CHAPTER

29 STRUCTURED CLASSIFICATION

CHAPTER CONTENTS

When classifying a set of input patterns having certain data structure such as a sequence of letters and a parsing tree of a sentence, the classification performance is expected to be improved if such structural information is utilized. In this chapter, taking a sequence of letters as an example of such structure, methods of structured classification are introduced.

29.1 SEQUENCE CLASSIFICATION

Let us consider the problem of classifying a sequence of m patterns where each pattern belongs to one of the c classes. Let $\overline{\boldsymbol{x}}$ and $\overline{y}$ be sequences of m patterns and m classes, and let $\boldsymbol{x}^{(k)}$ and $y^{(k)}$ be their kth elements:

$$\overline{\boldsymbol{x}} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}),$$
$$\overline{y} = (y^{(1)}, \ldots, y^{(m)}).$$

For example, when recognizing a sequence of five hand-written digits (see Fig. 29.1), $m = 5$ and $c = 10$. If such a sequence is decomposed into five digits, ordinary classification algorithms introduced in the previous chapters can be used for classifying each digit. However, this decomposition approach loses useful contextual information that can drastically improve the classification accuracy, for example, the same number may rarely appear consecutively and some number appears frequently after a certain number. On the other hand, if the entire sequence of digits is directly recognized, the number of classes grows exponentially with respect to the length of

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00040-6

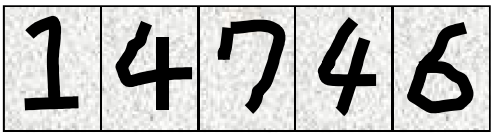


FIGURE 29.1

Classification of sequence of hand-written digits.

the sequence. In the example of Fig. 29.1, $(m, c) = (5, 10)$ and thus the number of classes is 10^5, which is hard to handle.

In this chapter, intermediate approaches are introduced, which take into account some contextual information with the computational complexity kept moderately. Note that, although only sequence data are considered, methods introduced in this chapter can be applied to more general structured data.

29.2 PROBABILISTIC CLASSIFICATION FOR SEQUENCES

In this section, the *logistic model* introduced in Section 28.1 is extended to the *conditional random field* [65] for sequence classification.

29.2.1 CONDITIONAL RANDOM FIELD

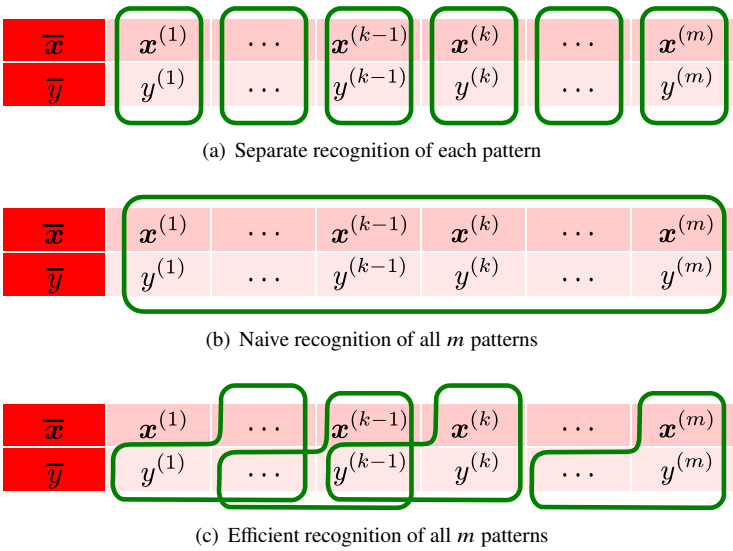
To introduce the conditional random field, let us first consider the logistic model applied to sequence classification, i.e. each pattern $\boldsymbol{x}^{(k)}$ is independently classified into one of the c classes (Fig. 29.2(a)):

$$q(y|\boldsymbol{x};\boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta}_y^\top \boldsymbol{\phi}(\boldsymbol{x})\right)}{\sum_{y'=1}^{c} \exp\left(\boldsymbol{\theta}_{y'}^\top \boldsymbol{\phi}(\boldsymbol{x})\right)},$$

where $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^b$ is the vector of basis functions for a single pattern $\boldsymbol{x}$, $\boldsymbol{\theta}_y \in \mathbb{R}^b$ is the parameter vector for class y, and

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_c^\top)^\top \in \mathbb{R}^{bc}$$

is the vector of all parameters. However, this decomposition approach cannot utilize the contextual information contained in sequence data.



(a) Separate recognition of each pattern

(b) Naive recognition of all m patterns

(c) Efficient recognition of all m patterns

FIGURE 29.2

Sequence classification.

On the other hand, if the entire sequence is classified at once, a classification problem with $\overline{c} = c^m$ classes needs to be solved (Fig. 29.2(b)):

$$q(\overline{y}|\overline{\boldsymbol{x}};\overline{\boldsymbol{\theta}}) = \frac{\exp\left(\overline{\boldsymbol{\theta}}_{\overline{y}}^{\top}\overline{\boldsymbol{\phi}}(\overline{\boldsymbol{x}})\right)}{\sum_{y^{(1)},\ldots,y^{(m)}=1}^{c} \exp\left(\overline{\boldsymbol{\theta}}_{\overline{y}'}^{\top}\overline{\boldsymbol{\phi}}(\overline{\boldsymbol{x}})\right)}, \tag{29.1}$$

where

$$\overline{\boldsymbol{\phi}}(\overline{\boldsymbol{x}}) = (\boldsymbol{\phi}(\boldsymbol{x}^{(1)})^{\top},\ldots,\boldsymbol{\phi}(\boldsymbol{x}^{(m)})^{\top})^{\top} \in \mathbb{R}^{bm}$$

is the vector of basis functions for pattern sequence $\overline{\boldsymbol{x}} = (\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)})$,

$$\overline{\boldsymbol{\theta}}_{\overline{y}} = (\boldsymbol{\theta}_{\overline{y}}^{(1)\top},\ldots,\boldsymbol{\theta}_{\overline{y}}^{(m)\top})^{\top} \in \mathbb{R}^{bm}$$

is the parameter vector for class $\overline{y} = (y^{(1)},\ldots,y^{(m)})$, and

$$\overline{\boldsymbol{\theta}} = (\overline{\boldsymbol{\theta}}_1{}^{\top},\ldots,\overline{\boldsymbol{\theta}}_{\overline{c}}{}^{\top})^{\top} \in \mathbb{R}^{bm\overline{c}}$$

is the vector of all parameters. However, since the number of classes $\overline{c} = c^m$ grows exponentially with respect to the sequence length m, handling the above logistic model may be intractable.

To avoid the exponential growth in the number of classes, let us assume that $y^{(k)}$ is determined only by $\boldsymbol{x}^{(k)}$ and $y^{(k-1)}$ when classifying the entire sequence $\overline{\boldsymbol{x}}$. Note that this approach is different from merely classifying two consecutive patterns into one of the c^2 classes, because of the overlap (see Fig. 29.2(c)).

To implement the above idea, let us consider the following model, called the *conditional random field*:

$$q(\overline{y}|\overline{\boldsymbol{x}};\boldsymbol{\zeta}) = \frac{\exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\overline{\boldsymbol{x}},\overline{y})\right)}{\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\overline{\boldsymbol{x}},\overline{y}')\right)}, \tag{29.2}$$

where the basis function $\boldsymbol{\varphi}(\overline{\boldsymbol{x}},\overline{y})$ depend not only on input $\overline{\boldsymbol{x}}$ but also on output $\overline{y}$. Conditional random field (29.2) is reduced to the logistic model for sequence $\overline{\boldsymbol{x}}$ given by Eq. (29.1), if

$$\boldsymbol{\zeta} = \overline{\boldsymbol{\theta}} \quad \text{and} \quad \boldsymbol{\varphi}(\overline{\boldsymbol{x}},\overline{y}) = \boldsymbol{e}_{\overline{y}}^{(\overline{c})} \otimes \overline{\boldsymbol{\phi}}(\overline{\boldsymbol{x}}),$$

where $\boldsymbol{e}_{\overline{y}}^{(\overline{c})}$ is the $\overline{c}$-dimensional indicator vector of class $\overline{y}$ (i.e. the element corresponding class $\overline{y}$ is 1 and all other elements are 0), and $\otimes$ denotes the *Kronecker product* (see Fig. 6.5):

$$\begin{aligned}&\text{For } \boldsymbol{f} = (f_1,\ldots,f_u)^\top \in \mathbb{R}^u,\ \boldsymbol{g} = (g_1,\ldots,g_v)^\top \in \mathbb{R}^v,\\ &\boldsymbol{f}\otimes\boldsymbol{g} = (f_1g_1, f_1g_2,\ldots,f_1g_v, f_2g_1, f_2g_2,\ldots,f_2g_v,\ldots,\\ &\qquad\qquad f_ug_1, f_ug_2,\ldots,f_ug_v)^\top \in \mathbb{R}^{uv}.\end{aligned}$$

Thus, naively using conditional random field (29.2) is just an overly redundant expression of Eq. (29.1).

Let us simplify conditional random field (29.2) by taking into account the structure described in Fig. 29.2(c). More specifically, basis function $\boldsymbol{\varphi}(\overline{\boldsymbol{x}},\overline{y})$ is defined as the sum of basis functions depending only on two consecutive patterns:

$$\boldsymbol{\varphi}(\overline{\boldsymbol{x}},\overline{y}) = \sum_{k=1}^{m} \boldsymbol{\varphi}(\boldsymbol{x}^{(k)}, y^{(k)}, y^{(k-1)}), \tag{29.3}$$

where $y^{(0)} = y^{(1)}$. As two-pattern basis function $\boldsymbol{\varphi}(\boldsymbol{x}^{(k)}, y^{(k)}, y^{(k-1)})$, for example,

$$\boldsymbol{\varphi}(\boldsymbol{x}^{(k)}, y^{(k)}, y^{(k-1)}) = \begin{pmatrix} \boldsymbol{e}_{y^{(k)}}^{(c)} \otimes \boldsymbol{\phi}(\boldsymbol{x}^{(k)}) \\ \boldsymbol{e}_{y^{(k)}}^{(c)} \otimes \boldsymbol{e}_{y^{(k-1)}}^{(c)} \end{pmatrix} \in \mathbb{R}^{cb+c^2}, \tag{29.4}$$

may be used. Then the dimensionality of parameter vector $\boldsymbol{\zeta}$ is $cb + c^2$, which is independent of the sequence length m.

1. Initialize $\boldsymbol{\zeta}$.
2. Choose the ith training sample randomly $(\overline{\boldsymbol{x}}_i, \overline{y}_i)$.
3. Update parameter $\boldsymbol{\zeta}$ to go up the gradient:

$$\boldsymbol{\zeta} \longleftarrow \boldsymbol{\zeta} + \varepsilon \left(\boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y}_i) - \frac{\displaystyle\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y})\right) \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y})}{\displaystyle\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y})\right)} \right),$$

   where $\varepsilon > 0$ is the step size and $\overline{y} = (y^{(1)}, \ldots, y^{(m_i)})$.
4. Iterate 2 and 3 until convergence.

FIGURE 29.3

Stochastic gradient algorithm for conditional random field.

29.2.2 MLE

Suppose that labeled sequences,

$$\left\{ (\overline{\boldsymbol{x}}_i, \overline{y}_i) \,\middle|\, \overline{\boldsymbol{x}}_i = (\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(m_i)}),\ \overline{y}_i = (y_i^{(1)}, \ldots, y_i^{(m_i)}) \right\}_{i=1}^{n},$$

are provided as training samples, where the length m_i of sequence $\overline{\boldsymbol{x}}_i$ can be different for each sample. With these training samples, the parameter $\boldsymbol{\zeta}$ in conditional random field (29.2) is learned by *MLE* (see Chapter 12):

$$\max_{\boldsymbol{\zeta}} \sum_{i=1}^{n} \log \frac{\exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y}_i)\right)}{\displaystyle\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y})\right)},$$

where

$$\boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y}_i) = \sum_{k=1}^{m_i} \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y_i^{(k)}, y_i^{(k-1)}).$$

The maximum likelihood solution may be obtained by a *stochastic gradient* algorithm (Section 15.3), as described in Fig. 29.3.

29.2.3 RECURSIVE COMPUTATION

However, the computational complexity of the second term of the gradient (see Fig. 29.3),

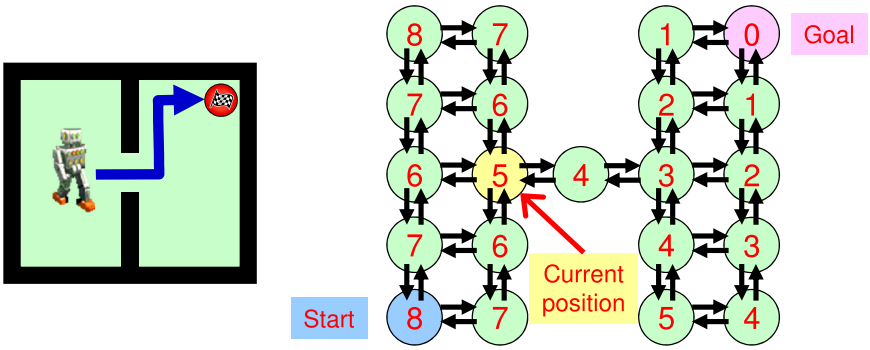


FIGURE 29.4

Dynamic programming, which solves a complex optimization problem by breaking it down into simpler subproblems recursively. When the number of steps to the goal is counted, dynamic programming trace back the steps from the goal. In this case, many subproblems of counting the number of steps from other positions are actually shared and thus dynamic programming can efficiently reuse the solutions to reduce the computation costs.

$$\frac{\displaystyle\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y})\right) \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y})}{\displaystyle\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y})\right)}, \tag{29.5}$$

grows exponentially with respect to the sequence length m_i. To cope with this problem, let us utilize the decomposition given by (29.3) based on *dynamic programming* (Fig. 29.4).

First, let us split the summation in the denominator of Eq. (29.5) into $(y^{(1)}, \ldots, y^{(m_i-1)})$ and $y^{(m_i)}$:

$$\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\sum_{k=1}^{m_i} \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)})\right) = \sum_{y^{(m_i)}=1}^{c} A_{m_i}(y^{(m_i)}), \tag{29.6}$$

where

$$A_\tau(y) = \sum_{y^{(1)},\ldots,y^{(\tau-1)}=1}^{c} \exp\left(\sum_{k=1}^{\tau-1} \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)}) + \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(\tau)}, y, y^{(\tau-1)})\right).$$

$A_\tau(y^{(\tau)})$ can be expressed recursively using $A_{\tau-1}(y^{(\tau-1)})$ as

$$\begin{aligned} A_\tau(y^{(\tau)}) &= \sum_{y^{(1)},\ldots,y^{(\tau-1)}=1}^{c} \exp\left(\sum_{k=1}^{\tau} \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)})\right) \\ &= \sum_{y^{(\tau-1)}=1}^{c} A_{\tau-1}(y^{(\tau-1)}) \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(\tau)}, y^{(\tau)}, y^{(\tau-1)})\right). \end{aligned}$$

The computational complexity of naively computing $A_{m_i}(y^{(m_i)})$ is $O(c^{m_i})$, while that of computing $A_{m_i}(y^{(m_i)})$ using the above recursive expression from $A_1(y^{(1)})$ is only $O(c^2 m_i)$. Note that the complexity for computing $\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}, y, y')$ is not included here.

Next, let us split the summation in the numerator of Eq. (29.5) into $(y^{(1)}, \ldots, y^{(k'-2)})$, $(y^{(k'-1)}, y^{(k')})$, and $(y^{(k'+1)}, \ldots, y^{(m_i)})$:

$$\begin{aligned} &\sum_{y^{(1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\sum_{k=1}^{m_i} \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)})\right)\left(\sum_{k'=1}^{m_i} \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k')}, y^{(k')}, y^{(k'-1)})\right) \\ &= \sum_{k'=1}^{m_i} \sum_{y^{(1)},\ldots,y^{(k'-2)}=1}^{c} \sum_{y^{(k'-1)},y^{(k')}=1}^{c} \sum_{y^{(k'+1)},\ldots,y^{(m_i)}=1}^{c} \\ &\qquad \exp\left(\sum_{k=1}^{m_i} \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)})\right) \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k')}, y^{(k')}, y^{(k'-1)}) \\ &= \sum_{k'=1}^{m_i} \sum_{y^{(k'-1)},y^{(k')}=1}^{c} \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k')}, y^{(k')}, y^{(k'-1)}) \\ &\qquad \times \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k')}, y^{(k')}, y^{(k'-1)})\right) A_{k'-1}(y^{(k'-1)}) B_{k'}(y^{(k')}), \end{aligned} \tag{29.7}$$

where

$$\begin{aligned} B_\tau(y) = \sum_{y^{(\tau+1)},\ldots,y^{(m_i)}=1}^{c} \exp\Bigg(&\sum_{k=\tau+2}^{m_i} \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)}) \\ &+ \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(\tau+1)}, y^{(\tau+1)}, y)\Bigg). \end{aligned}$$

$B_\tau(y^{(\tau)})$ can be expressed recursively using $B_{\tau+1}(y^{(\tau+1)})$ as

$$\begin{aligned} B_\tau(y^{(\tau)}) &= \sum_{y^{(\tau+1)},\ldots,y^{(m_i)}=1}^{c} \exp\left(\sum_{k=\tau+1}^{m_i} \boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)})\right) \\ &= \sum_{y^{(\tau+1)}=1}^{c} B_{\tau+1}(y^{(\tau+1)}) \exp\left(\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}_i^{(\tau+1)}, y^{(\tau+1)}, y^{(\tau)})\right). \end{aligned}$$

The computational complexity of naively computing $B_1(y^{(1)})$ is $O(c^{m_i})$, while that of computing $B_1(y^{(1)})$ using the above recursive expression from $B_{m_i}(y^{(m_i)})$ is only $O(c^2 m_i)$. Note that the complexity for computing $\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}, y, y')$ is not included here.

Summarizing the above computations, stochastic gradient ascent for the conditional random field can be performed efficiently by computing $\{A_k(y^{(k)})\}_{k=1}^{m_i}$ and $\{B_k(y^{(k)})\}_{k=1}^{m_i}$ in advance using the recursive expressions.

29.2.4 PREDICTION FOR NEW SAMPLE

Trained conditional random field $q(\overline{y}|\overline{\boldsymbol{x}}; \widehat{\boldsymbol{\zeta}})$ allows us to predict the most probable label sequence $\overline{y} = (y^{(1)}, \ldots, y^{(m)})$ for a test pattern sequence $\overline{\boldsymbol{x}} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)})$ as

$$\underset{y^{(1)},\ldots,y^{(m)} \in \{1,\ldots,c\}}{\operatorname{argmax}} q(\overline{y}|\overline{\boldsymbol{x}}; \widehat{\boldsymbol{\zeta}}).$$

However, the computational complexity for naive maximization over $y^{(1)}, \ldots, y^{(m)}$ grows exponentially with respect to the sequence length m. Here, dynamic programming is used again to speed up the maximization.

Let us simplify the maximization problem as

$$\underset{y^{(1)},\ldots,y^{(m)} \in \{1,\ldots,c\}}{\operatorname{argmax}} \frac{\exp\left(\widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\overline{\boldsymbol{x}}, \overline{y})\right)}{\sum_{\overline{y}=1}^{\overline{c}} \exp\left(\widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\overline{\boldsymbol{x}}, \overline{y})\right)} = \underset{y^{(1)},\ldots,y^{(m)} \in \{1,\ldots,c\}}{\operatorname{argmax}} \widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\overline{\boldsymbol{x}}, \overline{y})$$

and decompose the maximization in the right-hand side into $(y^{(1)}, \ldots, y^{(m-1)})$ and $y^{(m)}$:

$$\max_{y^{(1)},\ldots,y^{(m)} \in \{1,\ldots,c\}} \widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\overline{\boldsymbol{x}}, \overline{y}) = \max_{y^{(m)} \in \{1,\ldots,c\}} P_m(y^{(m)}),$$

where

$$P_\tau(y) = \max_{y^{(1)},\ldots,y^{(\tau-1)} \in \{1,\ldots,c\}} \left[\sum_{k=1}^{\tau-1} \widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\boldsymbol{x}^{(k)}, y^{(k)}, y^{(k-1)}) + \widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\boldsymbol{x}^{(\tau)}, y, y^{(\tau-1)}) \right].$$

$P_\tau(y^{(\tau)})$ can be expressed by recursively using $P_{\tau-1}(y^{(\tau-1)})$ as

$$\begin{aligned} P_\tau(y^{(\tau)}) &= \max_{y^{(1)},\ldots,y^{(\tau-1)} \in \{1,\ldots,c\}} \left[\sum_{k=1}^{\tau} \widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\boldsymbol{x}^{(k)}, y^{(k)}, y^{(k-1)}) \right] \\ &= \max_{y^{(\tau-1)} \in \{1,\ldots,c\}} \left[P_{\tau-1}(y^{(\tau-1)}) + \widehat{\boldsymbol{\zeta}}^\top \boldsymbol{\varphi}(\boldsymbol{x}^{(\tau)}, y^{(\tau)}, y^{(\tau-1)}) \right]. \end{aligned}$$

The computational complexity of naively computing $P_m(y^{(m)})$ is $O(c^m)$, while that of computing $P_m(y^{(m)})$ using the above recursive expression from $P_1(y^{(1)})$ is only $O(c^2 m)$. Note that the complexity for computing $\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\boldsymbol{x}, y, y')$ is not included here.

29.3 DETERMINISTIC CLASSIFICATION FOR SEQUENCES

The conditional random field models the class-posterior probability, which corresponds to a probabilistic classification method. On the other hand, in deterministic classification, only the class label is learned. In this section, based on the formulation of *multiclass support vector classification* given in Section 27.5, a deterministic sequence classification method is introduced. More specifically,

$$f_{\boldsymbol{w}_y}(\boldsymbol{x}) = \boldsymbol{w}_y^\top \boldsymbol{x}$$

is used as a score function for class y, and a test sample $\boldsymbol{x}$ is classified into class $\widehat{y}$ as

$$\widehat{y} = \operatorname*{argmax}_{y=1,\ldots,c} f_{\boldsymbol{w}_y}(\boldsymbol{x}).$$

The parameter $\boldsymbol{w}_y$ is learned as

$$\begin{aligned}&\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_c,\boldsymbol{\xi}} \left[\frac{1}{2}\sum_{y=1}^{c}\|\boldsymbol{w}_y\|^2 + C\sum_{i=1}^{n}\xi_i\right]\\ &\text{subject to } \boldsymbol{w}_{y_i}^\top \boldsymbol{x}_i - \boldsymbol{w}_y^\top \boldsymbol{x}_i \geq t_{i,y} - \xi_i, \quad \forall i = 1,\ldots,n,\ \forall y = 1,\ldots,c,\end{aligned}$$

where

$$t_{i,y} = \begin{cases} 0 & (y = y_i),\\ 1 & (y \neq y_i).\end{cases}$$

Note that, when $y = y_i$, the constraint $\boldsymbol{w}_{y_i}^\top \boldsymbol{x}_i - \boldsymbol{w}_y^\top \boldsymbol{x}_i \geq t_{i,y} - \xi_i$ is reduced to $\xi_i \geq 0$.

Applying the feature expression of the conditional random field, $\boldsymbol{\zeta}^\top \boldsymbol{\varphi}(\overline{\boldsymbol{x}}, \overline{y})$, to the above optimization yields

$$\begin{aligned}&\min_{\boldsymbol{\zeta},\boldsymbol{\xi}} \left[\frac{1}{2}\|\boldsymbol{\zeta}\|^2 + C\sum_{i=1}^{n}\xi_i\right]\\ &\text{subject to } \boldsymbol{\zeta}^\top \Delta\boldsymbol{\varphi}_i(\overline{y}) \geq t_{i,\overline{y}} - \xi_i,\\ &\qquad \forall i = 1,\ldots,n,\ \forall \overline{y} = \underbrace{(1,\ldots,1)}_{m},\ldots,\underbrace{(c,\ldots,c)}_{m},\end{aligned}$$

where

$$\Delta\boldsymbol{\varphi}_i(\overline{y}) = \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y}_i) - \boldsymbol{\varphi}_i(\overline{\boldsymbol{x}}_i, \overline{y}).$$

The *Lagrange dual* (Fig. 23.5) of the above optimization problem is given by

$$\begin{aligned}&\max_{\{\alpha_{i,\overline{y}}\}_{i,\overline{y}}} J(\{\alpha_{i,\overline{y}}\}_{i,\overline{y}})\\ &\text{subject to } \sum_{y^{(1)},\ldots,y^{(m)}=1}^{c} \alpha_{i,\overline{y}} = C, \quad \forall i = 1,\ldots,n,\\ &\qquad \alpha_{i,\overline{y}} \geq 0, \quad \forall i = 1,\ldots,n,\ \forall y^{(1)},\ldots,y^{(m)} = 1,\ldots,c,\end{aligned} \tag{29.8}$$

where

$$J(\{\alpha_{i,\overline{y}}\}_{i,\overline{y}}) = \sum_{i=1}^{n} \sum_{y^{(1)},\ldots,y^{(m)}=1}^{c} \alpha_{i,\overline{y}} t_{i,\overline{y}} - \frac{1}{2} \sum_{i,i'=1}^{n} \sum_{y^{(1)},\ldots,y^{(m)}=1}^{c} \sum_{y'^{(1)},\ldots,y'^{(m)}=1}^{c} \alpha_{i,\overline{y}} \alpha_{i',\overline{y}'} \Delta\boldsymbol{\varphi}_i(\overline{y})^\top \Delta\boldsymbol{\varphi}_{i'}(\overline{y}').$$

This is a quadratic programming problem and thus the solution $\{\widehat{\alpha}_{i,\overline{y}}\}_{i,\overline{y}}$ can be obtained by standard optimization software in principle. However, the number of optimization variables is exponential with respect to the sequence length, m, and thus it is not computationally tractable.

Here, let us utilize the same simplification trick as the conditional random field:

$$\boldsymbol{\varphi}(\overline{\boldsymbol{x}}, \overline{y}) = \sum_{k=1}^{m} \boldsymbol{\varphi}(\boldsymbol{x}^{(k)}, y^{(k)}, y^{(k-1)}),$$

which yields

$$\Delta\boldsymbol{\varphi}_i(\overline{y}) = \sum_{k=1}^{m_i} \Delta\boldsymbol{\varphi}_i(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)}),$$

where

$$\Delta\boldsymbol{\varphi}_i(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)}) = \boldsymbol{\varphi}_i(\boldsymbol{x}_i^{(k)}, y_i^{(k)}, y_i^{(k-1)}) - \boldsymbol{\varphi}_i(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)}).$$

Then the objective function in optimization problem (29.8) can be expressed as

$$J(\{\alpha_{i,\overline{y}}\}_{i,\overline{y}}) = \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{y^{(k)}=1}^{c} \mu_i(y^{(k)}) t_{i,\overline{y}} - \frac{1}{2} \sum_{i,i'=1}^{n} \sum_{k,k'=1}^{m} \sum_{y^{(k-1)},y^{(k)}=1}^{c} \sum_{y^{(k'-1)},y^{(k')}=1}^{c} \Delta\boldsymbol{\varphi}_i(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)})^\top \times \Delta\boldsymbol{\varphi}_{i'}(\boldsymbol{x}_{i'}^{(k')}, y^{(k')}, y^{(k'-1)}) \mu_i(y^{(k-1)}, y^{(k)}) \mu_i(y^{(k'-1)}, y^{(k')}),$$

where

$$\mu_i(y^{(k)}) = \sum_{y^{(1)},\ldots,y^{(k-1)},y^{(k+1)},\ldots,y^{(m_i)}=1}^{c} \alpha_{i,\overline{y}},$$

$$\mu_i(y^{(k-1)}, y^{(k)}) = \sum_{y^{(1)},\ldots,y^{(k-2)},y^{(k+1)},\ldots,y^{(m_i)}=1}^{c} \alpha_{i,\overline{y}}.$$

Thus, exponentially many variables $\{\alpha_{i,\overline{y}}\}_{i,\overline{y}}$ do not have to be optimized, but only optimizing $\{\mu_i(y^{(k)})\}_{k=1}^{m_i}$ and $\{\mu_i(y^{(k-1)}, y^{(k)})\}_{k=1}^{m_i}$ for $i = 1,\ldots,n$ is sufficient.

However, since $\{\mu_i(y^{(k)})\}_{k=1}^{m_i}$ and $\{\mu_i(y^{(k-1)}, y^{(k)})\}_{k=1}^{m_i}$ are related to each other, the additional constraint,

$$\sum_{y^{(k-1)}=1}^{c} \mu_i(y^{(k-1)}, y^{(k)}) = \mu_i(y^{(k)}),$$

is necessary, together with the original constraints:

$$\sum_{y^{(k-1)}=1}^{c} \mu_i(y^{(k)}) = C \quad \text{and} \quad \mu_i(y^{(k-1)}, y^{(k)}) \geq 0.$$

From the dual solution $\{\widehat{\mu}_i(y^{(k)})\}_{k=1}^{m_i}$ and $\{\widehat{\mu}_i(y^{(k-1)}, y^{(k)})\}_{k=1}^{m_i}$, the primal solution $\widehat{\boldsymbol{\zeta}}$ can be obtained as

$$\widehat{\boldsymbol{\zeta}} = \sum_{i=1}^{n} \sum_{k=1}^{m_i} \sum_{y^{(k-1)}, y^{(k)}=1}^{c} \widehat{\mu}_i(y^{(k-1)}, y^{(k)}) \Delta\boldsymbol{\varphi}_i(\boldsymbol{x}_i^{(k)}, y^{(k)}, y^{(k-1)}).$$

Once the solution $\widehat{\boldsymbol{\zeta}}$ is obtained, classification of a test sample $\overline{\boldsymbol{x}} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)})$,

$$\underset{y^{(1)}, \ldots, y^{(m)} \in \{1, \ldots, c\}}{\operatorname{argmax}} \quad \widehat{\boldsymbol{\zeta}}^{\top} \boldsymbol{\varphi}(\overline{\boldsymbol{x}}, \overline{y}),$$

can be performed exactly in the same way as that described in Section 29.2.4.

This method is referred to as the *structured support vector machine* [114].