Typical ML goals

- Two inference problems
  - Given a set of observations find parameters that best explain the observations
  - Find probability of a new observation, predict missing part of observation
- What do we mean by i.i.d (Independent and identically distributed)?
- Why Maximum Likelihood?

Coin toss example

The data set $\mathcal{X} \triangleq \{x_i\}_{i=1}^{|\mathcal{X}|}$ can be considered a sequence of independent and identically distributed (i.i.d.) realisations of a random variable (r.v.) X. The parameters ϑ are dependent on the distributions considered, e.g., for a Gaussian, $\vartheta = \{\mu, \sigma^2\}$.

$$p(\vartheta|\mathcal{X}) = \frac{p(\mathcal{X}|\vartheta) \cdot p(\vartheta)}{p(\mathcal{X})}, \qquad \text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}.$$

- Assume deformed coin with probability p of throwing a heads: HHTHTTHTHHTH

  Intuitively, what could p be?

$$p(C{=}c|p) = p^c\,(1-p)^{1-c} \triangleq \mathrm{Bern}(c|p) \tag{8}$$

where we define $c{=}1$ for heads and $c{=}0$ for tails[4].

Building an ML estimator for the parameter p can be done by expressing the (log) likelihood as a function of the data:

$$\begin{aligned}\mathcal{L} &= \log \prod_{i=1}^{N} p(C{=}c_i|p) = \sum_{i=1}^{N} \log p(C{=}c_i|p) \qquad (9)\\ &= n^{(1)} \log p(C{=}1|p) + n^{(0)} \log p(C{=}0|p)\\ &= n^{(1)} \log p + n^{(0)} \log(1-p) \qquad (10)\end{aligned}$$

where $n^{(c)}$ is the number of times a Bernoulli experiment yielded event c. Differentiating with respect to (w.r.t.) the parameter p yields:

$$\frac{\partial \mathcal{L}}{\partial p} = \frac{n^{(1)}}{p} - \frac{n^{(0)}}{1-p} \overset{!}{=} 0 \quad \Leftrightarrow \quad \hat{p}_{\mathrm{ML}} = \frac{n^{(1)}}{n^{(1)} + n^{(0)}} = \frac{n^{(1)}}{N}, \tag{11}$$

Classification Models

- Training Data { (**x**, y)_i }...

  Test Data { (**x**, ?)_j}
- Discriminant functions:

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0\right)$$

class $\mathcal{C}_1$ if $y(\mathbf{x}) \geqslant 0$ and to class $\mathcal{C}_2$ otherwise

- Find parameters that optimize (minimize/maximize) some function on the training data.

  Example loss functions: L1, L2, hinge loss, probabilistic...

Generative Models

- Generative models (eg. Naïve Bayes, HMMs, topic models): Come up with a step-by-step process for generating the data: assumptions on dependence, underlying distributions
- Quantities of interest : $p(\mathbf{x}, \mathcal{C}_k)$, $p(\mathcal{C}_k)$, $p(\mathbf{x}|\mathcal{C}_k)$

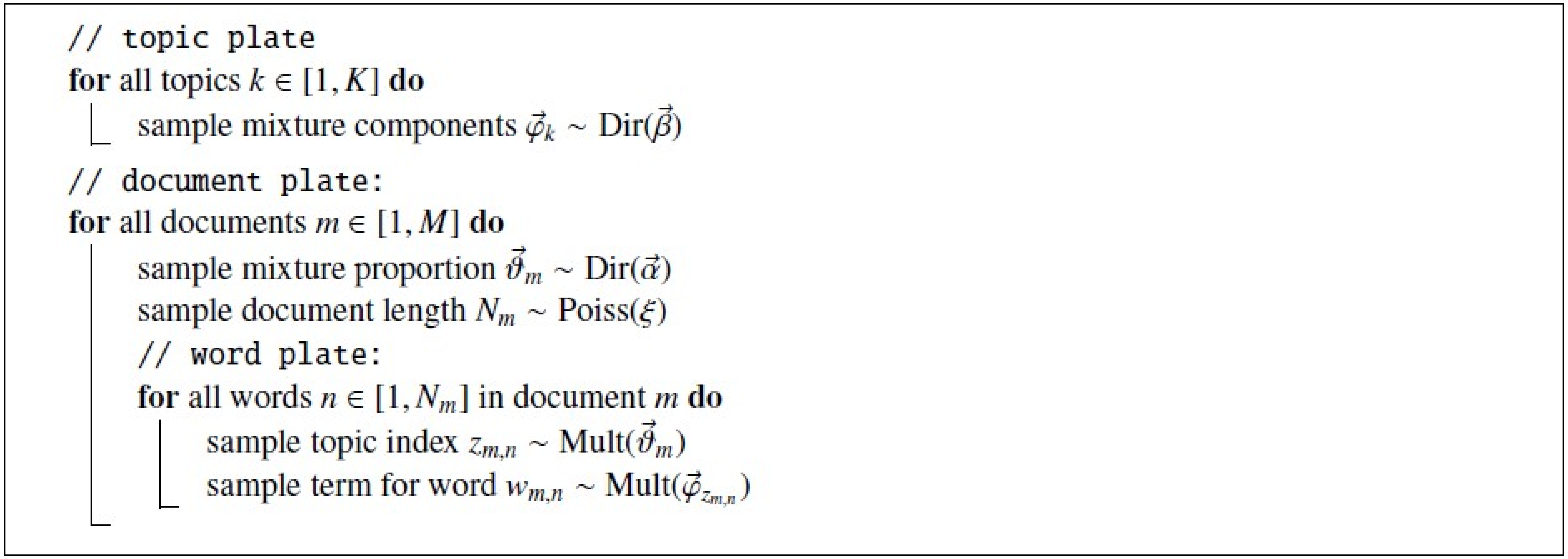
$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

Multinomial Naïve Bayes for text classification

- How to generate a document?
  - First pick a class, with probability p(y)
  - Sample each word in the document using Mult(y) (class-specific multinomial distribution)
  - Given independence assumptions, MLE estimates yield probabilities proportional to frequencies in the training data!

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k | y).$$

Latent Dirichlet Allocation



```
// topic plate
for all topics $k \in [1, K]$ do
    sample mixture components $\vec{\varphi}_k \sim \mathrm{Dir}(\vec{\beta})$
// document plate:
for all documents $m \in [1, M]$ do
    sample mixture proportion $\vec{\vartheta}_m \sim \mathrm{Dir}(\vec{\alpha})$
    sample document length $N_m \sim \mathrm{Poiss}(\xi)$
    // word plate:
    for all words $n \in [1, N_m]$ in document $m$ do
        sample topic index $z_{m,n} \sim \mathrm{Mult}(\vec{\vartheta}_m)$
        sample term for word $w_{m,n} \sim \mathrm{Mult}(\vec{\varphi}_{z_{m,n}})$
```

Fig. 7. Generative model for latent Dirichlet allocation.

LDA equations

$$p(\vec{w}_m, \vec{z}_m, \vec{\vartheta}_m, \underline{\Phi} | \vec{\alpha}, \vec{\beta}) = \overbrace{\underbrace{\prod_{n=1}^{N_m} p(w_{m,n} | \vec{\varphi}_{z_{m,n}}) p(z_{m,n} | \vec{\vartheta}_m)}_{\text{word plate}} \cdot p(\vec{\vartheta}_m | \vec{\alpha})}^{\text{document plate (1 document)}} \cdot \underbrace{p(\underline{\Phi} | \vec{\beta})}_{\text{topic plate}}$$

$$p(\vec{z} | \vec{w}) = \frac{p(\vec{z}, \vec{w})}{p(\vec{w})} = \frac{\prod_{i=1}^{W} p(z_i, w_i)}{\prod_{i=1}^{W} \sum_{k=1}^{K} p(z_i{=}k, w_i)}$$

$$p(\vec{w} | \vec{z}, \underline{\Phi}) = \prod_{i=1}^{W} p(w_i | z_i) = \prod_{i=1}^{W} \varphi_{z_i, w_i}.$$

Discriminative Models

- Model the posterior probabilities (p(C|**x**)) directly!
  - Example: logistic regression,

    log(odds-ratio) = $\ln\left[p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})\right]$ is a linear

    function on variables, i.e. $\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0\right)$$

Finding parameters

log-loss over the data set L:

$$\ell(\mathbf{u}; L) = \sum_{(\mathbf{x},y)\in L} \log\left(1 + e^{-\mathbf{u}^T\mathbf{x}y}\right)$$

- Or maximize (conditional) likelihood on training data

For a data set $\{\phi_n, t_n\}$, where $t_n \in \{0,1\}$ and $\phi_n = \phi(\mathbf{x}_n)$, with $n = 1,\ldots,N$, the likelihood function can be written

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n} \tag{4.89}$$

where $\mathbf{t} = (t_1,\ldots,t_N)^{\mathrm{T}}$ and $y_n = p(\mathcal{C}_1|\phi_n)$. As usual, we can define an error function by taking the negative logarithm of the likelihood, which gives the *cross-entropy* error function in the form

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \tag{4.90}$$

Generative/Discriminative pairs

logistic regression classifier can be converted into a naive Bayes classifier with the same decision boundary, and vice versa. Another way of saying this is that the naive Bayes model (1.5) defines the same family of distributions as the logistic regression model (1.7), if we interpret it generatively as

$$p(y, \mathbf{x}) = \frac{\exp\left\{\sum_k \lambda_k f_k(y, \mathbf{x})\right\}}{\sum_{\tilde{y}, \tilde{\mathbf{x}}} \exp\left\{\sum_k \lambda_k f_k(\tilde{y}, \tilde{\mathbf{x}})\right\}}. \tag{1.9}$$

This means that if the naive Bayes model (1.5) is trained to maximize the conditional likelihood, we recover the same classifier as from logistic regression. Conversely, if the logistic regression model is interpreted generatively, as in (1.9), and is trained to maximize the joint likelihood $p(y, \mathbf{x})$, then we recover the same classifier as from naive Bayes. In the terminology of Ng and Jordan [2002], naive Bayes and logistic regression form a *generative-discriminative pair*.

Simple example for Markov Models

- label space ={ rainy, windy, sunny }

  Data from several periods in time
- For e.g. RWWSRRSWW, SSRRRRW...
- Your goal: predict the probability of seeing a particular label (pattern)

  Intuitively: what could effect this pattern? probability of seeing a particular weather independent of other things (e.g. depending on weather patterns in a city), “dependent“ probability (probability of raining today given that it has rained yesterday, all of last week...)
- visualize as a probabilistic state machine

First-order Markov Chain

To express such effects in a probabilistic model, we need to relax the i.i.d. assumption, and one of the simplest ways to do this is to consider a *Markov model*. First of all we note that, without loss of generality, we can use the product rule to express the joint distribution for a sequence of observations in the form

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}). \tag{13.1}$$

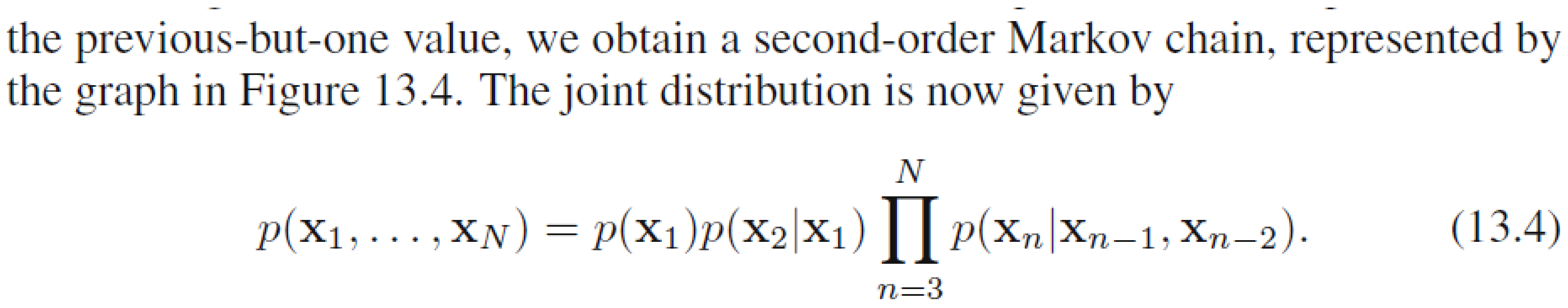
If we now assume that each of the conditional distributions on the right-hand side is independent of all previous observations except the most recent, we obtain the *first-order Markov chain*, which is depicted as a graphical model in Figure 13.3. The

joint distribution for a sequence of N observations under this model is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n | \mathbf{x}_{n-1}). \tag{13.2}$$

First-order Markov Chains

- Parameters of interest: start probability of a state, p(x_i) and transition probability between states, p(x_j|x_i)
- As expected, MLE estimates boil down to counting the frequencies in the training data...



the previous-but-one value, we obtain a second-order Markov chain, represented by the graph in Figure 13.4. The joint distribution is now given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)\prod_{n=3}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}). \qquad (13.4)$$

Hidden Markov Models

- Intuition, states vs. observables

  Example (slightly changed)
- State space { Rainy, Windy, Sunny }
- Observable (features, arbitrary), sample boolean for our example
  - Grass is wet
  - Dayu wears a hat
  - Sky is clear

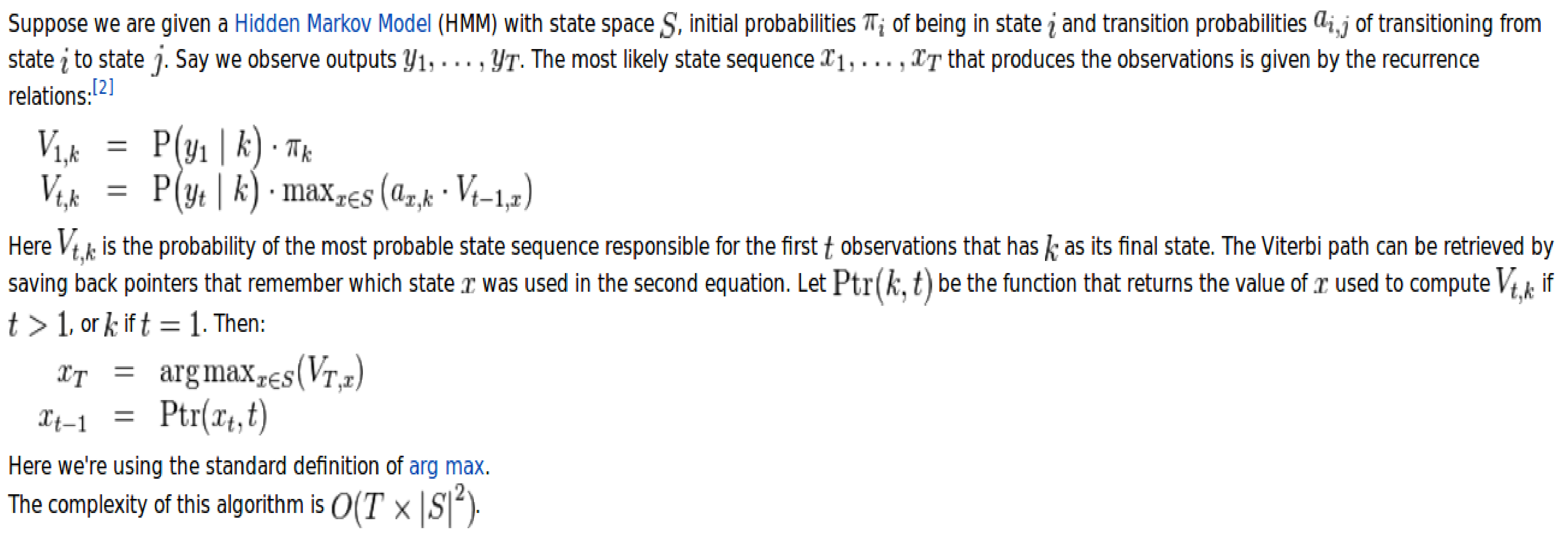
First-order HMMs

- Each observation has a probability depending on the state
  - p(D wears a hat|rainy) = 0.45, p(D wears a hat|sunny) = 0.45, p(D wears a hat|windy) = 0.1
  - p(grass is wet|rainy) = 0.7, p(grass is wet|windy) = 0.3

The joint distribution for this model is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n | \mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{z}_n).$$

Viterbi paths for predicting state patterns



Suppose we are given a Hidden Markov Model (HMM) with state space S, initial probabilities π_i of being in state i and transition probabilities $a_{i,j}$ of transitioning from state i to state j. Say we observe outputs $y_1, \dots, y_T$. The most likely state sequence $x_1, \dots, x_T$ that produces the observations is given by the recurrence relations:[2]

$$\begin{array}{rcl} V_{1,k} & = & \mathrm{P}\big(y_1 \mid k\big) \cdot \pi_k \\ V_{t,k} & = & \mathrm{P}\big(y_t \mid k\big) \cdot \max_{x \in S}\left(a_{x,k} \cdot V_{t-1,x}\right) \end{array}$$

Here $V_{t,k}$ is the probability of the most probable state sequence responsible for the first t observations that has k as its final state. The Viterbi path can be retrieved by saving back pointers that remember which state x was used in the second equation. Let $\mathrm{Ptr}(k,t)$ be the function that returns the value of x used to compute $V_{t,k}$ if $t > 1$, or k if $t = 1$. Then:

$$\begin{array}{rcl} x_T & = & \arg\max_{x \in S}(V_{T,x}) \\ x_{t-1} & = & \mathrm{Ptr}(x_t, t) \end{array}$$

Here we're using the standard definition of arg max.

The complexity of this algorithm is $O(T \times |S|^2)$.

Discriminative counterpart: CRFs

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\lambda_y + \sum_{j=1}^{K} \lambda_{y,j} x_j\right\}; \qquad p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y, \mathbf{x})\right\}.$$

$$f_{y',j}(y, \mathbf{x}) = \mathbf{1}_{\{y'=y\}} x_j \qquad f_{y'}(y, \mathbf{x}) = \mathbf{1}_{\{y'=y\}}$$

Let Y, X be random vectors, $\Lambda = \{\lambda_k\} \in \Re^K$ be a parameter vector, and $\{f_k(y, y', \mathbf{x}_t)\}_{k=1}^K$ be a set of real-valued feature functions. Then a *linear-chain conditional random field* is a distribution $p(\mathbf{y}|\mathbf{x})$ that takes the form

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}, \qquad (1.16)$$

where $Z(\mathbf{x})$ is an instance-specific normalization function

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}. \qquad (1.17)$$

Parameter estimation is typically performed by penalized maximum likelihood. Because we are modeling the conditional distribution, the following log likelihood, sometimes called the *conditional log likelihood*, is appropriate:

$$\ell(\theta) = \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}). \tag{1.18}$$

One way to understand the conditional likelihood $p(\mathbf{y}|\mathbf{x};\theta)$ is to imagine combining it with some arbitrary prior $p(\mathbf{x};\theta')$ to form a joint $p(\mathbf{y},\mathbf{x})$. Then when we optimize the joint log likelihood

$$\log p(\mathbf{y},\mathbf{x}) = \log p(\mathbf{y}|\mathbf{x};\theta) + \log p(\mathbf{x};\theta'), \tag{1.19}$$