Chapter 6

Electronic Ink Analysis

6.1 Introduction

This thesis has so far only considered plain text documents, modelled as simple strings of tokens. No use has been made of richer forms of information which may be available, such as the layout of the text or multimedia elements. This final chapter makes a departure from this theme to consider the task of interpreting documents containing hand drawn electronic ink diagrams. These diagrams are typically entered using a stylus directly onto the screen of the device. Devices which are capable of this kind of interaction are becoming increasingly widely available, including Tablet PCs, Personal Digital Assistants (PDAs) and high end portable telephones.

Examples of the sort of diagram considered in this chapter are shown in Figure 6.1. The particular task which is of interest here is that of grouping fragments of ink strokes into perceptually meaningful objects (in other words, groups which would be recognised as an object by a human), and to label those groups as belonging to a particular object class. The approach will be illustrated on organisational charts such as those shown in Figure 6.1, where the object classes will be containers, which are box-like objects representing members of the organisation, and connectors, which are lines representing the relationships between members. As the diagram is created by direct digitisation of the stylus movement, there is no need to reconstruct ink strokes from a bitmap image as the data is already available. Furthermore, the temporal ordering of the strokes is available, which will be shown to be useful in the interpretation process.

A number of other attempts have been made to extract perceptually relevant objects from drawings [82] [83] [54], although these methods have not in general used a principled probabilistic framework. The work presented here was done with the assistance of Dr Martin Szummer, who provided the raw data, the definition of the feature set and library code used to perform basic manipulation of the input data and the resulting undirected graphs. Material in this chapter was presented in [90] and [23].

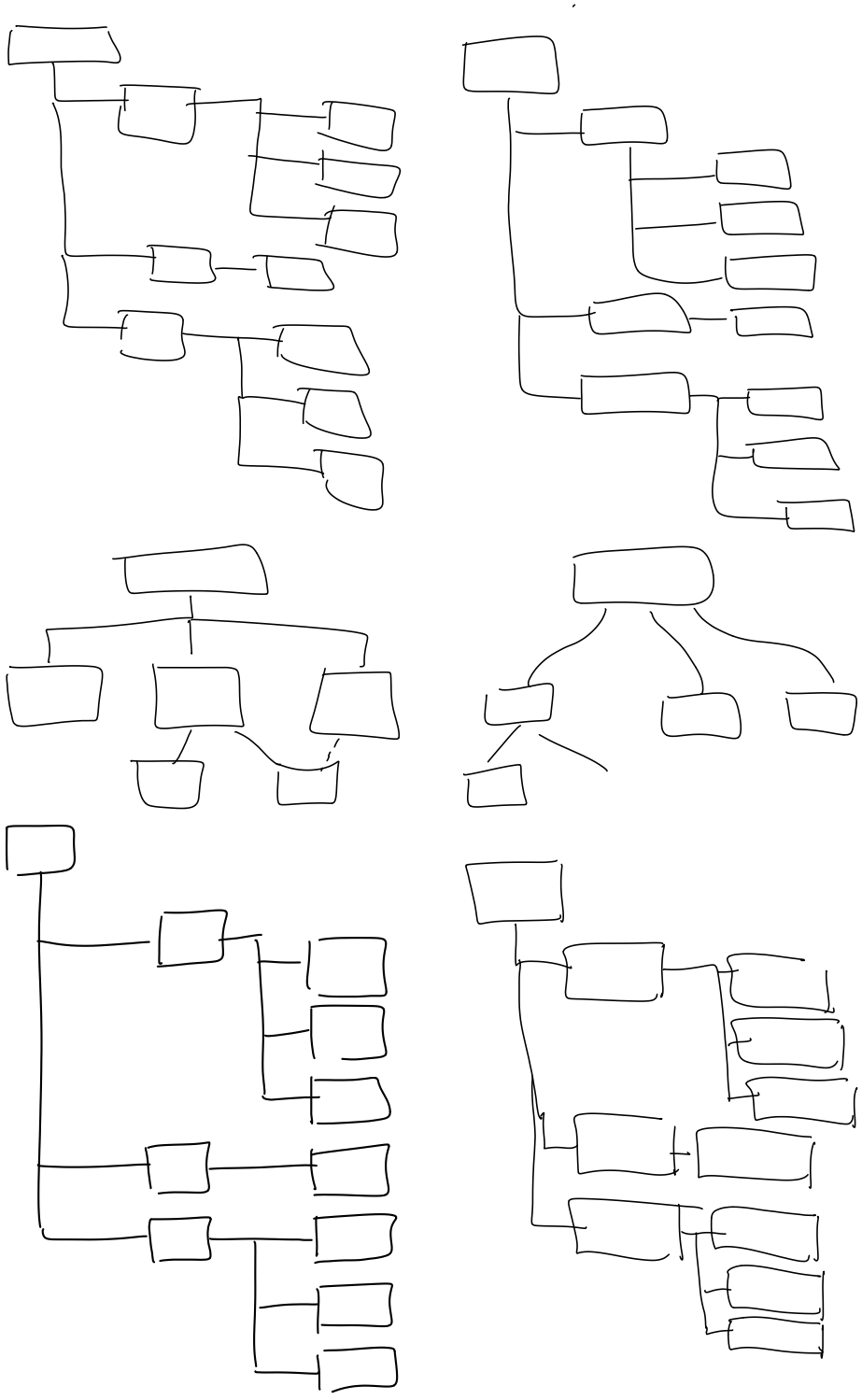
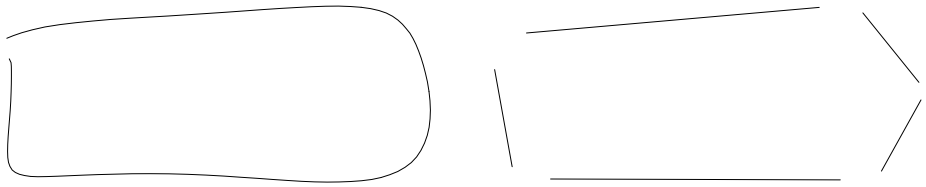
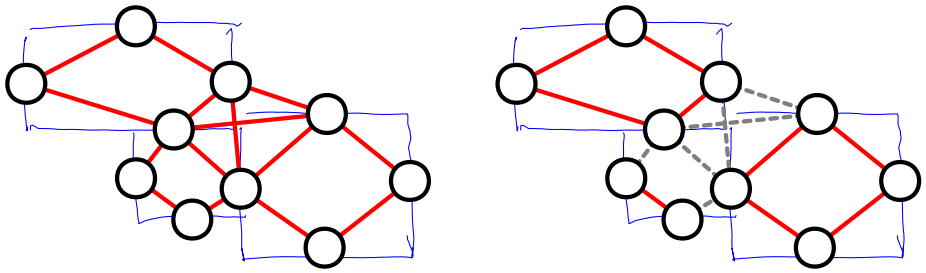


Figure 6.1: Six examples of ink diagrams used as input to the interpretation system.



(a) (b)

Figure 6.2: Division of the original input data (a) into fragments (b). The division is done on the basis that fragments must be straight to within a given tolerance.



(a) (b)

Figure 6.3: (a) An example of an undirected graph constructed from the input data in which each vertex represents an ink fragment. (b) Partition of the graph representing objects in the diagram. Dashed grey lines represent edges connecting vertices which are in different parts.

6.2 Overview of the algorithm

The input data is a set of ink strokes, which may span multiple objects. The first stage of the interpretation process is to split the strokes into fragments, which are assumed to belong to a single object. Fragments are defined to be sections of strokes which are straight to within a given tolerance (see Figure 6.2).

Having fragmented the strokes, an undirected graph, $\mathcal{G}$, containing one vertex for each ink fragment is constructed (See Figure 6.3(a)). Precise details of the construction of $\mathcal{G}$ will be given in Section 6.7.1. The goal of the algorithm will be to partition this graph so that each part contains the vertices corresponding to ink fragments which make up a single object, and to label each part according to the corresponding object class (See Figure 6.3(b)). The approach which will be taken is to construct a probabilistic model over labelled partitions conditioned on observed ink data.

6.3 Undirected graphical models

Multivariate probability distributions often contain internal structure, which can be exploited in order to make computation more efficient. Any representation which elucidates this structure is therefore desirable. The search for such representations has a long history, and has generated methods such as Bayesian networks [38], undirected graphical models [51] and factor graphs [45].

The most relevant of these representations is the undirected graphical model. Consider a distribution over three variables, y_1, y_2 and y_3, denoted $P(y_1, y_2, y_3)$. In the most general case, the distribution is specified by a function of all three variables. In some cases however, it may be possible to represent the distribution as a product of two functions (referred to as *potentials*), one of y_1 and y_2 only, and the other of y_2 and y_3 only. In other words

$$P(y_1, y_2, y_3) = \frac{1}{Z}\psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.1}$$

where Z is a normalising constant,

$$Z = \sum_{y_1, y_2, y_3} \psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.2}$$

Now suppose that y_2 is observed. In this case the conditional distribution can be written

$$P(y_1, y_3 \mid y_2) = \frac{1}{Z'}\psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.3}$$

where

$$Z' = \sum_{y_1, y_3} \psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.4}$$

Conditioning further on y_3,

$$\begin{aligned} P(y_1 \mid y_2, y_3) &= \frac{P(y_1, y_3 \mid y_2)}{P(y_3 \mid y_2)} && (6.5)\\ &= \frac{\psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3)}{\sum_{y_1} \psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3)} && (6.6)\\ &= \frac{\psi_{23}(y_2, y_3) \cdot \psi_{12}(y_1, y_2)}{\psi_{23}(y_2, y_3) \cdot \sum_{y_1} \psi_{12}(y_1, y_2)} && (6.7)\\ &= \frac{\psi_{12}(y_1, y_2)}{\sum_{y_1} \psi_{12}(y_1, y_2)} && (6.8) \end{aligned}$$

The conditional distribution is independent of y_3. In other words, y_1 and y_3 are *conditionally independent* [25] given y_2, written $y_1 \perp\!\!\!\perp y_3 \mid y_2$.

This structure can be represented by constructing a graph consisting of a vertex for each variable and an edge connecting the vertices corresponding to any pair of variables which appear together in a potential in the full distribution. Viewed another way, there is one potential in the distribution for each clique, or fully connected subgraph in the graph, so the distribution corresponding to an arbitrary graph can be written as

$$P(\{x_i\}) = \frac{1}{Z}\prod_{j \in \mathcal{C}} \psi_j\left(\{x_i\}_j\right) \tag{6.9}$$

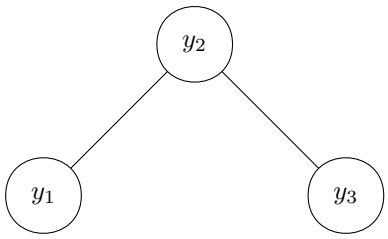


Figure 6.4: The undirected graphical model corresponding to the probability distribution given in (6.1). Vertices correspond to the variables over which the distribution is defined, and cliques, which in this case are (y_1, y_2) and (y_2, y_3), correspond to the potentials.

Where $\mathcal{C}$ is the set of cliques and the notation $\{x_i\}_j$ is used to represent the set of vertices in clique j. The graph corresponding to (6.1) is shown in Figure 6.4. In general, two variables y_i and y_j are conditionally independent given a subset of the variables if the subset *separates* y_i and y_j on the graph, or equivalently, if all paths between y_i and y_j on the graph pass through a member of the subset. This result has powerful implications in terms of the efficiency of computation using the model [51], a topic which will be returned to in detail in a later section.

Moral graphs [38] derived from Bayesian networks, constitute a special case of the undirected graphical model, where the potentials have a simple interpretation as the conditional probability tables of each variable given its parents.

6.3.1 Conditional random fields

Conditional random fields are a particular type of undirected graphical model used when a conditional distribution is required based on observed data [48]. Let $\boldsymbol{x}$ be a vector representing the observations, and $\{y_i\}$ be the set of variables over which the distribution is to be defined. As in Section 6.3, when conditioned on $\boldsymbol{x}$, the model has an independence structure represented by an undirected graph whose nodes are variables in $\{y_i\}$. To produce such a distribution, we use potentials which depend on *features* of the observed data. Typically, these are binary features defined for all vertices and edges on the graph, denoted $f_{ik}^{(1)}(\boldsymbol{x})$ and $f_{ijk}^{(2)}(\boldsymbol{x})$ respectively. The model has the form

$$P(\{y_i\} \mid \boldsymbol{x}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi_i(y_i, \boldsymbol{x}) \prod_{ij \in \mathcal{E}} \psi_{ij}(y_i, y_j, \boldsymbol{x}) \tag{6.10}$$

where $\mathcal{V}$ and $\mathcal{E}$ are the vertices and edges of the graph respectively. The potentials are given by

$$\psi_i(y_i, \boldsymbol{x}) = \exp \sum_k f_{ik}^{(1)}(\boldsymbol{x}) w_k^{(1)}(y_i) \tag{6.11}$$

$$\psi_{ij}(y_i, y_j, \boldsymbol{x}) = \exp \sum_k f_{ijk}^{(2)}(\boldsymbol{x}) w_k^{(2)}(y_i, y_j) \tag{6.12}$$

where $w^{(1)}$ and $w^{(2)}$ are weights which are associated with the values of $\{\boldsymbol{y}\}$. In many cases binary features are used, taking the values $+1$ and -1. Roughly speaking, positive weights indicate that the corresponding labels are compatible with a particular feature and negative weights indicate incompatibility, with larger magnitudes increasing the strength of the effect. The features are dependent on the index of the clique to which they are applied, which allows systematic variation where there is a natural correspondance between the observed data and the variables over which the distribution is defined. For example, features which may be used in a natural language processing setting include "The word at position i is 'the'" and "Position i is in the first 3 words of a sentence". One of the key strengths of the conditional random field is the arbitrary way in which the distribution can depend on the data. There is no computational cost involved in extending the range of the data dependences beyond that associated with the computation of the feature vectors themselves. Note that the simple model presented here can be generalised in a straightforward way, for example to include features that span larger cliques, or that take a continuous value rather than being binary.

The original applications of conditional random fields were in tasks associated with natural language processing, such as part-of-speech tagging or shallow parsing [85]. However, the approach is general and may be applied to any graph structure for which inference in the corresponding undirected model is tractable. Conditional random fields, including a Bayesian extension of the original maximum likelihood approach, have previously been applied to the same data set as used below but for the simpler task of labelling ink fragments without reference to partitioning [68]. Other work [88] has extended the CRF to perform multiple inference tasks simultaneously but has not considered partitioning of non-chain graphs (graphs with a chain- or tree-structure generally result in relatively simple inference problems. See later for a discussion of computational requirements). For an overview of conditional random fields see [94].

6.3.2 Models over partitions

Before introducing the approach which will be used in the remainder of this chapter, it is instructive to consider another possibility. One simple way of constructing a model over partitions would be simply to view the process as a labelling task, assigning each vertex a label and then defining the parts to be contiguous regions with the same label. This approach yields a standard labelling problem, so a conditional random field would be well suited to the task. There are however a number of serious problems.

Firstly, it is difficult to know in advance how many labels are necessary. In order to make sure that any partition can be realised, the number of labels must be equal to the size of the largest clique in the graph. The computational complexity of the algorithm scales as L^C where L is the number of labels and C is the size of the largest clique in the graph. This complexity potentially limits the extent to which this approach can be applied to real-life problems (see Section 6.6 for a more detailed discussion of computational requirements). A second, and possibly more serious problem is that representation in terms of labels introduces

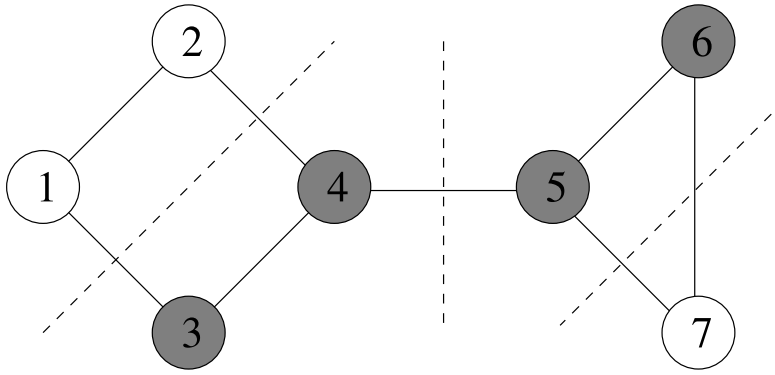


Figure 6.5: An example of a labelled partition. Vertices are partitioned as follows: $(1, 2, +)$, $(3, 4, -)$, $(5, 6, -)$, $(7, +)$, where the last symbol in each group indicates the label assigned to that part.

a degeneracy: the particular choice of labels is unimportant. In other words, the labels can be permuted without affecting the final partition. The number of ways of labelling a given partition depends on the number of parts present, so for example, if three mutually connected vertices are labelled with three labels, there are three assignments corresponding to the partition in which all vertices belong to the same part, but six assignments which correspond to the fully disjoint partition. If probability mass is assigned evenly amongst the possible ways of labelling the partition, it is possible that the disjoint case, even if it is more probable, might not be the single most probable arrangement, simply because the mass is spread more thinly.

To avoid these problems, the approach taken here is to extend the graphical model framework to work directly with labelled partitions of the graph. The following section develops these extensions from a theoretical viewpoint. Application to the specific problem of ink analysis will be left until later.

6.4 Theoretical development

Let $\mathcal{G}$ be an undirected graph as described in Section 6.2, consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$. Assume that $\mathcal{G}$ is triangulated, so that every cycle of length greater than three is spanned by a chord. Triangulation can always be achieved by adding edges, but usually at the expense of increasing the maximum clique size, and therefore computational complexity.

Let S be a partition of $\mathcal{G}$, that is, a set of non-empty subsets of $\mathcal{V}$, such that each vertex in $\mathcal{V}$ is a member of precisely one subset. Each subset is referred to as a *part* of $\mathcal{G}$. In the following, the term *partition* will always refer to a *contiguous* partition:

Definition 1. *A partition of $\mathcal{G}$ is **contiguous** if and only if all parts are internally connected. In other words, if i and j are vertices contained within the same part, there exists a path on $\mathcal{G}$ between i and j entirely contained within that part.*

A labelled partition of $\mathcal{G}$ is represented by $\mathcal{Y} = (\mathsf{S}, \boldsymbol{y})$, where S describes the partition and $\boldsymbol{y} \in \{-1, +1\}^M$ is a vector containing the labels associated with each part. For example, a partition of seven elements into four parts could be $\mathsf{S} = \{\{1,2\}\{3,4\}\{5,6\}\{7\}\}$, $\boldsymbol{y} = [+1, -1, -1, +1]$. This configuration is shown in Figure 6.5. Let $\mathbb{Y}$ be the set of all possible labelled partitions of $\mathcal{G}$. Note that M, the length of $\boldsymbol{y}$, depends on S. Let t_i be the index of the part to which vertex i is assigned, so that y_{t_i} is the label given to that vertex. The conditional probability distribution over $\mathbb{Y}$ has the form

$$P(\mathcal{Y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{i \in \mathcal{V}} \psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}) \prod_{i,j \in \mathcal{E}} \psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}), \tag{6.13}$$

where $\boldsymbol{x}$ is the observed data, $\boldsymbol{\theta}$ is a vector representing the model parameters collectively, and $Z(\boldsymbol{\theta})$ is a normalisation constant. $\psi_i^{(1)}$ are unary potentials defined for each vertex, and $\psi_{ij}^{(2)}$ are pairwise potentials defined for each edge. As for the standard CRF, the unary potentials introduce a data-dependent bias towards assigning one label or the other to each vertex. The pairwise potentials model the compatibility between the parts and labels of neighbouring vertices, and are also data dependent. These potentials depend on $\boldsymbol{x}$ through feature vectors, $\boldsymbol{g}_i$ and $\boldsymbol{f}_{ij}$, defined for each vertex i and edge (i,j) respectively. The potentials have the form

$$\psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi(\boldsymbol{w}_+ \cdot \boldsymbol{g}_i(\boldsymbol{x})) & \text{if } y_{t_i} = +1 \\ \phi(\boldsymbol{w}_- \cdot \boldsymbol{g}_i(\boldsymbol{x})) & \text{if } y_{t_i} = -1 \end{cases}, \tag{6.14}$$

where $\phi(\cdot)$ is a non-linear mapping, and $\boldsymbol{w}_+$ and $\boldsymbol{w}_-$ are vectors of feature weights depending on the label of the appropriate vertex. For now, an exponential non-linearity, $\phi : x \mapsto \exp(x)$ will be used, although in general other functions may be substituted (for example, the probit function). The pairwise potentials are defined by

$$\psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi\left(\boldsymbol{v}_{\mathrm{ss}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } t_i = t_j,\ y_{t_i} = y_{t_j} \\ \phi\left(\boldsymbol{v}_{\mathrm{sd}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } t_i \neq t_j,\ y_{t_i} = y_{t_j} \\ \phi\left(\boldsymbol{v}_{\mathrm{dd}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } t_i \neq t_j,\ y_{t_i} \neq y_{t_j} \end{cases} \tag{6.15}$$

where $\boldsymbol{v}_{\mathrm{ss}}$, $\boldsymbol{v}_{\mathrm{sd}}$ and $\boldsymbol{v}_{\mathrm{dd}}$ are vectors of feature weights to be used when i and j belong to the same part, different parts with the same label, and different parts with different labels respectively. The fourth case, corresponding to vertices with different labels in the same part, does not occur by definition. The parameters in $\boldsymbol{\theta}$ are therefore $(\boldsymbol{w}_+, \boldsymbol{w}_-, \boldsymbol{v}_{\mathrm{ss}}, \boldsymbol{v}_{\mathrm{sd}}, \boldsymbol{v}_{\mathrm{dd}})$. As adding a constant to all weights which correspond to a particular feature simply results in multiplication of the potentials by a constant, there is a redundancy in the weight vectors. In practice, $\boldsymbol{w}_-$ and $\boldsymbol{v}_{\mathrm{dd}}$ are therefore constrained to be $\mathbf{0}$.

6.4.1 Training

The overall goal of the model above is to provide labelled partitions of unseen data. Before this task can be performed however, it is necessary to estimate the model parameters, $\boldsymbol{\theta}$. These parameters are learned from example data. Given a labelled training example,$(\boldsymbol{x}, \mathcal{Y})$, the posterior probability of the parameters is given by Bayes' rule,

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}, \mathcal{Y}) \propto P(\mathcal{Y} \mid \boldsymbol{x}, \boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}), \tag{6.16}$$

where $P(\boldsymbol{\theta})$ is a prior distribution over the weights. The model is trained by finding the maximum *a posteriori* (MAP) weights using a quasi-Newton gradient ascent algorithm (specifically, the BFGS implementation contained in the MATLAB statistics toolbox). A significant advantage is that the model is convex in the parameters, meaning that a gradient ascent algorithm will always find the global maximum. Substituting the details of the distribution into (6.16) gives a log posterior, $\mathcal{L}$, of

$$\mathcal{L} = \sum_i \log \psi_i^{(1)} + \sum_{ij} \log \psi_{ij}^{(2)} - \log Z + \log\left(\mathrm{P}(\boldsymbol{\theta})\right) \tag{6.17}$$

Taking the gradient with respect to a parameter θ_k therefore gives

$$\frac{\partial}{\partial \theta_k}\mathcal{L} = \sum_{i \in \mathcal{V}} \frac{\partial}{\partial \theta_k} \log \psi_i^{(1)} + \sum_{ij \in \mathcal{E}} \frac{\partial}{\partial \theta_k} \log \psi_{ij}^{(2)} - \frac{1}{Z}\frac{\partial Z}{\partial \theta_k} + \frac{\partial}{\partial \theta_k} \log\left(\mathrm{P}(\boldsymbol{\theta})\right) \tag{6.18}$$

The partition function, Z, is given by

$$Z(\boldsymbol{\theta}) = \sum_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}) \prod_{i,j \in \mathcal{E}} \psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}). \tag{6.19}$$

So, if the θ_k is a parameter of the unary potentials, the gradient is given by

$$\begin{aligned}
\frac{1}{Z}\frac{\partial Z}{\partial \theta_k} &= \frac{1}{Z}\frac{\partial}{\partial \theta_k} \sum_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)} \prod_{ij \in \mathcal{E}} \psi_{ij}^{(2)} && (6.20)\\
&= \frac{1}{Z} \sum_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)} \prod_{ij \in \mathcal{E}} \psi_{ij}^{(2)} \sum_{m \in \mathcal{V}} \frac{1}{\psi_m^{(1)}} \frac{\partial}{\partial \theta_k} \psi_m^{(1)} && (6.21)\\
&= \sum_{m \in \mathcal{V}} \sum_{\mathcal{Y}} P(\mathcal{Y}) \frac{\partial}{\partial \theta_k} \log \psi_m^{(1)} && (6.22)\\
&= \sum_{m \in \mathcal{V}} \left\langle \frac{\partial}{\partial \theta_k} \log \psi_m^{(1)} \right\rangle && (6.23)
\end{aligned}$$

where explicit functional dependencies have been dropped for clarity. The brackets, $\langle \cdots \rangle$, represent expectation with respect to the distribution over $\mathbb{Y}$ given by the current parameter

values. Combining (6.23) with (6.18) gives

$$\frac{\partial}{\partial\theta_k}\mathcal{L} = \sum_{i\in\mathcal{V}}\left(\frac{\partial}{\partial\theta_k}\log\psi_i^{(1)} - \left\langle\frac{\partial}{\partial\theta_k}\log\psi_i^{(1)}\right\rangle\right) + \frac{\partial}{\partial\theta_k}\log\left(\mathrm{P}\left(\boldsymbol{\theta}\right)\right) \tag{6.24}$$

For the particular case of exponential non-linearities, this expression simply reduces to

$$\frac{\partial}{\partial\theta_k}\mathcal{L} = \sum_{i\in\mathcal{V}}\left(g_{ik}\delta_{y_{t_i},y_k} - \left\langle g_{ik}\delta_{y_{t_i},y_k}\right\rangle\right) + \frac{\partial}{\partial\theta_k}\log\left(\mathrm{P}\left(\boldsymbol{\theta}\right)\right) \tag{6.25}$$

where δ_{y_i,y_k} takes the value 1 if y_k, the label associated with weight k is equal to y_{t_i}, the actual label assigned to vertex i, and zero otherwise. The gradients with respect to parameters of the pairwise potentials have a similar form. If multiple training examples are provided then optimisation is performed on the joint probability of all of the data. As the algorithm works in log space, optimising the joint probability simply results in a summation over all training examples,

$$\frac{\partial}{\partial\theta_k}\mathcal{L} = \sum_j\sum_{i\in\mathcal{V}}\left(g_{ik}\delta_{y_i^{(j)},y_k} - \left\langle g_{ik}\delta_{y_i,y_k}\right\rangle\right) + \frac{\partial}{\partial\theta_k}\log\left(\mathrm{P}\left(\boldsymbol{\theta}\right)\right) \tag{6.26}$$

where the superscript (j) indicates the value in the j^{th} training example. This expression is closely related to the training rule for Boltzmann machines with hidden units (see Chapter 43 in [56] for further details).

The expectation in (6.26) requires the computation of marginal probability distributions for individual vertices and pairs of vertices connected by an edge. Furthermore, the optimisation algorithm needs to evaluate (6.13) explicitly, which in turn requires evaluation of the partition function. Both of these tasks involve summations over subsets of possible labelled partitions. This summation can be performed efficiently by message passing using a modified version of the sum-product algorithm. The details of this algorithm will be given in Section 6.5.

6.4.2 Inference

Having trained the model on example data, we next usually use the model to assign a labelled partition to previously unseen data. The approach taken is to return the most probable configuration,

$$\mathcal{Y}^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}}\prod_{i\in\mathcal{V}}\psi_i^{(1)}\left(\mathcal{Y},\boldsymbol{x};\boldsymbol{\theta}\right)\prod_{i,j\in\mathcal{E}}\psi_{ij}^{(2)}\left(\mathcal{Y},\boldsymbol{x};\boldsymbol{\theta}\right). \tag{6.27}$$

This process is similar to the computation required to find the partition function, Z, and can be performed using the max-product in much the same way. This algorithm will also be described in Section 6.5.

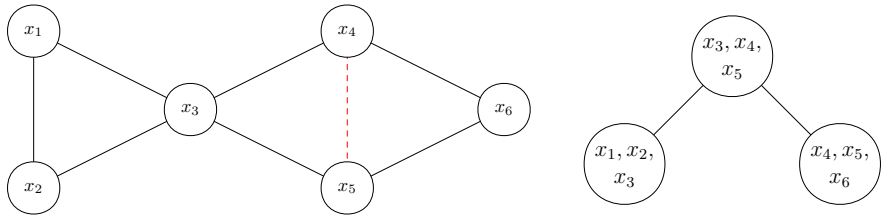


Figure 6.6: An example of a undirected graph (left) and the corresponding junction tree (right). The original graph, shown in black, has a cycle of length four, so an additional edge, shown in red, must be added to perform triangulation.

6.5 Operations over labelled partitions

6.5.1 Message passing

Efficient inference in undirected graphical models can be performed systematically by message passing algorithms. These algorithms reduce the task to local computations involving subsets of the vertices, the results of which are passed as messages along the edges in the graph.

For exact inference two such algorithms are widely used: the sum-product algorithm which is used for marginalisation over subsets of the variables, and the max-product algorithm which is used to find the most probable configurations of the variables. Equivalently, the max-sum algorithm performs the same role when logarithms of the probabilities are available. As an illustration, consider the sum-product algorithm.

The goal of this algorithm is to compute the sum over a subset of the variables, resulting in the marginal distribution of those which remain. The first stage of the algorithm is to *triangulate* the graph, adding additional edges so that there are no loops of size four or greater which are not spanned by a chord. This process has no effect on the distribution, as the potentials associated with the added edges simply evaluate to one for all configurations, but is necessary to ensure that the algorithm terminates.

Having triangulated the graph, a *junction tree* is constructed. The junction tree is a second graph containing one node for each maximal clique in the original graph (a clique is a maximally connected subgraph). The vertices on the junction tree are connected with edges so that they possess the *running intersection property*. This requirement is satisfied if for all pairs of vertices on the graph corresponding to cliques sharing a common vertex, there exists at least one path between them such that all vertices on the path also correspond to cliques containing the common vertex. As long as the original graph is triangulated, a junction graph with this property always exists. Further more, the junction graph will always be a tree. However, it may not be unique, and selection between the possible trees may affect the computational requirements of the algorithm. An example of a graph and the associated junction tree is shown in Figure 6.6.

Consider the sum over all of the vertices in the graph, which can be written as

$$S = \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6}\psi_{123}(x_1,x_2,x_3)\,\psi_{345}(x_3,x_4,x_5)\,\psi_{456}(x_4,x_5,x_6) \quad (6.28)$$

$$= \sum_{x_1}\sum_{x_2}\sum_{x_3}\psi_{123}(x_1,x_2,x_3)\sum_{x_4}\sum_{x_5}\psi_{345}(x_3,x_4,x_5)\sum_{x_6}\psi_{456}(x_4,x_5,x_6) \quad (6.29)$$

The summation over the third potential is clearly independent of x_1, x_2 and x_3. It is therefore not necessary to repeat this computation for all values of these variables. Instead, it can be pre-computed and stored as a function of x_4 and x_5. Conceptually, the result of this computation can be regarded as a message passed from clique (456) to clique (345).

To express the message passing framework more formally, let C_i represent the set of variables present in the i^{th} clique, which has potential ψ_i. Let S_{ij} be the separator between cliques i and j: the set of all variables appearing in both C_i and C_j. The message passed from i to j is then

$$\mu_{i\to j}(S_{ij}) = \sum_{C_i\setminus S_{ij}}\psi_i(C_i)\prod_{k\in\mathcal{N}(i)\setminus j}\mu_{k\to i}(S_{ik}) \quad (6.30)$$

in which $\mathcal{N}(i)$ is the neighbourhood of clique i on the junction graph. Assume that the set of variables over which summation is not to be performed is entirely contained within a single clique. This clique can be designated as the root of the junction tree, and messages are passed from the leaves of the tree to the root. Each node (excluding the root) can pass a message as soon as it has received incoming messages from all but one of its neighbours, with the outgoing message being passed to that which remains.

A new potential over the root clique, C_1, can now be defined as

$$\psi_1'(C_1) = \psi_1(C_1)\prod_{k\in\mathcal{N}(1)}\mu_{k\to 1}(C_1) \quad (6.31)$$

which is the un-normalised marginal distribution over configurations of that clique. Partial or full summation over these configurations can then be performed explicitly as required.

If marginal distributions over all cliques are required then performing the full message passing routine each time would result in wasteful duplication of calculations. A little thought reveals that a second, outward pass from the root back to the leaves will compute messages passed in both directions along all edges on the junction tree. The resulting messages are sufficient to compute all marginals at once.

The limiting factor in the computation is the size of the summations which must be performed. The largest summation will result from computing the outgoing messages from the largest clique, and involves L^C calculations, where C is the size of that clique and L is the number of values that each variable can take.

6.5.2 Message passing for labelled partitions

Message passing for traditional undirected graphical models is well established, but it cannot be applied directly to the model over labelled partitions; this can be understood intuitively as a result of the fact that a partition of the graph cannot be broken down into components localised to individual vertices. Fortunately, equivalent algorithms can be found, a formal derivation of which is presented below. The derivation presented here follows the conditions for the possibility of local computation provided by Shenoy and Shafer [87].

If G is a subset of $\mathcal{V}$, let $\mathcal{Y}_G \in \mathbb{Y}_G$ denote a labelled partition of the corresponding induced subgraph. *Consistency* is then defined as follows:

Definition 2. *Labelled partitions $\mathcal{Y}_G$ and $\mathcal{Y}_H$, of subgraphs G and H respectively, are **consistent**, denoted $\mathcal{Y}_H \backsim \mathcal{Y}_G$, if and only if:*

1. *All vertices appearing in $G \cap H$ are assigned the same label by $\mathcal{Y}_G$ and $\mathcal{Y}_H$, and*
2. *All pairs of vertices appearing in $G \cap H$ are in the same part in $\mathcal{Y}_G$ if and only if they are in the same part in $\mathcal{Y}_H$.*

The notation $\hat{\mathcal{Y}}_G(\mathcal{Y}_{G\cup H})$ is used to denote the unique labelled partition of G which is consistent with $\mathcal{Y}_{G\cup H}$. The maximal cliques of $\mathcal{G}$ are defined in the usual way, and are denoted $C_1, \ldots, C_N$. If b and t are two cliques, and b contains all vertices from t which appear in cliques other than t, then b is said to be a *branch* and t is the corresponding *twig*. For example, in Figure 6.6, the node (x_4, x_5, x_6) is a twig, and the node (x_3, x_4, x_5) is the corresponding branch.

Let ψ be a *valuation* on a subset of $\mathcal{V}$. In the case of standard belief propagation, valuations are functions assigning a real, non-negative value to possible configurations of subsets of the variables. Here however, a valuation on a subset G will be defined as a function mapping $\mathbb{Y}_G$ to the non-negative real numbers. $\mathbb{V}_G$ is the set of all valuations on G. In the case where the valuation is over the whole of $\mathcal{G}$, the range of the valuation will be interpreted as being proportional to the probability of the corresponding labelled partition. In the case of valuations defined over subsets of $\mathcal{V}$ the valuations are referred to as potentials of which those defined in (6.13) are an example. Two operations on valuations must be defined:

1. **Combination:** Suppose G and H are subsets of $\mathcal{V}$ and ψ_G and ψ_H are valuations on those subsets. The operation of combination defines a mapping $\otimes : \mathbb{V}_G \times \mathbb{V}_H \mapsto \mathbb{V}_{G\cup H}$, such that
$$\psi_G \otimes \psi_H(\mathcal{Y}_{G\cup H}) \triangleq \psi_G\left(\hat{\mathcal{Y}}_G(\mathcal{Y}_{G\cup H})\right) \cdot \psi_H\left(\hat{\mathcal{Y}}_H(\mathcal{Y}_{G\cup H})\right). \tag{6.32}$$
2. **Marginalisation:** Suppose G and H are subsets of $\mathcal{V}$ such that $G \subseteq H$, and ψ_G and ψ_H are valuations as before. Marginalisation is a mapping $\downarrow : \mathbb{V}_H \mapsto \mathbb{V}_G$ such that
$$\psi_H^{\downarrow G}(\mathcal{Y}_G) \triangleq \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_G} \psi_H(\mathcal{Y}_H). \tag{6.33}$$

A valuation over the whole graph is said to factor if it can be written as the combination of valuations on the cliques,

$$\psi\left(\mathcal{Y}\right) = \bigotimes_{i=1}^{N} \psi_i\left(\mathcal{Y}_i\right), \tag{6.34}$$

where i runs over the cliques in $\mathcal{G}$. As combination allows products of valuations over subsets of a clique to be written in terms of a single valuation over the whole clique, the model given in (6.13), excluding the partition function, is in this form. In order to proceed to develop efficient algorithms, it is first necessary to demonstrate that three axioms of Shenoy and Shafer are satisfied:

Axiom 1. Commutativity and associativity of combination. *If G, H and K are subsets of $\mathcal{V}$, for any valuations ψ_G, ψ_H and ψ_K, we have $\psi_G \otimes \psi_H = \psi_H \otimes \psi_G$ and $\psi_G \otimes (\psi_H \otimes \psi_K) = (\psi_G \otimes \psi_H) \otimes \psi_K$.*

Proof. Follows directly from the definition of combination. □

Axiom 2. Consonance of marginalisation, *If G, H and K are subsets of $\mathcal{V}$ such that $K \subseteq G \subseteq H$, for any valuations ψ_G, ψ_H and ψ_K,*

$$\left(\psi_H^{\downarrow G}\right)^{\downarrow K} = \psi_H^{\downarrow K}. \tag{6.35}$$

Proof. Writing the marginalisation explicitly,

$$\begin{aligned}\left(\psi_H^{\downarrow G}\right)^{\downarrow K} &= \sum_{\mathcal{Y}_G \backsim \mathcal{Y}_K} \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_G} \psi_H\left(\mathcal{Y}_H\right) \\ &= \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_K} \psi_H\left(\mathcal{Y}_H\right) = \psi_H^{\downarrow K},\end{aligned} \tag{6.36}$$

where the second line follows as for any $\mathcal{Y}_H \backsim \mathcal{Y}_K$ there is a unique $\mathcal{Y}_G$ such that $\mathcal{Y}_G \backsim \mathcal{Y}_K$ and $\mathcal{Y}_H \backsim \mathcal{Y}_G$, and for any $\mathcal{Y}_H \not\backsim \mathcal{Y}_K$, no such $\mathcal{Y}_G$ exists. □

Axiom 3. Distributivity of marginalisation over combination, *If G and H are subsets of $\mathcal{V}$, for any valuations ψ_G and ψ_H, $(\psi_G \otimes \psi_H)^{\downarrow G} = \psi_G \otimes (\psi_H^{\downarrow G \cap H})$.*

Proof. Performing an explicit expansion gives

$$\begin{aligned}(\psi_G \otimes \psi_H)^{\downarrow G} &= \sum_{\mathcal{Y}_{G\cup H} \backsim \mathcal{Y}_G} \psi_G\left(\hat{\mathcal{Y}}_G\left(\mathcal{Y}_{G\cup H}\right)\right) \cdot \\ &\qquad \psi_H\left(\hat{\mathcal{Y}}_H\left(\mathcal{Y}_{G\cup H}\right)\right) \\ &= \psi_G\left(\mathcal{Y}_G\right) \cdot \sum_{\mathcal{Y}_{G\cup H} \backsim \mathcal{Y}_G} \psi_H\left(\hat{\mathcal{Y}}_H\left(\mathcal{Y}_{G\cup H}\right)\right) \\ &= \psi_G\left(\mathcal{Y}_G\right) \cdot \sum_{\mathcal{Y}_H \backsim \hat{\mathcal{Y}}_{G\cap H}(\mathcal{Y}_G)} \psi_H\left(\mathcal{Y}_H\right),\end{aligned} \tag{6.37}$$

which is equal to $\psi_G \otimes (\psi_H^{\downarrow G\cap H})$ by definition. □

6.5.3 Sum-product algorithm

Having shown that Shenoy and Shaffer's axioms are satisfied, the next two sections formally develop an extension of the sum-product algorithm suitable for probability distributions over partitions. As with the more usual form of this algorithm, the resulting method exploits the known structure of $\mathcal{G}$ by passing messages containing the results of local computations. The goal of the algorithm is to compute sums over a subset of all possible partitions, such as those needed for the partition function, as given in (6.19). This task should be contrasted with that of the usual sum-product algorithm [51], which sums over assignments of labels to the vertices. Since the summation is over a different domain it will be necessary to modify the messages passed and the ranges of summation. Later, in Section 6.5.5, the max-product algorithm for labelled partitions will be considered.

Suppose the cliques of $\mathcal{G}$ are numbered $C_1, \ldots, C_N$, such that C_1 is a clique whose marginal distribution we wish to compute. In other words, we wish to find the sum:

$$f_s(\mathcal{Y}_1) = \sum_{\mathcal{Y} \backsim \mathcal{Y}_1} P^*(\mathcal{Y}) = (P^*(\mathcal{Y}))^{\downarrow C_1}, \tag{6.38}$$

where $\mathcal{Y}_1$ represents a local labeled partition of clique C_1 and P^* is a (possibly unnormalised) probability distribution over labelled partitions of $\mathcal{G}$. Furthermore, suppose that the numbering of the cliques is such that for all k, C_k is a twig in the graph $C_1 \cup C_2 \cup \ldots \cup C_k$. Such an ordering is always possible if $\mathcal{G}$ is triangulated. According to Axiom 2, this can be expressed as

$$\begin{aligned} f_s(\mathcal{Y}_1) &= \left((P^*(\mathcal{Y}))^{\downarrow \mathcal{V}\setminus C_N} \right)^{\downarrow C_1} \\ &= \left(\left(\bigotimes_{i=1}^{N} \psi_i(\mathcal{Y}_i) \right)^{\downarrow \mathcal{V}\setminus C_N} \right)^{\downarrow C_1} \\ &= \left(\left(\bigotimes_{i=1}^{N-1} \psi_i(\mathcal{Y}_i) \right) \otimes \left(\psi_N(\mathcal{Y}_N)^{\downarrow C_N \cap \mathcal{V}} \right) \right)^{\downarrow C_1}. \end{aligned} \tag{6.39}$$

In the last step, Axiom 3 has been used. C_N is a twig by construction. Let C_B be a corresponding branch, then $C_N \cap \mathcal{V} = C_N \cap C_B$, hence

$$f_s(\mathcal{Y}_1) = \left(\left(\bigotimes_{\substack{i=1 \\ i\neq B}}^{N-1} \psi_i(\mathcal{Y}_i) \right) \otimes \psi_B(\mathcal{Y}_B) \otimes \left(\psi_N(\mathcal{Y}_N)^{\downarrow C_N \cap C_B} \right) \right)^{\downarrow C_1}. \tag{6.40}$$

In other words, the problem can be converted to an equivalent marginalisation over a graph with one less clique in which the potential for C_B has been replaced according to:

$$\psi_B \leftarrow \psi_B \otimes \left(\psi_N^{\downarrow C_N \cap C_B} \right). \tag{6.41}$$

By repeatedly eliminating cliques in this way we can systematically remove cliques until only C_1 remains. Any further summation which is required (either to give marginals over a smaller subset of vertices, or to calculate the partition function) can be performed explicitly.

6.5.4 Message passing

The result of the elimination illustrated in (6.41) can be interpreted in terms of a message passed from C_N to the rest of the graph. Messages are passed between cliques along edges in a *junction tree* equivalent to that described in Section 6.5.1. Let $\mu_{i\to j}(\mathcal{Y}_j)$ be the message passed from C_i to C_j. The form of the message is a list of labelled partitions of the intersection $C_i \cap C_j$, each of which has an associated scalar value. The messages are updated iteratively according to the rule:

$$\mu_{i\to j}(\mathcal{Y}_j) \leftarrow \sum_{\mathcal{Y}_i \backsim \mathcal{Y}_j} \psi_i(\mathcal{Y}_i) \prod_{\substack{k\in\mathcal{N}(i)\\ k\neq j}} \mu_{k\to i}(\mathcal{Y}_i), \tag{6.42}$$

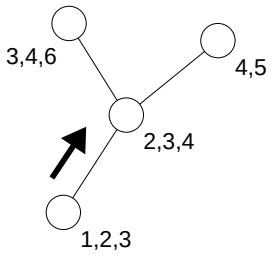
with the outgoing messages from a clique being updated once all incoming messages from the other neighbouring cliques $\mathcal{N}(\cdot)$ have been received. Note that the domain of the incoming messages is only a subset of the vertices in clique i, so the messages must be extended to fully cover $\mathcal{Y}_i$. This is done by simply assigning to each configuration of the clique the value of the unique consistent configuration of the separator. As the junction tree has no cycles, this process will terminate after a finite number of iterations. Having updated all of the messages, it is then possible to find f_s using

$$f_s(\mathcal{Y}_1) = \psi_1(\mathcal{Y}_1) \prod_{k\in\mathcal{N}(1)} \mu_{k\to 1}(\mathcal{Y}_1). \tag{6.43}$$

Having defined the algorithm formally, it is useful to also give an intuitive interpretation. The message passed from C_i to C_j can be interpreted as a statement summarising the values of the 'upstream' potentials for labelled partitions which are consistent with each labelled partition of the separator between C_i and C_j. See Figure 6.7 for an example of the message passing process. As is the case with the usual form of the sum-product algorithm, the same messages are used in computing different marginals. Marginal distributions for all cliques can be found simultaneously with a single bidirectional pass of the message update rule.

6.5.5 Max-product algorithm

Just as is the case for the usual form of the sum-product algorithm, it is possible to replace the summation in (6.38) with a maximisation to obtain the max-product algorithm. This replacement is equivalent to a redefinition of marginalisation to represent the maximum valuation consistent with the sub-partition rather than the sum over all valuations. This algorithm is used to compute maximisations, for example the configuration of C_1 in the most probable



(a)

Potential	Value
$\psi_{ij}\,(t_i = t_j, y_i = y_j)$	0.6
$\psi_{ij}\,(t_i \neq t_j, y_i = y_j)$	0.4
$\psi_{ij}\,(t_i \neq t_j, y_i \neq y_j)$	0.2

(b)

Partition	Label	Value
(123)	+	0.216
(123)	-	0.216
(12)(3)	+, +	0.096
(12)(3)	+, -	0.024
⋮	⋮	⋮
(1)(2)(3)	-,-,-	0.064

(c)

Partition	Label	Value
(23)	+	0.336
(23)	-	0.336
⋮	⋮	⋮
(2)(3)	-,-	0.272

(d)

Figure 6.7: An example of message passing. (a) The junction tree corresponding to $\mathcal{G}$. (b) The potentials, in this case uniform and independent of data for clarity. (c) The clique potential for the clique consisting of vertices 1, 2 and 3. (d) The message passed from (123) to (234), concerning labelled partitions of vertices 2 and 3.

labelled partition,

$$\mathcal{Y}_1^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}_1} \max_{\mathcal{Y} \backsim \mathcal{Y}_1} P^*\left(\mathcal{Y}\right). \tag{6.44}$$

In the context of probabilistic inference, such a maximisation is necessary when searching for the most probable configuration. Message passing is done in the same way as described above, with a modified message update rule.

$$\mu_{i \to j}\left(\mathcal{Y}_j\right) \leftarrow \max_{\mathcal{Y}_i \backsim \mathcal{Y}_j} \psi_i\left(\mathcal{Y}_i\right) \prod_{\substack{k \in \mathcal{N}(i) \\ k \neq j}} \mu_{k \to i}\left(\mathcal{Y}_i\right). \tag{6.45}$$

Having updated all of the messages, $\mathcal{Y}_1^{\mathrm{MAX}}$ can be found using

$$\mathcal{Y}_1^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}_1} \psi_1\left(\mathcal{Y}_1\right) \prod_{k \in \mathcal{N}(1)} \mu_{k \to 1}\left(\mathcal{Y}_1\right). \tag{6.46}$$

To find the global maximum configuration, this method can be repeated for all possible roots, and the global partition can be reconstructed as the union of the local configurations (which will be consistent with one another).

Again, it is instructive to consider the intuitive meaning of the messages. In this case they can be interpreted as statements about the maximum value that can be achieved 'upstream' as a function of the clique separator configuration. When the next cluster computes its maximum configuration, the contribution of downstream potentials can therefore be incorporated from the messages rather than having to be recomputed from scratch each time.

6.6 Complexity and the edge-dual representation

Section 6.3.2 presented an alternative approach to labelled partitions which was shown to be too demanding in terms of computational requirements for real-life applications. Yet another alternative representation which avoids these problems is to use indicator variables, $\check{\boldsymbol{x}}\left(\mathcal{Y}\right)$,

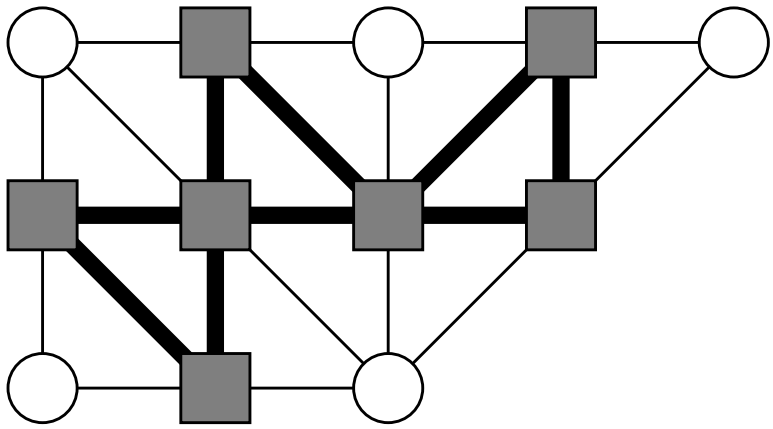


Figure 6.8: An example of an undirected graph (circular vertices and light lines) and the corresponding edge-dual graph (square vertices and heavy lines).

for each edge in $\mathcal{G}$. For binary labels, these variables are over six possible values: two states corresponding to segments belonging to the same part with each label, and four corresponding to different parts with all four combinations of labels. To construct a graphical model for these variables, we define the *edge-dual graph*:

Definition 3. *For any graph $\mathcal{G}$, the **edge-dual graph**, $\check{\mathcal{G}} = \left(\check{\mathcal{V}}, \check{\mathcal{E}}\right)$ contains one vertex for each edge in $\mathcal{G}$. Vertices in $\check{\mathcal{G}}$ are connected by an edge if and only if all vertices connected to their corresponding edges in $\mathcal{G}$ belong to the same clique.*

An example of an edge-dual graph is shown in Figure 6.8. Every labelled partition of $\mathcal{G}$ corresponds to a unique configuration of the edge-dual vertices, but there are configurations of the edge-dual vertices which do not correspond to labelled partitions. Hence,

Definition 4. *A configuration of the edge-dual vertices is **valid** if and only if it corresponds to a labelled partition of $\mathcal{G}$.*

Invalid configurations arise when pairwise constraints yield contradictory information; following one path between two vertices on $\mathcal{G}$ indicates that they are in the same part, whereas another path indicates that they are not, or their labels disagree. It is possible to establish the validity of a configuration using only calculations local to cliques on $\check{\mathcal{G}}$.

Suppose $P^*\left(\check{\boldsymbol{x}}\left(\mathcal{Y}\right)\right)$ is a probability distribution over labelled partitions of $\mathcal{G}$ as represented by the edge-dual variables. As before, consider the task of summing P^* over all partitions in order to find the partition function. Rather than expressing the summation in terms of partitions, we can work directly with $\check{\boldsymbol{x}}$, provided that the summation is limited to those configurations which are valid. This constraint can be achieved by introducing an indicator function, $\mathbb{I}\left(\check{\boldsymbol{x}}\right)$, which takes the value 1 if $\check{\boldsymbol{x}}$ is valid and 0 otherwise,

$$\sum_{\mathcal{Y}} P^*\left(\check{\boldsymbol{x}}\left(\mathcal{Y}\right)\right) = \sum_{\check{\boldsymbol{x}}} \mathbb{I}\left(\check{\boldsymbol{x}}\right) \cdot P^*\left(\check{\boldsymbol{x}}\right). \tag{6.47}$$

	Clique Size					
	2	3	4	5	6	n
(a)	2	5	15	52	203	Bell no. B_n
(b)	*6*	*22*	*94*	*454*	*2430*	A001861 [100]
(c)	6	216	46656	6.0×10^7	4.7×10^{11}	$6^{n(n-1)/2}$
(d)	16	216	4096	1.0×10^5	3.0×10^6	$(2n)^n$

Table 6.1: Sizes of the configuration tables for each of the methods. (a) Unlabelled Partitions (these are the Bell numbers). (b) Binary labelled partitions (c) Binary labelled edge-dual representation. (d) Binary labelled part IDs (lower bound).

There is a one-to-one correspondence between cliques in $\mathcal{G}$ and $\check{\mathcal{G}}$, so functions which factor according to $\mathcal{G}$ also factor according to $\check{\mathcal{G}}$. If P^* factors, we can write

$$\sum_{\mathcal{Y}} P^* \left(\check{\boldsymbol{x}} \left(\mathcal{Y} \right) \right) = \sum_{\check{\boldsymbol{x}}} \left(\prod_i \mathbb{I}_i \left(\check{\boldsymbol{x}}_i \right) \cdot \check{\psi}_i \left(\check{\boldsymbol{x}}_i \right) \right), \tag{6.48}$$

where i ranges over the cliques of $\check{\mathcal{G}}$. In (6.48), the local nature of $\mathbb{I}$ has been used to factor it as well as P^*. The result is a sum over a function which factors according to $\check{\mathcal{G}}$, so it can be found using the standard sum-product algorithm.

As there is a one-to-one correspondence between valid edge-dual configurations, and labelled partitions of $\mathcal{G}$, this algorithm is in many respects equivalent to that presented in Section 6.5.3. However, in two important respects it is less efficient. Firstly, as the sum includes edge-dual configurations which are invalid, the number of terms in the sum is significantly greater. Secondly, it is necessary to determine the validity of the current configuration for each term, which introduces additional overhead. The algorithm presented in Section 6.5.3 may be regarded as an efficient implementation of this algorithm, where the validity of configurations is pre-computed, and only those which are valid are included in the sum.

6.6.1 Complexity

The dominant factor in the complexity of the message passing algorithm is the time taken to process all possible partitions of the largest clique. Table 6.1 lists the number of possible configurations for the various cases, which are shown graphically in Figure 6.9. It can be seen from the table that the method described in Section 6.5 offers a considerable improvement in the complexity of the calculations.

6.7 Implementation details

Having derived a probabilistic model over labelled partitions of an undirected graph, and shown that efficient inference is possible, the remainder of this section returns to the intended application of electronic ink analysis.

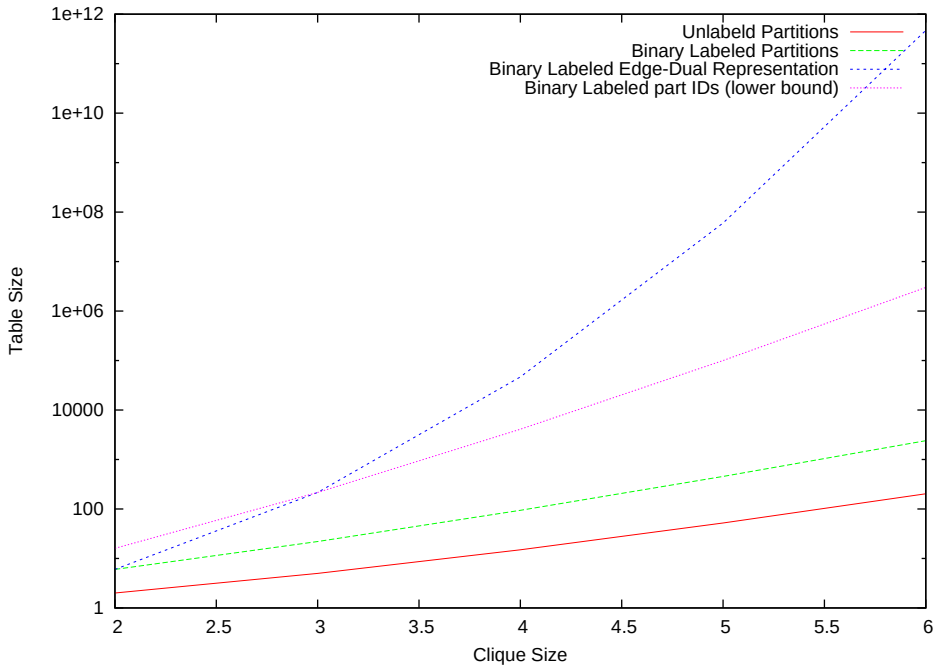


Figure 6.9: Sizes of the message tables for each of the methods.

6.7.1 Graph construction

The graph which is partitioned, $\mathcal{G}$, is constructed by first building a candidate graph by connecting all pairs of fragments which are closer than a preset threshold distance. This graph is not necessarily triangulated, so additional edges are added as necessary. Pairwise feature vectors are generated for all edges on the new graph, including those which were added during triangulation, as there is no significant computational cost involved in doing so, and these features might provide useful information. In fact, the whole process can be viewed as an approximation to the fully connected graph, so these additional features actually improve the faithfulness of the approximation.

6.7.2 Feature set

We chose features to reflect the spatial and temporal distribution of ink strokes, for example lengths and angles of fragments, whether two fragments were drawn with a single stroke, and the temporal ordering of strokes. In addition, a number of 'template' features are used, which were designed to capture important higher level aspects of the ink, such as the presence of T-junctions.

As some of the features correspond to real-valued quantities such as lengths and angles, some care needs to be taken. One approach would be to use the quantity directly. However, features of this kind would enforce a linear relationship between the value of the quantity and the strength of the weight, which may not be appropriate. Instead, we divided the range of

Name	Feature Count	Type
Length	11	Histogram
Angle	9	Histogram
Neighbour Angle	9	Histogram
Neighbour Relative Angle	11	Histogram
Neighbour Distance	11	Histogram
Long Fragment	1	Indicator
Full Box	1	Template
T-Junction Count	2	Count
Neighbour Full Box	1	Template
Right/Bottom	1	Template
Box	3	Template
Bias	1	Constant

Table 6.2: Unary features

Name	Feature Count	Type
Angle	11	Histogram
Distance	11	Histogram
Temporal Ordering	6	Histogram
T-Junction	1	Template
Box Full	1	Template
Right	1	Template
Bottom	1	Template
neighbour_two_box	1	Template
Similar Direction	1	template
Aligned	1	Template
Right Angle	1	Template
Bias	1	Constant

Table 6.3: Pairwise features

the feature value into subintervals, each of which was associated with a binary valued feature. Collectively, these groups are referred to as 'histogram features'.

We used a total of 61 unary features and 37 pairwise features were used, including bias features which take the value 1 for all potentials and provide a way of adjusting the overall preference for labels as well as the 'natural' degree of segmentation. Descriptions of the features are given in Tables 6.2 and 6.3.

6.7.3 Priors

We used Gaussian priors for all of the weights. In the case of histogram features the Gaussian was correlated so that the weights of intervals corresponding to similar bins were encouraged

to take on similar values. More formally, the covariance matrix, S had the form

$$S_{ij} = w_s \exp\left(-\frac{(i-j)^2}{w_r}\right) \tag{6.49}$$

which may be viewed as an approximation to a continuous mapping from real valued features to weights, with a Gaussian process prior. For the experiments described below, we used values of 2.0 for w_s and w_r, as well as for the variance on the independent priors for the non-histogram features.

6.8 Experimental evaluation

To test the performance of the method we used a database of 40 example diagrams, consisting of a total of 2157 ink fragments. We generated three random splits, each consisting of 20 examples used for training and 20 used for evaluation. We trained the model by finding the MAP weights as described in Section 6.4.1. The model was then tested by finding the most probable partition and labelling as described in Section 6.4.2, and counting errors made against ground-truth data. The undirected graphs constructed for the example diagrams had a mean tree-width of 4.0.

For comparison, we also evaluated two related models. These models performed labelling only, without considering partitioning, and both are implemented as standard CRFs. The first of these models includes unary potentials which are dependent on the class of the vertex, and uses pairwise potentials given by

$$\psi_{ij}^{(2)}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi\left(\boldsymbol{v}_{\mathrm{s}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } y_i = y_j \\ \phi\left(\boldsymbol{v}_{\mathrm{d}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } y_i \neq y_j \end{cases}, \tag{6.50}$$

where $\boldsymbol{v}_{\mathrm{s}}$ and $\boldsymbol{v}_{\mathrm{d}}$ are weights corresponding to vertices i and j having the same and different labels respectively. The second related model does not use pairwise potentials at all — ink fragments are labelled independently of the other labellings. In the following, the full model performing labelling and partitioning is referred to as model PLI. LI is the model performing labelling only with pairwise potentials, and L is the model with unary potentials only.

6.8.1 Learned weights

Figures 6.10 and 6.11 show the weights which were learned for each of the features. The majority of the features play some role in discriminating between labelled partitions, with a few features having particularly large weights. A number of the template features come near the top of the lists, particularly the T-junction template. This result illustrates the power of the CRF framework in allowing such arbitrary dependence on the observed data. Also note that one of the temporal features plays an important role in deciding whether fragments are

Model	Split			Mean
	1	2	3	
L	8.2%	9.3%	7.9%	$8.5\% \pm 0.4\%$
LI	4.6%	6.8%	1.9%	$4.4\% \pm 1.4\%$
% Δ LI/L	−44%	−27%	−76%	$-49\% \pm 14\%$
PLI	3.1%	3.8%	0.8%	$2.6\% \pm 0.9\%$
% Δ PLI/LI	−33%	−44%	−58%	$-45\% \pm 7\%$

Table 6.4: Labelling errors for the three models. Results are the mean of three cross-validation splits. Relative differences are shown between models L and LI, and between LI and PLI. The mean relative differences are aggregations of the differences for each split, rather than the differences between the means for individual models in order to reduce the effect of systematic variation between splits.

in the same object or not, indicating that the additional information available as a result of working directly with digitised ink strokes is valuable.

The need for histogram features can be seen by considering the length features as an example of the learned weights, shown in Figure 6.12. While intermediate lengths provide evidence for a container, both long and short fragments are likely to belong to connectors (the short fragments resulting from the decomposition of curves into short line fragments). This relationship could not be achieved using the implied linear mapping produced by using the feature value directly.

6.8.2 Error rates

Labelling error rates were measured as the fraction of the fragments which were incorrectly labelled, and are shown in Table 6.4 and Figure 6.13.

Evaluation of the performance in partitioning presented a more significant challenge, partially as there is not a standard metric for this measurement, and partially because of the lack of a comparable algorithm with which to compare the results. The metric which was chosen is a modified version of the error metric used by Martin *et. al.* [58],

$$E = \sum_i \frac{|R(S_1, i) \setminus R(S_2, i)| + |R(S_2, i) \setminus R(S_1, i)|}{|R(S_1, i)|} \tag{6.51}$$

where $R(S, i)$ is the set of vertices in the same part as vertex i in partition S. S_1 is the ground truth partition and S_2 is the partition returned by the algorithm. The mean grouping error using this metric was 36. Figure 6.14 shows the output of the algorithm on an example diagram.

6.9 Discussion

The results given in Section 6.8 show that the present approach is capable of providing high-quality labelled partitions. The data also illustrate an important point: simultaneous labelling and partitioning produces a significant improvement in labelling performance. This is easily understandable — the constraint that vertices within the same part must be labelled identically provides strong evidence for the labelling part of the algorithm, and the boundaries between regions of different labels are strong candidates for part boundaries. Hence the two aspects of the algorithm reinforce each other.

6.10 Future extensions

6.10.1 Additional features

There are a number of additional aspects of the data which cannot easily be taken into account using the method described above. In particular, properties of individual parts essentially produce features which depend on the partition itself. Examples of such features include information containing the relative closure of the ink strokes making up a part, or the convexity defined as the ratio of the area of the part to its convex hull. These features are likely to be highly informative in the application to electronic diagram analysis.

One possible approach to this problem is to use Monte-Carlo techniques, which will be discussed below. Alternatively, it may be possible to produce lists of the top n most probable configurations using a model without these per-part features, and then explicitly evaluate the probabilities under the more complex model in order to re-rank them.

6.10.2 Tree-width constraints

It has already been shown that the time taken to perform inference is strongly dependent on the tree-width of the constructed graph. Ideally the fully connected graph should be used, as no matter how weak the interaction between two vertices there is no harm in taking it into account. Any deletion of edges from this graph should therefore be viewed as an approximation, and there is a trade-off between computational complexity and performance which can be varied by adjusting the graph which is used.

The method described above used a heuristic based on the spatial distance between ink strokes to construct the graph. While this was observed to give good results in practice, it may be worthwhile using a more sophisticated approach to refine the model further. Such an approach would attempt to construct a graph which is close to optimal in terms of the complexity/accuracy tradeoff.

While it is clearly not possible to know the results of the inference in advance, it is possible to estimate the importance of an edge from the local potential. By assigning a weight to each potential edge using this information, the goal would be to construct the graph with the

largest total edge weight given a constrained tree-width, with the constraint being set by the resources available to do inference.

Restricting the tree-width in this way is of course a non-trivial task and in practice it would most probably be necessary to resort to an approximate method. A simple rule would be a greedy algorithm, which could proceed either by starting with the fully disconnected graph and adding the most heavily weighted edges subject to the tree-width bound not being exceeded. Alternatively, the algorithm could start with a fully connected graph and delete edges.

6.10.3 Approximate inference

Monte Carlo methods

Monte Carlo methods provide a way of sampling from otherwise intractable probability distributions. By using such a method it is possible to sample from the set of labelled partitions of a previously unseen graph using an already trained model. One technique which is particularly appropriate for this model is that of Swendsen–Wang cuts [89]. This method would permit sampling from partitions of a much more densely connected graph than is tractable using exact inference.

While not exactly representing the marginal distribution, the pairwise potentials provide a relatively informative estimate of the relationship between neighbouring vertices. At each update step Swendsen–Wang cuts use this information to propose updates to the labelled partition, in the form of merges and splits of partitions and changes to the labels of existing parts. These proposals can then be accepted stochastically using a Metropolis–Hastings like update rule. This method has previously been applied to image segmentation [3], which is in many respects very similar to ink analysis.

Other approximations

An alternative to using a Monte Carlo algorithm is to introduce an approximation to the full distribution which is designed to be as close as possible but permit efficient inference.

One such approach is loopy belief propagation [59]. In traditional problems this is performed by using the same update rule as ordinary belief propagation, but operating on untriangulated graphs. In this case the junction graph is not a tree, and messages can potentially propagate infinitely around loops, which invalidates the proofs of exact inference provided for the non-loopy case. However, in many cases the process converges and good performance is achieved.

In the case of partitioning, there is however an additional problem, namely that partitions are only permitted which are consistent. Consider the case of a large ring of vertices in which most of the edges give strong evidence that all vertices should be assigned to the same part, with the exception of a single edge which has strong evidence that its vertices belong to different parts. In this situation the graph is frustrated - going around the loop there would

be a preference for a single boundary, which is not consistent. The exact inference method effectively introduces an additional potential for the whole ring enforcing consistency, as is illustrated by the edge dual representation, but constraining things in this way is not possible without triangulating the ring. It is not clear how a loopy algorithm would behave in such a situation, so further investigation is needed.

An alternative approach would be to use an approximation based on a reduced tree width graph. Consistency could be enforced on the approximate graph. Such an approach might be based on variational inference, where an approximate distribution is selected from a permitted family to minimise the KL divergence between the approximation and the exact distribution. A further option is the use of expectation propagation, which optimises the approximate distribution by matching moments.

6.11 Conclusions

This chapter has presented a probabilistic model over labelled partitions of an undirected graph, and has shown that the structure of the graph may be used to efficiently perform exact inference with message passing algorithms. An application of the model to the task of parsing hand-drawn diagrams has been demonstrated. Experimental results illustrate that it is possible to obtain high-quality results using this technique. The results obtained prove that in the present application, labelling accuracy is improved by performing partitioning at the same time.

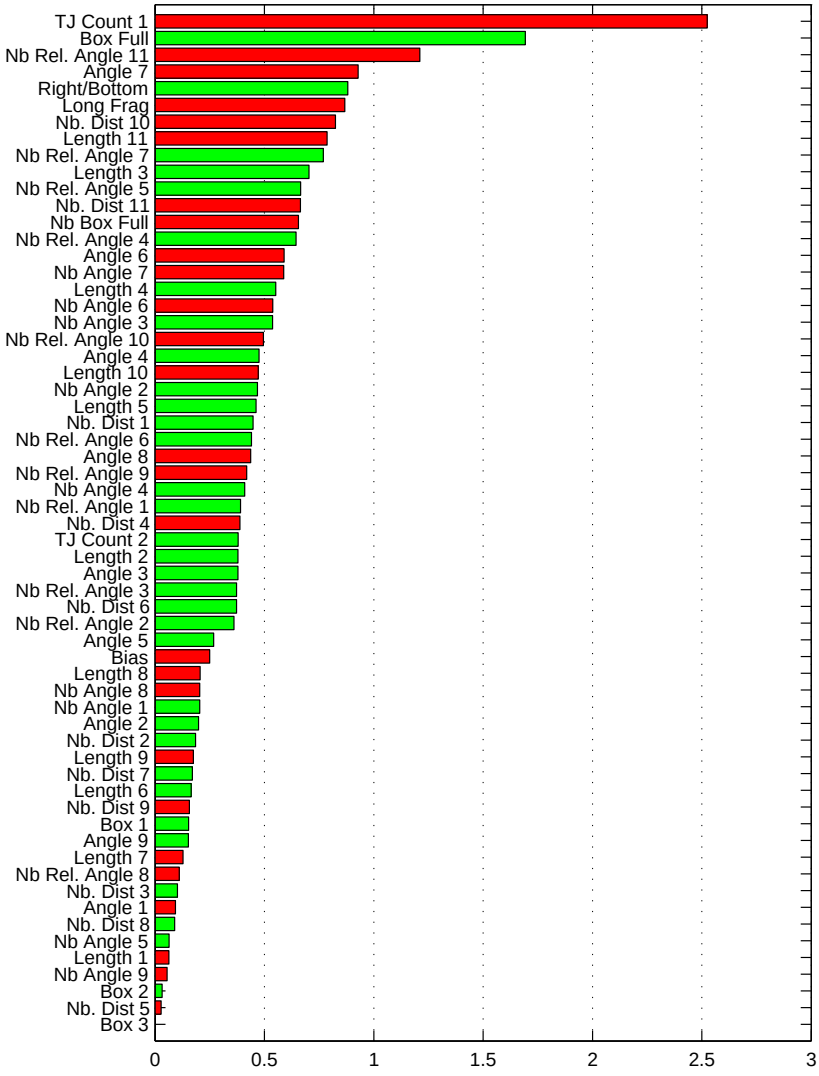


Figure 6.10: Learned unary weights. Positive values are shown in green and indicate a preferences for containers. Negative weights are shown in red and indicate a preference for connectors.

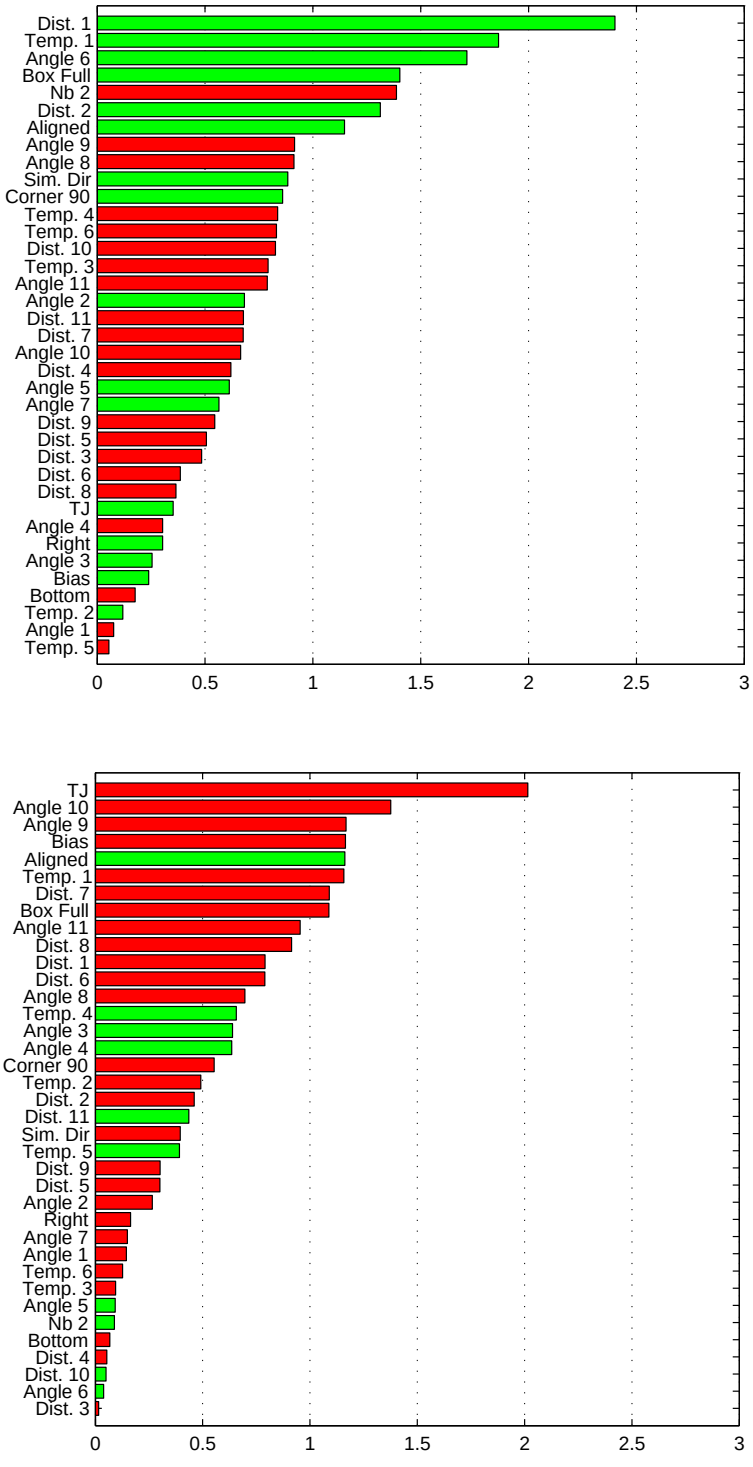


Figure 6.11: Pairwise weights for same object/same label (top) and different object/same label (bottom) configurations. Positive weights are shown in green and favour the named configuration. Negative weights are shown in red.

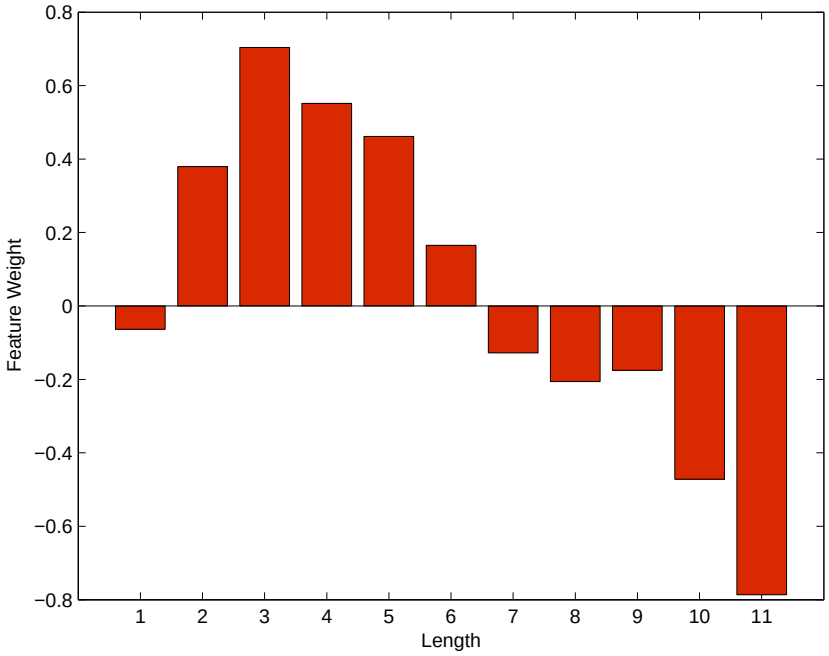


Figure 6.12: Feature for the unary length features. Positive weights correspond to a preference for categorisation as a container rather than a connector.

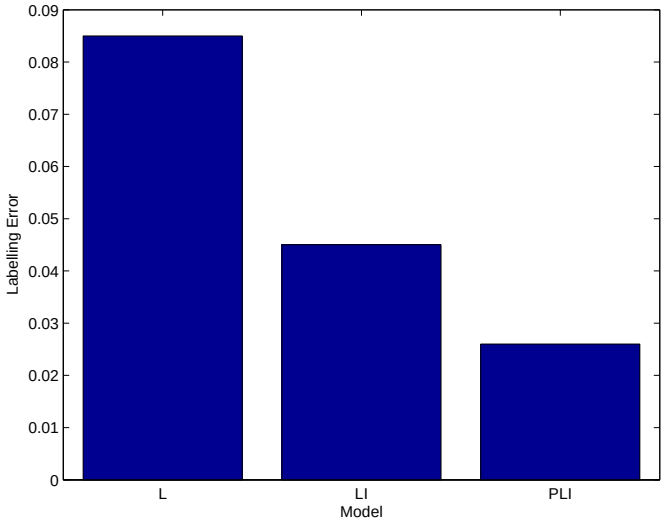


Figure 6.13: Mean labelling errors for the three models.

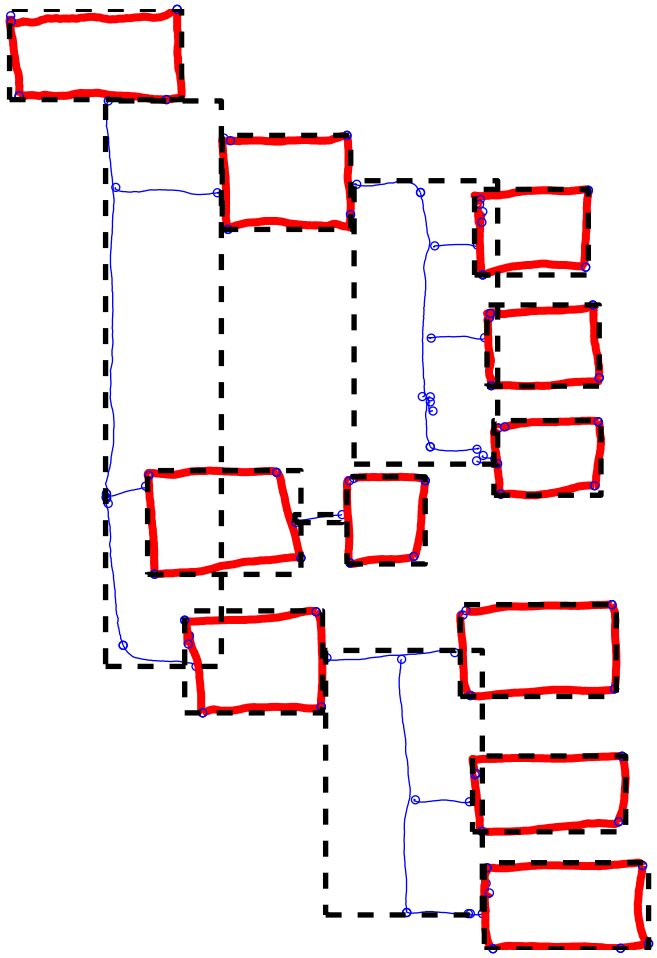


Figure 6.14: Example labellings and groupings: the most probable partition and labelling using model PLI. Heavy lines indicate fragments which have been classified as containers and lighter lines indicate connectors. Groups of fragments which belong to the same part are outlined using a dashed box.