mid x; w) = \frac{\exp(w \cdot f(x, y))}{\sum_{y' \in \mathcal{Y}} \exp\left(w \cdot f(x, y')\right)}$$

Here $w \cdot f(x, y) = \sum_{k=1}^{d} w_k f_k(x, y)$ is the inner product between w and $f(x, y)$.

Parameter Estimation in Log-Linear Models

The log-likelihood considers the sum of log probabilities of examples in the training data:

$$L(w) = \sum_{i=1}^{n} \log p\left(y^{(i)} \mid x^{(i)}; w\right). \tag{4.4}$$

The first estimation method we will consider is maximum-likelihood estimation, where we choose our parameters as

$$w_{ML} = \arg\max_{v \in \mathbb{R}^d} L(w). \tag{4.5}$$

But the maximum-likelihood estimates can run into problems, in particular in cases where the number of features in the model is very large.

Parameter Estimation in Log-Linear Models

For example, assume that we have some trigram (u, v, l) which is seen only once in the training data, and to be concrete, assume that $f_{N(\text{any,statistical,model})}\left(x^{(100)}, y^{(100)}\right) = 1$,.

In addition, assume that this is the only trigram (u, v, l) in training data with u = *any*, and v = *statistical*.

In this case, it can be shown that the maximum-likelihood parameter estimate for w_{100} is $+\infty$, which gives

$$p\left(y^{(100)} \mid x^{(100)}; w\right) = 1 \tag{4.6}$$

Parameter Estimation in Log-Linear Models

Under the above assumptions, the feature $f_{N(any,statistical,model)}(x, y)$ is equal to 1 on only a single pair $x^{(i)}, y$. Because of this, as $w_{100} \to \infty$ we will have $p\left(y^{(100)} \mid x^{(100)}; w\right)$ tending closer and closer to a value of 1, **with all other values** $p\left(y^{(i)} \mid x^{(i)}; w\right)$ **remaining unchanged**.

A very common solution for log-linear models is to modify the objective function to include a regularization term, which prevents parameter values from becoming too large:

$$L(w) = \sum_{i=1}^{n} \log p\left(y^{(i)} \mid x^{(i)}; w\right) - \frac{\lambda}{2} \sum_{k} w_k^2. \tag{4.7}$$

MEMM

Definition 1 (a conditional tagging mode)

- A set of words $\mathcal{V}$ (this set may be finite, countably infinite, or even uncount- ably infinite)
- A finite set of tags $\mathcal{K}$.
- A function $p(y_1 \dots y_n \mid x_1 \dots x_n)$ such that:
  1. For any $\langle x_1 \dots x_n, y_1 \dots y_n \rangle \in \mathcal{S}$

$$p(y_1 \dots y_n \mid x_1 \dots x_n) \geq 0,$$

  where $\mathcal{S}$ is the set of all sequenceltag-sequence pairs $\langle x_1 \dots x_n, y_1 \dots y_n \rangle$ such that $n \geq 1$, $x_i \in \mathcal{V}$ for $i = 1 \dots n$, and $y_i \in \mathcal{K}$ for $i = 1 \dots n$.

  2. For any $x_1 \dots x_n$ such that $n \geq 1$ and $x_i \in \mathcal{V}$ for $i = 1 \dots n$

$$\sum_{y_1 \dots y_n \in \mathcal{Y}(n)} p(y_1 \dots y_n \mid x_1 \dots x_n) = 1,$$

  where $\mathcal{Y}(n)$ is the set of all tag sequences $y_1 \dots y_n$ such that $y_i \in \mathcal{K}$ for $i = 1 \dots n$.

MEMM

Given a conditional tagging model, the function from sentences $x_1 \dots x_n$ to tag sequences $y_1 \dots y_n$ is defined as

$$f(x_1 \dots x_n) = \arg \max_{y_1 \dots y_n \in \mathcal{Y}(n)} p(y_1 \dots y_n \mid x_1 \dots x_n)$$

Thus for any input $x_1 \dots x_n$, we take the highest probability tag sequence as the output from the model.
We are left with the following three questions:

- How we define a conditional tagging model $p(y_1 \dots y_n \mid x_1 \dots x_n)$?
- How do we estimate the parameters of the model from training examples?
- How do we efficiently find

$$\arg \max_{y_1 \dots y_n \in \mathcal{Y}(n)} p(y_1 \dots y_n \mid x_1 \dots x_n)$$

for any input $x_1 \dots x_n$?

Trigram MEMMs

We focus on trigram MEMMs, which make a second order Markov assumption, where each tag depends on the previous two tags. Our task is to model the conditional distribution

$$
\begin{aligned}
& P(Y_1 = y_1 \dots Y_n = y_n \mid X_1 = x_1 \dots X_n = x_n) \\
=& \prod_{i=1}^{n} P(Y_i = y_i \mid X_1 = x_1 \dots X_n = x_n, Y_1 = y_1 \dots Y_{i-1} = y_{i-1}) \\
=& \prod_{i=1}^{n} P(Y_i = y_i \mid X_1 = x_1 \dots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1}) .
\end{aligned} \tag{4.8}
$$

The final step is to use a log-linear model to estimate the probability

$$
P(Y_i = y_i \mid X_1 = x_1 \dots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1}) . \tag{4.9}
$$

Trigram MEMMs

For any pair of sequences $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, we define the ith "history" h_i to be the four-tuple.

We assume that we have a feature-vector representation $f(h_i, y) \in R^d$ for any history h_i paired with any tag $y \in K$. The feature vector could potentially take into account any information in the history h_i and the tag y.

Finally, we assume a parameter vector $w \in \mathbb{R}^d$, and that

$$\begin{aligned} & P(Y_i = y_i \mid X_1 = x_1 \ldots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1}) \\ = & \frac{\exp(w \cdot f(h_i, y_i))}{\sum_{y \in \mathcal{K}} \exp(w \cdot f(h_i, y))} \end{aligned} \tag{4.10}$$

Some Features

Word/tag features One example word/tag feature is the following:

$$f_{100}(h, y) = \begin{cases} 1, & \text{if } x_i \text{ is base and } y = \text{VB}, \\ 0, & \text{otherwise.} \end{cases}$$

Prefix and Suffix features An example of a suffix feature is as follows:

$$f_{101}(h, y) = \begin{cases} 1, & \text{if } x_i \text{ ends in ing and } y = \text{VBG}, \\ 0, & \text{otherwise.} \end{cases}$$

Trigram, Bigram and Unigram Tag features An example of a trigram tag feature is as follows:

$$f_{103}(h, y) = \begin{cases} 1, & \text{if } \langle y_{-2}, y_{-1}, y \rangle = \langle \text{DT}, \text{JJ}, \text{VB} \rangle, \\ 0, & \text{otherwise.} \end{cases}$$

Decoding with MEMMs: Another Application of the Viterbi Algorithm

First, note the similarity to the decoding problem for a trigram HMM tagger, which is to find

$$\arg \max_{y_1 \ldots y_n \in \mathcal{Y}(n)} q(\text{STOP} \mid y_{n-1}, y_n) \times \prod_{i=1}^{n} q(y_i \mid y_{i-2}, y_{i-1})\, e(x_i \mid y_i)\,.$$

Putting aside the $q(\text{STOP} \mid y_{n-1}, y_n)$ term, we have essentially replaced

$$\prod_{i=1}^{n} q(y_i \mid y_{i-2}, y_{i-1}) \times e(x_i \mid y_i)$$

by

$$\prod_{i=1}^{n} p(y_i \mid h_i; w)\,.$$

Decoding with MEMMs: Another Application of the Viterbi Algorithm

We again use dynamic programming, the recursive case is

$$\pi(k, u, v) = \max_{w \in \mathcal{K}_{k-2}} (\pi(k-1, w, u) \times p(v \mid h; w) \tag{4.11}$$

where $h = \langle w, u, x_1 \ldots x_n, k \rangle$. Recall that the recursive case for a trigram HMM tagger is

$$\pi(k, u, v) = \max_{w \in \mathcal{K}_{k-2}} (\pi(k-1, w, u) \times q(v \mid w, u) \times e(x_k \mid v)) . \tag{4.12}$$

Hence we have simply replaced $q(v \mid w, u) \times e(x_k \mid v)$ by $p(v \mid h; w)$.

MEMM-label bias problem

The disadvantage of MEMM is that it suffers from a subtle problem known as the **label bias problem** (Lafferty et al. 2001) that local features at time t do not influence states prior to time t.

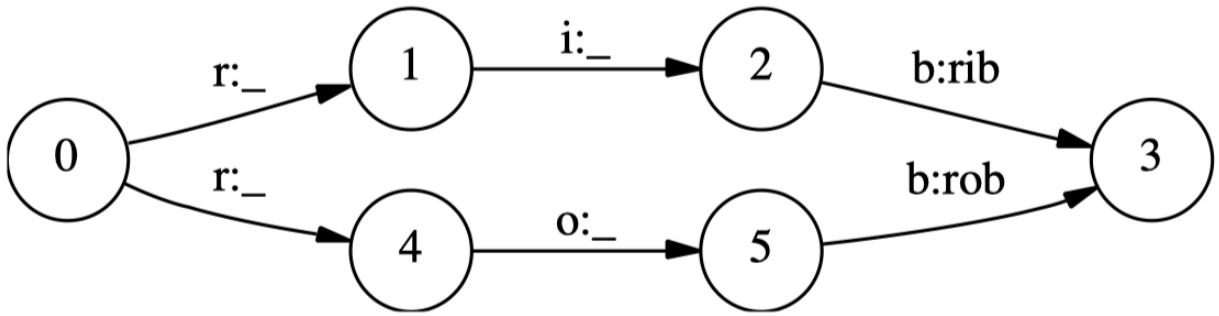


Figure 1. Label bias example, after (Bottou, 1991). For conciseness, we place observation-label pairs $o : l$ on transitions rather than states; the symbol '_' represents the null output label.

States with a single outgoing transition effectively ignore their observations. More generally, states with low-entropy next state distributions will take little notice of observations.

MEMM-label bias problem

Proper solutions require models that account for **whole state sequences at once** by letting some transitions "vote" more strongly than others depending on the corresponding observations.

This implies that score mass will not be conserved, but instead individual transitions can **"amplify" or "dampen"** the mass they receive.

CRFs are model class that does this in a purely probabilistic setting, with guaranteed global maximum likelihood convergence.

MEMM-label bias problem

This freedom from the label bias problem comes at a cost. Computing the normalization term exactly is more expensive with a CRF than with an MEMM. With an MEMM we normalize locally. Each local normalization costs $\mathcal{O}(c)$ where c is the number of classes, and we have to compute T of them, so the total cost is $\mathcal{O}(cT)$. With a linear chain CRF, on the other-hand, the total cost using an efficient dynamic programming algorithm called the forward-backward algorithm, is $\mathcal{O}\left(c^2 T\right)$.

Conditional random fields

For convenience, we'll use $\underline{x}$ to refer to an input sequence $x_1 \ldots x_m$, and $\underline{s}$ to refer to a sequence of states $s_1 \ldots s_m$. The set of all possible states is again $\mathcal{S}$; the set of all possible state sequences is $\mathcal{S}^m$. In conditional random fields we'll again build a model of

$$p\left(s_1 \ldots s_m \mid x_1 \ldots x_m\right) = p(\underline{s} \mid \underline{x}) \tag{4.13}$$

A first key idea in CRFs will be to define a feature vector

$$\underline{\Phi}(\underline{x}, \underline{s}) \in \mathbb{R}^d \tag{4.14}$$

that maps an **entire input sequence** $\underline{x}$ paired with an entire state sequence $\underline{s}$ to some d-dimensional feature vector.

Conditional random fields

We will often refer to $\underline{\Phi}$ as being a "global" feature vector (it is global in the sense that it takes the entire state sequence into account).

We then build a **giant** log-linear model,

$$p(\underline{s} \mid \underline{x}; \underline{w}) = \frac{\exp(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s}))}{\sum_{\underline{s}' \in \mathcal{S}^m} \exp\left(\underline{w} \cdot \underline{\Phi}\left(\underline{x}, \underline{s}'\right)\right)} \tag{4.15}$$

It is is "giant" in the sense that: 1) the space of possible values for s, i.e., $\mathcal{S}^m$, is huge. 2) The normalization constant (denominator in the above expression) involves a sum over the set $\mathcal{S}^m$.

Define the features

The next question is how to define $\underline{\Phi}(\underline{x}, \underline{s})$? Our answer will be

$$\underline{\Phi}(\underline{x}, \underline{s}) = \sum_{j=1}^{m} \underline{\phi}\left(\underline{x}, j, s_{j-1}, s_j\right) \tag{4.16}$$

where $\phi\left(\underline{x}, j, s_{j-1}, s_j\right)$ are the same as the feature vectors used in MEMMs. Or put another way, we're assuming that for $k = 1 \ldots d$, the k 'th global feature is

$$\Phi_k(\underline{x}, \underline{s}) = \sum_{j=1}^{m} \phi_k\left(\underline{x}, j, s_{j-1}, s_j\right) \tag{4.17}$$

Thus Φ_k is calculated by summing the "local" feature vector ϕ_k over the m different state transitions in $s_1 \ldots s_m$

Decording with CRFs

$$
\begin{aligned}
\arg\max_{\underline{s}\in\mathcal{S}^m} p(\underline{s} \mid \underline{x}; \underline{w}) &= \arg\max_{\underline{s}\in\mathcal{S}^m} \frac{\exp(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}))}{\sum_{\underline{s}'\in\mathcal{S}^m}\exp\left(\underline{w}\cdot\underline{\Phi}\left(\underline{x},\underline{s}'\right)\right)} \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \exp(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})) \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}) \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \quad \underline{w}\cdot\sum_{j=1}^{m}\underline{\phi}\left(\underline{x},j,s_{j-1},s_j\right) \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \sum_{j=1}^{m}\underline{w}\cdot\underline{\phi}\left(\underline{x},j,s_{j-1},s_j\right)
\end{aligned}
\tag{4.18}
$$

Decording with CRFs

So we have shown that finding the most likely sequence under the model is equivalent to finding the sequence that maximizes

$$\arg\max_{\underline{s}\in\mathcal{S}^m} \sum_{j=1}^{m} \underline{w}\cdot\underline{\phi}\left(\underline{x}, j, s_{j-1}, s_j\right) \tag{4.19}$$

This problem has a clear intuition. Each transition from state s_{j-1} to state s_j has an associated score

$$\underline{w}\cdot\underline{\phi}\left(\underline{x}, j, s_{j-1}, s_j\right) \tag{4.20}$$

Decording with CRFs

We can again solve this problem using a variant of the Viterbi algorithm, in a very similar way to the decoding algorithm for HMMs or MEMMs:

- Initialization: for $s \in \mathcal{S}$

$$\pi[1, s] = \underline{w} \cdot \phi\left(\underline{x}, 1, s_0, s\right) \tag{4.21}$$

  where s_0 is a special "initial" state.

- For $j = 2 \ldots m, s = 1 \ldots k$:

$$\pi[j, s] = \max_{s' \in \mathcal{S}} \left[\pi\left[j-1, s'\right] + \underline{w} \cdot \underline{\phi}\left(\underline{x}, j, s', s\right)\right] \tag{4.22}$$

We then have

$$\max_{s_1 \ldots s_m} \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}\left(\underline{x}, j, s_{j-1}, s_j\right) = \max_{s} \pi[m, s] \tag{4.23}$$

Parameter Estimation in CRFs

We then proceed in exactly the same way as for regular log-linear models. The *regularized log-likelihood function* is

$$L(\underline{w}) = \sum_{i=1}^{n} \log p\left(\underline{s}^i \mid \underline{x}^i; \underline{w}\right) - \frac{\lambda}{2}\|\underline{w}\|^2 \tag{4.24}$$

Our parameter estimates are then

$$\underline{w}^* = \arg\max_{\underline{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \log p\left(\underline{s}^i \mid \underline{x}^i; \underline{w}\right) - \frac{\lambda}{2}\|\underline{w}\|^2 \tag{4.25}$$

We'll again use gradient-based optimization methods to find $\underline{w}^*$. As before, the partial derivatives are

$$\frac{\partial}{\partial w_k} L(\underline{w}) = \sum_i \Phi_k\left(\underline{x}^i, \underline{s}^i\right) - \sum_i \sum_{s \in \mathcal{S}^m} p\left(\underline{s} \mid \underline{x}^i; \underline{w}\right) \Phi_k\left(\underline{x}^i, \underline{s}\right) - \lambda w_k \tag{4.26}$$

Parameter Estimation in CRFs

The first term is easily computed, because

$$\sum_i \Phi_k\left(\underline{x}^i, \underline{s}^i\right) = \sum_i \sum_{j=1}^{m} \phi_k\left(\underline{x}^i, j, s_{j-1}^i, s_j^i\right) \tag{4.27}$$

The second term is more difficult to deal with, because it involves a sum over $\mathcal{S}^m$, a very large set. However, we will see that this term can be computed efficiently using **dynamic programming**.

Parameter Estimation in CRFs

The derivation is as follows:

$$
\begin{aligned}
\sum_{s \in \mathcal{S}^m} p\left(\underline{s} \mid \underline{x}^i; \underline{w}\right) \Phi_k\left(\underline{x}^i, \underline{s}\right) &= \sum_{\underline{s} \in \mathcal{S}^m} p\left(\underline{s} \mid \underline{x}^i; \underline{w}\right) \sum_{j=1}^{m} \phi_k\left(\underline{x}^i, j, s_{j-1}, s_j\right) \\
&= \sum_{j=1}^{m} \sum_{\underline{s} \in \mathcal{S}^m} p\left(\underline{s} \mid \underline{x}^i; \underline{w}\right) \phi_k\left(\underline{x}^i, j, s_{j-1}, s_j\right) \\
&= \sum_{j=1}^{m} \sum_{a \in \mathcal{S}, b \in \mathcal{S}} \sum_{s \in \mathcal{S}^m, s_j = b} p\left(\underline{s} \mid \underline{x}^i; \underline{w}\right) \phi_k\left(\underline{x}^i, j, s_{j-1}, s_j\right) \qquad (4.28) \\
&= \sum_{j=1}^{m} \sum_{a \in \mathcal{S}, b \in \mathcal{S}} \phi_k\left(\underline{x}^i, j, a, b\right) \sum_{\substack{s \in \mathcal{S}^m: \\ s_{j-1} = a, s_j = b}} p\left(\underline{s} \mid \underline{x}^i; \underline{w}\right) \\
&= \sum_{j=1}^{m} \sum_{a \in \mathcal{S}, b \in \mathcal{S}} q_j^i(a, b) \phi_k\left(\underline{x}^i, j, a, b\right)
\end{aligned}
$$

where

$$
q_j^i(a, b) = \sum_{\underline{s} \in \mathcal{S}^m : s_{j-1} = a, s_j = b} p\left(\underline{s} \mid \underline{x}^i; \underline{w}\right) \qquad (4.29)
$$

Parameter Estimation in CRFs

The quantity $q_j^i(a, b)$ has a fairly intuitive interpretation: it is the probabilty of the i 'th training example $\underline{x}^i$ having state a at position $j-1$ and state b at position j, under the distribution $p(\underline{s} \mid \underline{x}; \underline{w})$

A critical result is that for a given i, all $q_j^i(a, b)$ terms can be calculated together, in $O\left(mk^2\right)$ time. The algorithm that achieves this is the forward-backward algorithm. This is another dynamic programming algorithm, and is closely related to the Viterbi algorithm.