Efficient Training of Conditional Random Fields

Hanna Wallach

Master of Science
School of Cognitive Science
Division of Informatics
University of Edinburgh

2002

Abstract

This thesis explores a number of parameter estimation techniques for conditional random fields, a recently introduced [31] probabilistic model for labelling and segmenting sequential data. Theoretical and practical disadvantages of the training techniques reported in current literature on CRFs are discussed. We hypothesise that general numerical optimisation techniques result in improved performance over iterative scaling algorithms for training CRFs. Experiments run on a a subset of a well-known text chunking data set [28] confirm that this is indeed the case. This is a highly promising result, indicating that such parameter estimation techniques make CRFs a practical and efficient choice for labelling sequential data, as well as a theoretically sound and principled probabilistic framework.

Acknowledgements

I would like to thank my supervisor, Miles Osborne, for his support and encouragement throughout the duration of this project.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Hanna Wallach*)

Table of Contents

Chapter 1

Introduction

The task of assigning label sequences to a set of observation sequences arises in many fields, including bioinformatics, computational linguistics, speech recognition and information extraction. As an example, consider the natural language processing (NLP) task of labelling the words in a sentence with their corresponding part-of-speech (POS) tags. In this task, each word is labelled with a tag or indicating its appropriate part of speech, resulting in annotated text, such as:

(1.1) [PRP He] [VBZ reckons] [DT the] [JJ current] [NN account] [NN deficit] [MD will] [VB narrow] [TO to] [RB only] [# #] [CD 1.8] [CD billion] [IN in] [NNP September] [. .]

Labelling sentences in this way is a useful preprocessing step for higher level natural processing tasks: POS tags augment the information contained within words alone by explicitly indicating some of the structure inherent in language. Another NLP task involving sequential data is that of text chunking, or shallow parsing. Text chunking involves the segmentation of natural sentences (usually augmented with POS tags) into non-overlapping phrases, such that syntactically related words are grouped together in the same phrase. For example, the sentence used in the POS tagging example may be divided as follows:

(1.2) [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September] [O .].

Like POS tagging, which is used as a preprocessing step for tasks such as text chunking, shallow parsing provides a useful intermediate step when fully parsing natural language data – a task that is highly complex and benefits from as much additional information as possible.

One of the most common methods for performing such labelling and segmentation tasks is that of employing hidden Markov models [45] (HMMs) or probabilistic finite state automata [40] to identify the most likely sequence of labels for the words in any given sentence. HMMs are a form of generative model, that assign a joint probability $p(\mathbf{x}, \mathbf{y})$ to pairs of observation and label sequences, $\mathbf{x}$ and $\mathbf{y}$ respectively. In order to define a joint probability of this nature, generative models must enumerate all possible observation sequences – a task which, for most domains, is intractable unless observation elements are represented as isolated units, independent from the other elements in an observation sequence. This is an appropriate assumption for a few simple data sets, however most real-world observation sequences are best represented in terms of multiple interacting features and long-range dependencies between observation elements.

This representation issue is one of the most fundamental problems when labelling sequential data. Clearly, a model that supports tractable inference is necessary, however a model that represents the data without making unwarranted independence assumptions is also desirable. One way of satisfying both these criteria is to use a model that defines a conditional probability $p(\mathbf{y}|\mathbf{x})$ over label sequences given a particular observation sequence, rather than a joint distribution over both label and observation sequences. Conditional models are used to label a novel observation sequence $\mathbf{x}$ by selecting the label sequence $\mathbf{y}$ that maximises the conditional probability $p(\mathbf{y}|\mathbf{x})$. The conditional nature of such models means that no effort is wasted on modelling the observation sequences. Furthermore, by specifying the conditional model in terms of a log-linear distribution, one is free from making unwarranted independence assumptions. Arbitrary facts about the observation data can be captured without worrying about how to ensure that the model is correct.

A number of conditional probabilistic models have been recently developed for use instead of generative models when labelling sequential data. Some of these models [12, 33] fall into the category of non-generative Markov models, while others [31] define a single probability distribution for the joint probability of an entire label sequence given an observation sequence. As expected, the conditional nature of models such as McCallum *et al.*'s maximum entropy Markov models (MEMMs) [33], a form of next-state classifier, result in improved performance on a variety of well-known NLP labelling tasks. For instance, a comparison of HMMs and MEMMs for POS tagging [31] showed that use of conditional models such as MEMMs resulted in a significant reduction in the per-word error rate from that obtained using HMMs. In particular, use of an MEMM that incorporated a small set of orthographic features reduced the overall per-word error rate by around 25% and the out-of-vocabulary error rate by around 50%.

Unfortunately, non-generative finite-state models are susceptible to a weakness known as the *label bias problem* [31]. This problem, discussed in detail in Chapter 2, arises from the per-state normalisation requirement of next-state classifiers – the probability transitions leaving any given state must sum to one. Each transition distribution defines the conditional probabilities of possible next states given the current state and next observation element. Therefore, the per-state normalisation requirement means that observations are only able to affect which successor state is selected, and not the probability mass passed onto that state. This results in an bias towards states with low entropy transition distributions and, in the case of states with a single outgoing transition, causes the observation to be effectively ignored. The label bias problem can significantly undermine the benefits of using a conditional model to label sequences, as indicated by experiments performed by Lafferty *et al.* [31]. These experiments show that simple MEMMs, equivalent to HMMs in the observation representation used, perform considerably worse than HMMs on POS tagging tasks as a direct consequence of the label bias problem.

To reap the benefits of using a conditional probabilistic framework for labelling sequential data and simultaneously overcome the label bias problem, Lafferty *et al.* [31] introduced *conditional random fields* (CRFs), a form of undirected graphical model that defines a single log-linear distribution over for the joint probability of an entire label sequence given a particular observation sequence. This single distribution neatly removes the per-state normalisation requirement and allows entire state sequences to be accounted for at once by letting individual states pass on amplified or dampened probability mass to their successor states. Sure enough, when simple CRFs are compared with the MEMMs and HMMs used to demonstrate the performance effects of the label bias problem, CRFs outperform both MEMMs and HMMs, indicating that a using a principled method of dealing with the label bias problem is highly advantageous.

Lafferty *et al.* propose two algorithms for estimating the parameters of CRFs. These algorithms are based on improved iterative scaling (IIS) and generalised iterative scaling (GIS) – two techniques for estimating the parameters of non-sequential maximum entropy log-linear models. Unfortunately, careful analysis (described in Chapter 3) reveals Lafferty *et al.*'s GIS-based algorithm to be intractable[1] and their IIS-based algorithm to make a mean-field approximation in order to deal with the sequential nature of the data being modelled that may result in slowed convergence. Lafferty *et al.*'s experimental results involving CRFs for POS tagging indicate that convergence of their IIS variant is very slow indeed – when attempting to train a CRF initialised with an all zero parameter vector, Lafferty *et al.* found that convergence had not been reached even after 2000 iterations. To deal with this very slow convergence, Lafferty *et al.* train a MEMM to convergence, taking 100 iterations, and then use its parameters as

[1] Calculating the expected value of each correction feature (necessary to enable analytic calculation of parameter update values) is intractable due to the global nature of the correction features.

the initial parameter vector for the CRF. Convergence of the IIS-variant for CRF parameter estimation then converged in around 1000 iterations. Although this technique enabled Lafferty *et al.* to train CRFs in a reasonable time, this is not a principled technique and is entirely dependent on the availability of trained MEMMs that are structurally equivalent to the CRF being trained. Additionally, a recent study by Bancarz and Osborne [4] has shown that IIS can yield multiple globally optimal models that result in radically differing performance levels, depending on initial parameter values. This observation may mean that the decision to start CRF training using the trained parameters of an MEMM is in fact biasing the performance of CRFs reported in the current literature.

These theoretical and practical problems with the parameter estimation methods currently proposed for CRFs provide significant impetus for investigating alternative parameter estimation algorithms that are easy to implement and efficient. Interestingly, recent experimental work of Malouf [32] indicated that despite widespread use of iterative scaling algorithms for training (non-sequential) conditional maximum entropy models, general numerical optimisation techniques outperform iterative scaling by a wide margin on a number of NLP datasets. The functional form of the distribution over label sequences given an observation sequence defined by a CRF is very similar to that of a non-sequential conditional maximum entropy model. This functional correspondence suggests that use of general optimisation techniques for CRF parameter estimation is highly likely to result in similar performance advantages to those obtained by using general numerical optimisation techniques for estimating the parameters of a non-sequential conditional maximum entropy model.

This thesis explores a number of parameter estimation techniques for conditional random fields, highlighting theoretical and practical disadvantages of the training techniques reported in current literature on CRFs and confirming that general numerical optimisation techniques do indeed result in improved performance over Lafferty *et al.*'s iterative scaling algorithm. To compare performance of the parameter estimation algorithms considered, a subset of a well-known text chunking data set [28] was used to train a number of CRFs, each with a different parameter estimation technique. Although the particular subset of data chosen was not representative of the size and complexity of the data sets found in most NLP tasks, the experiments performed did indicate that numerical optimisation techniques for CRF parameter estimation result in faster convergence than iterative scaling. This is a highly promising result, indicating that such parameter estimation techniques make CRFs a practical and efficient choice for labelling sequential data, as well as a theoretically sound and principled probabilistic framework.

The structure of this thesis is as follows: In Chapter 2, generative and conditional data labelling techniques based on directed graphical models are introduced and a thorough description of HMMs, MEMMs and the label bias problem is given. Chapter 3 addresses the theoretical framework underlying conditional random fields, including parameter estimation algorithms described in current literature and their theoretical and practical limitations. In Chapter 4, a number of first- and second-order numerical optimisation techniques are discussed and an outline of how such techniques may be applied to the task of estimating the parameters of a CRF is given. Following this, the software implemented to perform CRF parameter estimation is described, and the experimental data used to compare the algorithms is presented. Finally, the results of the experiments and their implications are detailed, before a summary of the work covered in this thesis is presented in Chapter 5.

Chapter 2

Directed Graphical Models

Hidden Markov models [45], probabilistic finite-state automata [40] and maximum entropy Markov models [33] may all be represented as *directed graphical models* [26]. Directed graphical models are a framework for explicating the independence relations between a set of random variables, such as the variables $\mathbf{S}_1, \ldots, \mathbf{S}_n$ representing the state of a HMM at times $t = 1$ through to $t = n$. These independence relations may then be used to construct a concise factorisation of the joint distribution over the states in a Markovian model (and in the case of HMMs, over the observations also). When labelling sequential data using a HMM or MEMM, each of the labels is represented by one or more states in the Markov model, so defining a probability distribution over state sequences is equivalent to defining a distribution over possible sequences of labels.

This chapter introduces the theory underpinning directed graphical models and explains how they may be used to identify a probability distribution over a set of random variables. A description of hidden Markov models and their uses in natural language processing is presented, along with a discussion of the limitations of using generative models for labelling sequential data. Maximum entropy Markov models [33], a form of conditional next-state classifier, are introduced in the context of a solution to the problems encountered when using generative models for segmentation of sequence data. Finally, the label bias problem [31], a fundamental weakness of non-generative finite-state models, is described, motivating the need for a conditional model that also provides a principled method of overcoming this problem.

2.1 Directed Graphical Models

A directed graphical model consists of an acyclic directed graph $G = (V, E)$ where V is the set of nodes belonging to G and E is the set of directed edges between the nodes in V. Every node V_i in the set of nodes V is in direct one-to-one correspondence with a random variable, also denoted as V_i[1]. This correspondence between nodes and random variables enables every directed graphical model to represent a class of joint probability distributions over the random variables in V.

The directed nature of G means that every node V_i has a set of parent nodes V_{π_i}, where π_i is the set of indices of the parents of node V_i. The relationship between a node and its parents enables the expression for the joint distribution defined over the random variables V to be concisely factorised into a set of functions that depend on only a subset of the nodes in G. Specifically, we allow the joint distribution to be expressed as the product of a set of local functions, such that every node in G is associated with a distinct function $f_i(v_i, v_{\pi_i})$ in this set defined over the node and its parents:

$$p(v_1, v_2, \ldots, v_n) \triangleq \prod_{i=1}^{n} f_i(v_i, v_{\pi_i}). \tag{2.1}$$

To identify the functional form of each of these f_i, we turn to the notion of conditional independence. In particular, we observe that the structure of a directed graphical model embodies specific conditional independence assumptions which can be used to factor the joint distribution such that a natural probabilistic interpretation of each f_i emerges. Given three non-overlapping sets

[1]This one-to-one correspondence means that we ignore any distinction between nodes and random variables, and use the terms "node" and "random variable" interchangeably.

of nodes V_A, V_B and V_C the definition of conditional independence states that nodes V_A and V_C are conditionally independent given the nodes in V_B if and only if the probability of v_A given v_C and v_B can be is given by

$$p(v_A|v_B, v_C) = p(v_A|v_B). \tag{2.2}$$

To relate the concept of conditional independence to the structure of a directed graphical model, we define a topological ordering of the nodes V in G, such that the nodes in V_{π_i} appear before V_i in the ordering for all V_i. Having chosen an ordering of the nodes, all conditional independence relations between random variables in G can be expressed by the statement

node V_i is conditionally independent of V_{V_i} given V_{π_i}

where V_{V_i} is the set of nodes that appear before V_i in the topological ordering exclusive of the parents V_{π_i} of V_i. This conditional independence statement allows the joint probability distribution over the random variables in a directed graphical model to be factorised using the probability chain rule, giving an explicit probabilistic interpretation of each local function $f_i(v_i, v_{\pi_i})$. More precisely, each f_i is in fact the conditional probability of v_i given v_{π_i}

$$f_i(v_i, v_{\pi_i}) = p(v_i|v_{\pi_i}) \tag{2.3}$$

which enables the the joint distribution to be defined as

$$p(v_1, v_2, \ldots, v_n) = \prod_{i=1}^{n} p(v_i|v_{\pi_i}). \tag{2.4}$$

To see how this method of factorising a joint distribution over random variables may be used to concisely express the probability distribution over a sequence of labels, we look at two forms of Markovian model – hidden Markov models [45] and maximum entropy Markov models [33].

2.2 Hidden Markov Models

Hidden Markov models have been successfully applied to many data labelling tasks including POS tagging [30], shallow parsing [43, 51, 34], speech recognition [44?] and gene sequence analysis [18]. Revisiting the part-of-speech tagging scenario introduced in Chapter 1, we illustrate the use of HMMs for labelling and segmenting sequential data using the task of annotating words in a body of text with appropriate part-of-speech tags, producing labelled sentences of the form:

(2.5) [PRP He] [VBZ reckons] [DT the] [JJ current] [NN account] [NN deficit] [MD will] [VB narrow] [TO to] [RB only] [# #] [CD 1.8] [CD billion] [IN in] [NNP September] [. .]

HMMs are probabilistic finite state automata [40, 22] that model generative processes by defining joint probabilities over observation and label sequences [45]. Each observation sequence is considered to have been generated by a sequence of state transitions, beginning in some start state and ending when some predesignated final state is reached. At each state an element of the observation sequence is stochastically generated, before moving to the next state. In the context of POS tagging, each state of the HMM is associated with a POS tag. A one-to-one relationship between tags and states is not necessary, however, to simplify matters we consider this to be the case. Although POS tags do not generate words, the tag associated with any given word can be considered to account for that word in some fashion. It is, therefore, possible to find the sequence of POS tags that best accounts for any given sentence by identifying the sequence of states most likely to have been traversed when "generating" that sequence of words.

The states in an HMM are considered to be hidden because of the doubly stochastic nature of the process described by the model. For any observation sequence, the sequence of states that best accounts for that observation sequence is essentially hidden from an observer and can only be viewed through the set of stochastic processes that generate an observation sequence. Returning to the POS tagging example, the POS tags associated with any sequence of words and may must identified by inspecting the process by which the words were "generated". The principle of identifying the most state sequence that best accounts for an observation sequence forms the foundation underlying the use of finite-state models for labelling sequential data.

Formally, an HMM is fully defined by

- A finite set of states $\mathcal{S}$.
- A finite output alphabet $\mathcal{X}$.
- A conditional distribution $P(s'|s)$ representing the probability of moving from state s to state s', where $s, s' \in \mathcal{S}$.
- An observation probability distribution $P(x|s)$ representing the probability of emitting observation x when in state s, where $x \in \mathcal{X}$ and $s \in \mathcal{S}$.
- An initial state distribution $P(s)$, $s \in \mathcal{S}$.

Returning to the notion of a directed graphical model as an expression of the conditional independence relationships between a set of random variables, a HMM may be represented as a directed graph G with nodes $\mathbf{S}_t$ and $\mathbf{X}_t$ representing the state of the HMM (or label) at time t and the observation at time t, respectively. This structure is shown in Figure 2.1. This representation

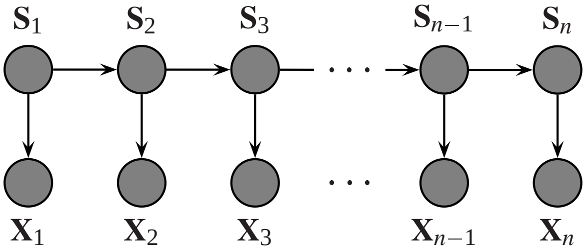

Figure 2.1: Dependency graph structure for first-order HMMs for sequences.

of a HMM clearly highlights the conditional independence relations within a HMM. Specifically, the probability of the state at time t depends only on the state at time $t-1$. Similarly, the observation generated at time t only depends on the state of the model at time t. In the POS tagging application, this means that we are considering the tag $\mathbf{y}_t$ (recall that we are assuming a one-to-one correspondence between states and tags) of each word $\mathbf{x}_t$ to depend only on the tag assigned to the previous word $\mathbf{y}_{t-1}$, and each word $\mathbf{x}_t$ to depend only on the current POS tag $\mathbf{y}_t$. These conditional independence relations, combined with the probability chain rule, may be used to factorise the joint distribution over a state sequence $\mathbf{s}$ and observation sequence $\mathbf{x}$ into the product of a set of conditional probabilities:

$$p(\mathbf{s},\mathbf{x}) = p(\mathbf{s}_1)p(\mathbf{x}_1|\mathbf{s}_1)\prod_{t=2}^{n} p(\mathbf{s}_t|\mathbf{s}_{t-1})p(\mathbf{x}_t|\mathbf{s}_t). \tag{2.6}$$

2.2.1 Labelling Sequential Data

As stated above, labelling an observation sequence is the task of identifying the sequence of labels that best accounts for the observation sequence. In other words, when choosing the most appropriate label sequence for an observation sequence $\mathbf{x}$ we want to choose the label sequence $\mathbf{y}^*$ that maximises the conditional probability of the label sequence given the observation sequence:

$$\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) \tag{2.7}$$

However, since the distribution defined by an HMM is a joint distribution $p(\mathbf{x},\mathbf{s})$ over observation and state sequences, the most appropriate label sequence for any observation sequence is obtained by finding the finding sequence of states $\mathbf{s}^*$ that maximises the conditional probability of the state sequence given the observation sequence, which may be calculated from the joint distribution using Bayes' rule:

$$\mathbf{s}^* = \mathrm{argmax}_{\mathbf{s}} \frac{p(\mathbf{x},\mathbf{s})}{p(\mathbf{x})}. \tag{2.8}$$

and then reading off the labels $\mathbf{y}$ associated with the states in this sequence. Finding the optimal state sequence is most efficiently performed using a dynamic programming technique known as Viterbi alignment. The Viterbi algorithm is described in [45].

2.2.2 Limitations of Generative Models

Despite their widespread use, HMMs and other generative models are not the most appropriate sort of model for the task of labelling sequential data. Generative models define a joint probability distribution $p(\mathbf{x}, \mathbf{y})$ over observation and label sequences. This is useful if the trained model is to be used to generate data, however, the distribution of interest when labelling data is the conditional distribution $p(\mathbf{y}|\mathbf{x})$ over label sequences given the observation sequence in question. Defining a joint distribution over label and observation sequences means that all possible observation sequences must be enumerated – a task which is hard if observations elements are assumed to have long-distance dependencies. Therefore, generative models must make strict independence assumptions in order to make inference tractable. In the case of an HMM, the observation at time t is assumed to depend only on the state at time t, ensuring that each observation element is treated as an isolated unit, independent from all other elements in the sequence.

In fact, most sequential data cannot be accurately represented as a set of isolated elements. Such data contain long-distance dependencies between observation elements and benefit from being represented in by a model that allows such dependencies and enables observation sequences to be represented by non-independent overlapping features. For example, when assigning POS tags to words, performance is improved significantly by assigning tags on the basis of complex feature sets that utilise information such as the identity of the current word, the identity of surrounding words, the previous two POS tags, whether a word starts with a number or upper case letter, whether the word contains a hyphen, and the suffix of the word [46, 31]. These features are not independent (for example, the suffix of the current word is entirely dependent on the identity of the word) and contain dependencies other than those between the current and previous tags, and the current word and current tag.

Fortunately, the use of conditional models for labelling data sequences provides a convenient method of overcoming the strong independence assumptions required by practical generative models. Rather than modelling the joint probability distribution $p(\mathbf{x},\mathbf{s})$ over observations and states, conditional models define a conditional distribution $p(\mathbf{s}|\mathbf{x})$ over state sequences given a particular observation sequence. This means that when identifying the most likely state sequence for a given observation sequence, the conditional distribution may be used directly, rather than using

$$\mathbf{s}^* = \text{argmax}_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}) = \text{argmax}_{\mathbf{s}} \frac{p(\mathbf{x},\mathbf{s})}{p(\mathbf{x})}, \tag{2.9}$$

which requires enumeration of all possible observation sequences so that the marginal probability $p(\mathbf{x})$ can be calculated.

2.3 Maximum Entropy Markov Models

Maximum entropy Markov models [33] are a form of conditional model for labelling sequential data designed to address the problems that arise from the generative nature and strong independence assumptions of hidden Markov models. MEMMs have been applied to a number of labelling and segmentation tasks including POS tagging [31] and the segmentation of text documents [33].

Like HMMs, MEMMs are also based on the concept of a probabilistic finite state model, however, rather generating observations the model is a probabilistic finite state acceptor [40] that outputs label sequences when presented with an observation sequence. MEMMs consider observation sequences to be events to be conditioned upon rather than generated. Therefore, instead of defining two types of distribution – a transition distribution $P(s'|s)$ representing the probability of moving from state s to state s' and an observation distribution $P(x|s)$ representing the probability of emitting observation x when in state s – a MEMM has only a single set of $|S|$ separately trained distributions of the form

$$P_s(s'|x) = P(s'|s,x) \tag{2.10}$$

which represent the probability of moving from state s to s' on observation x. The fact that each of these functions is specific to a given state means that the choice of possible states at any given instant in time $t+1$ depends only on the state of the model at time t. The use of state-observation transition functions which are conditioned on the observations means that the dependency graph for a MEMM takes the form shown in Figure 2.2. Note that the observation

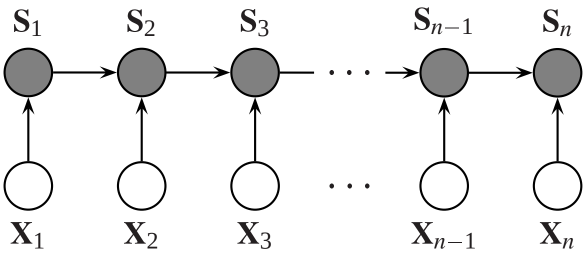

Figure 2.2: Graphical structure of first-order MEMMs for sequences. The variables corresponding to unshaded nodes are not generated by the model.

sequence is being conditioned upon rather than generated, and so the distribution associated with the graph is the joint distribution of only those random variables $\mathbf{S}_t$ representing the state of the MEMM at time t. Assuming that each state corresponds to a particular label, the chain rule of probability and the conditional independences embodied in the MEMM dependency graph structure may be used to factorise the joint distribution over label sequences $\mathbf{y}$ given the observation sequence $\mathbf{x}$ as:

$$p(\mathbf{y}\mathbf{x}) = p(\mathbf{y}_1|\mathbf{x}_1)\prod_{t=2}^{n} p(\mathbf{y}_t|\mathbf{y}_{t-1},\mathbf{x}_t) \tag{2.11}$$

Treating observations as events to be conditioned upon rather than generated means that the probability of each transition may depend on non-independent, interacting features of the observation sequence. McCallum *et al.* [33] do this by making use of the maximum entropy framework, which is discussed in detail in Chapter 3, and defining each state-observation transition function to be a log-linear model:

$$P_s(s'|x) = \frac{1}{Z(s,x)} \exp\left(\sum_k \lambda_k f_k(s',x)\right) \tag{2.12}$$

where $Z(s,x)$ is a normalisation factor, each λ_k are parameters to be estimated and each f_k is a feature function that takes two arguments, the current observation and a potential next state. The free parameters of each log-linear model can be estimated using Generalised Iterative Scaling [17]. Iterative scaling is also covered in Chapter 3. Each feature function makes use of a binary feature b of the observation which expresses some characteristic of the empirical training distribution that should hold of the trained model distribution also. An example of such a feature is

$$b(x) = \begin{cases} 1 & \text{if the observation is the word "the"} \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

Each feature function f_k indicates whether a particular boolean feature b is true of the observation, and whether the possible next state takes on a particular value:

$$f_{<b,s>}(s',x) = \begin{cases} 1 & \text{if } b(x) \text{ is true and } s = s' \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

2.3.1 Labelling Sequential Data

Like HMMs, MEMMs are used to label novel data by identifying the state sequence that best describes the observation sequence to be labelled. Each state has a label associated with it and so the most probable label sequence for that observation sequence may be trivially identified once the most likely state sequence has been calculated. To find the most probable state sequence $\mathbf{s}^*$, where

$$\mathbf{s}^* = \text{argmax}_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}), \tag{2.15}$$

it is desirable to use some form of dynamic programming algorithm. McCallum *et al.* [33] present a brief overview of a variant on Viterbi alignment that enables the this state sequence to be efficiently identified.

2.3.2 The Label Bias Problem

Maximum entropy Markov models and other discriminative finite-state models that define a set of separately trained per-state probability distributions [40, 12] exhibit undesirable behaviour in certain situations, termed *label bias* by Lafferty *et al.* [31]. The label bias problem is best described by through use of an example. Consider the MEMM in Figure 2.3. This finite-state acceptor is

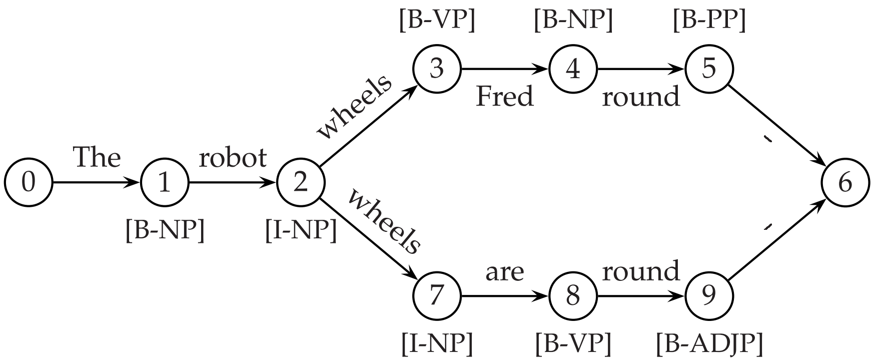

Figure 2.3: Finite-state acceptor for shallow parsing two sentences.

designed to shallow parse the sentences

(2.16) the robot wheels Fred round

(2.17) the robot wheels are round

by segmenting them into into non-overlapping chunks or phrases such that syntactically related words are grouped together.

Suppose we wish to determine the most likely chunk sequence for observation sentence 2.17. This is done by identifying the state sequence that best accounts for the observation sequence and then reading the chunk labels off the states in the sequence chosen. Recalling that the joint probability of a state sequence **s** given an observation sequence **x** may be decomposed so that

$$p(\mathbf{s}|\mathbf{x}) = \prod_{t=1}^{n} p(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{x}_t), \tag{2.18}$$

we must calculate $p(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{s}_t)$ for every s_t in each of the state sequences that accounts for the observation sequence. Having done this, establishing which which state sequence results in the highest conditional probability $p(\mathbf{s}|\mathbf{x})$ is trivial.

At the first and second time steps, the observation words "the" and "robot" match the transitions from state 0 to state 1 and state 1 to state 2 respectively. So far, the joint probability of the only possible state sequence given the observations so far is

$$p(1, 2|\text{the robot}) = p(1|0, \text{the})p(2|1, \text{robot}). \tag{2.19}$$

Moving on, the observation element "wheels" matches both transitions from state 2 and so the probability mass accumulated so far must be distributed between states 3 and 7. This conservation of probability mass arises from the fact that the probability transitions leaving any given state must sum to one. There are now two possible state sequences given the observation sequence so far – 0123 and 0127. To determine which of these sequences best accounts for the observation sequence, we must observe how each of these sequences can be extended to account for the rest of the observation sequence, and compare the probabilities of the state sequences that result.

States 3 and 7 both have a single outgoing transition and so each one must pass the probability mass accumulated so far to its successor to ensure that the distribution over all possible state sequences given this observation sequence sums to one. This per-state normalisation requirement that arises from the fact that the joint distribution over state sequences is decomposed to a product of conditional probabilities of next states given the current states and the next observation. We therefore assume that

$$p(4|3, \text{Fred}) = p(8|7, \text{Fred}) = 1. \tag{2.20}$$

However, closer inspection reveals that this assumption is entirely unwarranted by the data used to train the MEMM – the training data never contained a transition from state 7 to state 8 on the observation word "Fred". Ideally, we would like to a low probability to this event given that it never occurred in the data used to train the model, however, the per-state normalisation requirement means we have no choice but to ignore the observation and and respect the fact that

$$\sum_{\substack{\text{all states } s \\ \text{reachable from 7}}} p(s|7, \text{Fred}) = 1 \tag{2.21}$$

Essentially, states with a single outgoing transition are forced to ignore observations. This may be generalised to a tendency for states with low-entropy next-state distributions to take little notice of observations.

Continuing to identify state sequences that account for the observation sequence in question, we end up with two possible state sequences: 0123456 and 0127896 corresponding to the chunk labels "B-NP I-NP B-VP B-NP B-PP" and "B-NP I-NP I-NP B-VP B-ADJP" respectively. Assuming the probabilities of each of the transitions out of state 2 are approximately equal, the label bias problem means that the probability of each of these chunk sequences given an observation sequence $\mathbf{x}$ will also be roughly equal irrespective of the observation sequence $\mathbf{x}$.

On a related note, had one of the transitions out of state 2 occurred more frequently in the training data set, the probability of that transition would always be greater, causing state 2 to pass more of its probability mass to the successor state associated with that transition. This situation would result in the sequence of chunk tags associated with that path being preferred irrespective of the observation sentence. This consequence of the label bias problem is likely to be a significant problem for natural language data – for example, consider a situation in which a word is almost always associated with a particular part-of-speech tag. When labelling a sentence for which the rarer interpretation of that word is the correct one, the word will be assigned the more common POS tag irrespective of the context contained in the rest of the sentence.

Generative models such as HMMs do not suffer from the label bias problem. This is because the Viterbi algorithm used to identify the most likely state sequence given an observation sequence is able to down-weight a possible branch of a state sequence on the basis of observations that appear after the branch point.

2.4 Performance of HMMs and MEMMs

To compare the performance of MEMMs and HMMs, McCallum *et al.* [33] performed a number of experiments involving the segmentation of Usenet multipart FAQs. Three HMM variants (see [33] for details) and a simple MEMM were trained on trained on a single document in a group of FAQs and tested on the remaining documents. The features used in the MEMM transition functions were all formatting features such as indentation, numbered questions and styles of paragraph breaks. Such features are not necessarily independent. The results of these experiments are summarised in Table 2.1. This table clearly indicates that MEMMs outperform HMMs when non-independent salient features are an appropriate representation of the observation data.

Model	COAP	Precision	Recall
TokenHMM	0.865	0.276	0.140
FeatureHMM	0.941	0.413	0.529
MEMM	0.965	0.867	0.681

Table 2.1: Co-occurance agreement probability (COAP), segmentation precision and segmentation recall of HMM variants and an MEMM on a text segmentation task [33]

Although MEMMs exhibited good performance on this text segmentation task, it is possible that they perform worse than HMMs when labelling data that results in the occurance of the label bias problem. To investigate this, Lafferty *et al.* [31] performed a number of experiments comparing HMMs, which do not suffer from label bias, with MEMMs both for POS tagging and labelling synthetic data specifically generated to verify the label bias problem. On both the synthetic and natural language data, MEMMs performed much worse than simple HMMs as a direct result of the label bias problem. The results of the POS tagging task are shown in Table 2.2.

Model	Error	OOV Error
HMM	5.69%	45.99%
MEMM	6.37%	54.61%

Table 2.2: Per-word error rate and out-of-vocabulary per-word error rate for POS tagging using HMMs and MEMMs [31].

2.5 Chapter Summary

This chapter introduced the concept of directed graphical models for labelling sequential data. An overview of hidden Markov models [45] was provided, and the limitations of such generative models were discussed. Maximum entropy Markov models [33] were introduced as a potential method of overcoming the strong independence assumptions required by practical generative models. Finally, the label bias problem [31] was outlined, motivating the need for a conditional model that does not suffer from this problem.

Chapter 3

Conditional Random Fields

Conditional random fields (CRFs) are a recently introduced [31] form of conditional model that allow the strong independence assumptions of HMMs to be relaxed, as well as overcoming the label-bias problem exhibited by MEMMs [33] and other non-generative directed graphical models such as discriminative Markov models [12]. Like MEMMs, CRFs are conditional probabilistic sequence models, however, rather than being directed graphical models, CRFs are undirected graphical models. This allows the specification of a single joint probability distribution over the entire label sequence given the observation sequence, rather than defining per-state distributions over the next states given the current state. The conditional nature of the distribution over label sequences allows CRFs to model real-world data in which the conditional probability of a label sequence can depend on non-independent, interacting features of the observation sequence. In addition to this, the exponential nature of the distribution chosen by Lafferty *et al.* [31] enables features of different states to be traded off against each other, weighting some states in a sequence as being more important than others.

In this chapter, an introduction to undirected graphical models is given, followed by an explanation of conditional random fields as a form of undirected graphical model. The maximum entropy principle, which heavily influences

Lafferty *et al.*'s [31] choice of CRF potential functions, is described, leading to an explanation of the CRF functional form. Finally, an overview and analysis of the parameter estimation techniques that are currently used for CRF training is given, highlighting the theoretical reasons for desiring an alternative method of training CRFs.

3.1 Undirected Graphical Models

A Markov random field, or undirected graphical model, is an acyclic graph $G = (V, E)$ where V is a set of nodes and E is a set of undirected edges between nodes. The nodes V represent a set of continuous or discrete random variables such that there is a one-to-one mapping between the nodes and variables. Every graphical model is associated with a class of joint probability distributions over the random variables represented by nodes in the graph. The parameterisation of these probability distributions depend on conditional independence relations between the random variables within the graph.

In Section 2.1, we saw that the joint probability distribution associated with a directed graphical model $G^d = (V^d, E^d)$ can be factorised into a product of conditional probabilities once conditional independence relationships between the topologically ordered set of nodes in G^d has been identified. Specifically, the joint probability of the random variables represented by nodes $V^d = V_1^d, V_2^d \ldots, V_n^d$ can be written as

$$p(v_1^d, v_2^d, \ldots, v_n^d) = \prod_{i=1}^{n} p(v_i^d | v_{\pi_i}^d), \tag{3.1}$$

where $V_{\pi_i}^d$ is the set of parent nodes belonging to node V_i^d. The parameterisation of a Markov random field is different to that of a directed graphical model. Although it would be possible to assign each node a conditional probability given its neighbours, the undirected nature of Markov random fields means that it is difficult to ensure that the conditional probability of any node given its neighbours is consistent with the conditional probabilities of the other nodes in the graph. This potential for inconsistency means we cannot ensure that the conditional probabilities assigned to the nodes yield a single joint distribution over all random variables in the graph. For this reason, the joint distribution of a Markov random field is not parameterised in terms of conditional probabilities, but is defined as the product of a set of local functions derived from a set of conditional independence axioms.

The first step in parameterising an undirected graphical model $G = (V, E)$ is to identify the sets of nodes upon which each local function should operate. To do this, we use the notion of conditional independence. Letting A, B and C represent disjoint index subsets, the random variables represented by nodes V_A are conditionally independent of those represented by V_B given the nodes represented by V_C if the set of nodes V_B separates V_A from V_C. For an undirected graphical model, we utilise a naïve graph theoretic notion of separation, and say that for V_A to be conditionally independent of V_C given V_B, every path from a node $V_{\{i|i \in A\}}$ to a node $V_{\{j|j \in C\}}$ must pass through at least one node $V_{\{k|k \in B\}}$. To identify the groups of nodes operated on by the set of local functions we note that according to the conditional independence properties of an undirected graphical model G, the absence of an edge between two nodes V_i and V_j in G implies that the nodes must be conditionally independent given all other nodes in the graph. Therefore, when choosing local functions, we must ensure that it is possible to factories the joint probability such that V_i and V_j do not appear in the same local function.

The easiest way to fulfil this factorisation requirement is to assert that each local function may operate only on a set of nodes that forms a fully connected subset of nodes, or clique, within G. Clearly, this ensures that no local function refers to any pair of nodes that are not directly connected, and, if two nodes appear together in a clique, this dependence is made explicit by defining a local function on the clique in which they appear. Further refining this concept of a local function, we observe that if we define each local function to operate on a *maximal* clique, or clique that cannot be extended to include additional nodes and simultaneously remain fully connected, we gain nothing by also defining potential functions on any cliques that form subsets of this maximal clique. Therefore, the simplest set of local functions that equivalently correspond to the conditional independence properties associated with the graph G are the set of functions in which each function is defined on the possible realisations v_c of a maximal clique c of G. These local functions $\psi_{V_c}(v_c)$ are known as *potential functions* and may have any strictly positive, and real-valued functional form.

Unfortunately, the product of a set of positive, real-valued functions is not guaranteed to satisfy the axioms of probability. Therefore, to satisfy these axioms and ensure that the product is indeed a joint probability distribution over the random variables represented by nodes in G, we define a normalisation factor Z, given by

$$Z \triangleq \sum_{v_1,\ldots,v_n} \prod_{c \in C} \psi_{V_c}(v_c) \tag{3.2}$$

where C is the set of all maximal cliques in G. The joint distribution is therefore given by

$$p(v_1,\ldots,v_n) \triangleq \frac{1}{Z} \prod_{c \in C} \psi_{V_c}(v_c). \tag{3.3}$$

Equivalence of this parameterisation of the joint distribution with the conditional independence characterisations of the undirected graph is proved using the Hammersley-Clifford theorem [21, 15].

Even though the joint distribution over the random variables in an undirected graphical model is written as the product of potential functions, it is important to note that an isolated potential function does not have an direct probabilistic interpretation, but instead represent *constraints* on the configurations of the random variables that the function is defined on. This in turn affects the probability of global configurations – a global configuration with a high probability is likely to have satisfied more of these constraints than a global configuration with a low probability.

3.2 CRF Graph Structure

A conditional random field is a form of undirected graphical model that can be used to define the the joint probability distribution over a label sequences given a set of observation sequences to be labelled. Letting $\mathbf{X}$ and $\mathbf{Y}$ be jointly distributed random variables respectively ranging over observation sequences to be labelled and their corresponding label sequences, a conditional random field $(\mathbf{X}, \mathbf{Y})$ is a an undirected graphical model globally conditioned on $\mathbf{X}$, the observation sequence [31].

Formally, we define $G = (V, E)$ to be an an undirected graph such that $\mathbf{Y} = \{\mathbf{Y}_v \mid v \in V\}$. In other words, there is a node in the set V corresponding to each each of the random variables representing a component $\mathbf{Y}_v$ of the label sequence. The entire graph, and therefore the class of distributions associated with it, are considered to be conditioned upon $\mathbf{X}$ so the class of joint distributions associated with G will be of the form $p(\mathbf{y}_1, \ldots, \mathbf{y}_n | \mathbf{x})$ where $\mathbf{y}$ and $\mathbf{x}$ are particular realisations of label and observation sequences, respectively. If each random variable $\mathbf{Y}_v$ obeys the Markov property with respect to G – the probability of a random variable $\mathbf{Y}_v$ given $\mathbf{X}$ and all the other random variables $\mathbf{Y}_{\{u|u \neq v, \{u,v\} \in V\}}$

$$p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_u, u \neq v, \{u, v\} \in V) \tag{3.4}$$

is equal to the probability $p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_u, (u, v) \in E)$ of $\mathbf{Y}_v$ given $\mathbf{X}$ and those random variables corresponding to nodes neighbouring v in G

$$p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_u, (u, v) \in E), \tag{3.5}$$

then $(\mathbf{X}, \mathbf{Y})$ is a conditional random field.

In theory, the structure of the graph G may be arbitrary provided it represents the conditional independencies in the label sequences being modelled. However, when modelling sequences, the simplest and most common graph structure encountered is that in which the nodes corresponding to the elements of $\mathbf{Y}$ form a simple first-order chain structure, as illustrated in Figure 3.1. Note that only random variables representing elements of $\mathbf{Y}$ are part of the graph G because we wish to define a probability distribution of the form $p(\mathbf{y}|\mathbf{x})$. Additionally, the absence of any graphical structure between elements of $\mathbf{X}$ highlights the fact that we are merely conditioning upon observation sequences and so we do not make any independence assumptions about $\mathbf{X}$.

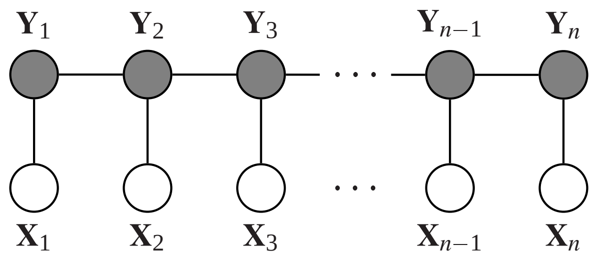

Figure 3.1: Graphical structure of the chain-structured case of CRFs for sequences. The variables corresponding to unshaded nodes are not generated by the model.

3.3 The Maximum Entropy Principle

Lafferty *et al.*'s [31] choice of potential functions for CRFs is based heavily on the principle of maximum entropy. Maximum entropy is a framework for estimating probability distributions from a set of training data that specifies that any assumptions made in constructing the distribution must by warranted by the data [23, 24, 47, 10]. Entropy [49] is a measure of uniformity of a probability distribution, or uncertainty. The conditional entropy $H(\mathbf{Y}|\mathbf{X})$ of a model distribution over label sequences given observation sequences $q(\mathbf{y}|\mathbf{x})$ is given by:

$$H(\mathbf{Y}|\mathbf{X}) = -\sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) \log q(\mathbf{y}|\mathbf{x}) \tag{3.6}$$

where $\tilde{p}(\mathbf{x},\mathbf{y})$ is the empirical distribution of the training data. This will be maximised when the distribution over label sequences $q(\mathbf{y}|\mathbf{x})$ is as uniform as possible. The principle of maximum entropy (maxent) asserts that the only probability distribution that can justifiably constructed from incomplete information, such as training data consisting of a set of constraints, is that which has maximum entropy subject to constraints representing what is known. Any other distribution would involve assumptions regarding unknown information which are entirely unwarranted. [23, 24]

In order to construct a model that accurately encodes all that we know about the training data, we need some method of representing this partial information. The most appropriate method of doing this is to encode information using positive valued *feature functions*. For example, suppose the training data contains the sentence

(3.7) The robot wheels Fred round.

and its corresponding sequence of chunk labels

(3.8) B-NP I-NP B-VP B-NP B-PP.

To express the information that the second POS tag is I-NP when the second word is "robot" and the first POS tag is B-BP, we define a feature

$$f(\mathbf{y}_1, \mathbf{y}_2, \mathbf{x}_2) = \begin{cases} 1 & \text{if } \mathbf{y}_2 = \text{I-NP}, \mathbf{x}_2 = \text{robot and } \mathbf{y}_1 = \text{B-NP} \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

A set of features of this sort can be used to summarise the important information contained within the training data. To ensure that the model agrees with the information encapsulated in the training data, the model distribution is constrained so that the expectation of each feature f with respect to the training data, given by

$$E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f] \triangleq \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) f(\mathbf{x},\mathbf{y}), \tag{3.10}$$

is equal to the expected value of that feature with respect to the model distribution:

$$E_q[f] \triangleq \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) q(\mathbf{y}|\mathbf{x}) f(\mathbf{x},\mathbf{y}) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f]. \tag{3.11}$$

The maximum entropy framework dictates that we must choose the distribution that satisfies all such constraints on feature values, while otherwise remaining as uniform as possible. Interestingly, this distribution is also the maximum likelihood Gibbs distribution, or the distribution that minimises the Kullback-Leibler divergence between the empirical distribution and the model distribution.

Identifying the maximum entropy distribution that satisfies the feature constraints imposed by the training data is a constrained optimisation problem. Berger *et al.* [10], Della Pietra *et al.* [41] and Ratnaparkhi [47] show that the parametric form of the maximum entropy constrained distribution is

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_k \lambda_k f_k(\mathbf{x}, \mathbf{y})\right) \tag{3.12}$$

where $Z(\mathbf{x})$ is a normalisation factor, and each λ_k is the Lagrangian multiplier associated with feature f_k. More intuitively, each parameter λ_k can also be considered to be a weighting of indicating the informativeness of feature f_k.

3.4 Potential Functions for CRFs

The maximum entropy framework provides significant justification for choosing the potential functions of a conditional random field such that the joint distribution over label sequences given observation sequences $p(\mathbf{y}|\mathbf{x})$ a parametric form similar to that given in Equation 3.12[1] This desideratum may be satisfied by defining each potential function as

$$\psi_{Y_c}(\mathbf{y}_c) = \exp\left(\sum_k \lambda_k f_k(c, \mathbf{y}|_c, \mathbf{x})\right), \tag{3.13}$$

where c is the set of indices of the nodes belonging to a maximal clique in the graph G of $\mathbf{Y}$, C is the set of all maximal cliques in G and $\psi_{Y_c}(\mathbf{y}_c)$ is a strictly positive, real valued potential function over the set of possible realisations of the

[1] In fact, it may be possible to choose potential functions for a CRF such that the distribution over label sequences forms a maximum entropy/minimum divergence (MEMD) model – a more general distribution for maximum entropy modelling, that incorporates a *reference distribution*. When this reference distribution is uniform, MEMD model identical to the maximum entropy model described in Section 3.3.

maximal clique $\mathbf{Y}_c$ in G. In addition to satisfying the requirement that potential functions must be positive, real-valued functions, this choice of potential function results in a joint distribution over the label sequence $\mathbf{Y}$ given $\mathbf{X}$ of the form

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{c\in C}\sum_{k} \lambda_k f_k(c, \mathbf{y}_c, \mathbf{x})\right) \tag{3.14}$$

where $Z(\mathbf{x})$ is a normalisation factor, given by

$$Z(\mathbf{x}) = \sum_{\mathbf{x},\mathbf{y}} \exp\left(\sum_{c\in C}\sum_{k} \lambda_k f_k(c, \mathbf{y}_c, \mathbf{x})\right). \tag{3.15}$$

In the case of the commonly used graph structure for modelling sequential data, a first-order chain $G = (V, E)$, the maximal cliques within G are its edges E. Therefore, for an edge $e = (i-1, i)$ the general form of Equation 3.13 can be expanded to

$$\psi_{Y_e}(\mathbf{y}_e) = \exp\left(\sum_{k} \lambda_k f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}) + \sum_{k} \mu_k g_k(\mathbf{y}_i, \mathbf{x})\right), \tag{3.16}$$

where each $f_k(i, \mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x})$ is some feature of the the entire observation sequence and the labels at positions i and $i-1$ in the corresponding label sequence, and each $g_k(i, \mathbf{y}_i, \mathbf{x})$ is a feature of label at position i and the observation sequence. This expansion enables the joint probability of a label sequence given an observation sequence to be written as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i}\sum_{k} \lambda_k f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}) + \sum_{i}\sum_{k} \mu_k g_k(\mathbf{y}_i, \mathbf{x})\right). \tag{3.17}$$

As a specific instance of this general situation, it is possible to create a CRF that takes on HMM-like properties by defining a single feature for each state-state pair (y', y) and state-observation pair (y, x) in the data set used to train the CRF:

$$f_{y',y}(\mathbf{y}_u, \mathbf{y}_v, \mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{y}_u = y' \text{ and } \mathbf{y}_v = y \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

$$g_{y,x}(\mathbf{y}_v, \mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{y}_v = y \text{ and } \mathbf{x}_v = x \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

In this situation, the parameters $\lambda_{y',y}$ and $\mu_{y,x}$ corresponding to these features are equivalent to the logarithm s of the HMM transition and emission probabilities $p(y'|y)$ and $p(x|y)$.

3.5 CRFs as a Solution to the Label Bias Problem

The global nature of the distribution defined by a conditional random field means that CRFs do not suffer from the label bias problem described in Section 2.3.2. To understand this, consider a CRF as a probabilistic automata with *unnormalised* weights associated with each state transition. The unnormalised nature of these weights means that transitions are not necessarily assigned equal importance. Therefore, any given state may amplify or dampen the probability mass passed on to its successor states, with the caveat that the final weighting given to any state sequence will be a properly defined probability due to the global normalisation factor.

3.6 Parameter Estimation for CRFs

There are two frameworks for inferring the structure of a model, including its parameters, from a set of training data – the *frequentist* and the *Bayesian*. These approaches each give rise a parameter estimation technique – *maximum likelihood estimation* (MLE) and *maximum a priori estimation* (MAP), respectively. Both techniques can be used to estimate the parameters of a CRF, however, current literature on CRFs [31] address only MLE parameter estimation.

3.6.1 Maximum Likelihood Parameter Estimation

The frequentist approach to parameter estimation is based on the use of some *estimator*, or function of the observed training data $\mathcal{D}$, to yield a single estimate of the parameter values Θ for the model in question. The most commonly used estimator within the frequentist community is the maximum likelihood estimator. Bayesians assume that all unknown quantities should be as random variables, allowing the probability of the training data for fixed theta to be viewed as a conditional probability distribution $p(\mathcal{D}|\Theta)$. However, the frequentist framework does not view Θ as a random variable. Instead, the the probability of the training data for fixed Θ is to be considered as a family of distributions over the training data indexed by Θ, written as $p_\theta(\mathcal{D})$ or (abusing notation) $p(\mathcal{D}|\Theta)$. This viewpoint permits the interpretation of $p(\mathcal{D}|\Theta)$ as a function of Θ for fixed data values, known as the *likelihood*:

$$L(\Theta) = p(\mathcal{D}|\Theta). \tag{3.20}$$

If we assume that the training data consists of a set of data points $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$ $\forall i = 1, \ldots, N$, each of which has been generated independently and identically from the joint empirical distribution $\tilde{p}(\mathbf{x}, \mathbf{y})$, then the likelihood of the training data according to some conditional model $p(\mathbf{y}|\mathbf{x}, \Theta)$ is

$$L(\Theta) = \prod_{\mathbf{x},\mathbf{y}} \log p(\mathbf{y}|\mathbf{x}, \Theta)^{\tilde{p}(\mathbf{x},\mathbf{y})}. \tag{3.21}$$

Two properties of the likelihood-function enable it to be used as a measure of the quality of a model $p(\mathbf{y}|\mathbf{x}, \Theta)$:

- $L(\Theta) \leq 0$ and
- $L(\Theta) = 0$ if and only if $\tilde{p}(\mathbf{x}, \mathbf{y}) > 0$ for all $p(\mathbf{y}|\mathbf{x}, \Theta) = 1$.

Maximum likelihood estimation therefore uses the likelihood function to rank possible values of Θ. Specifically, the MLE principle states that the value Θ should be chosen to maximise the likelihood function

$$\Theta_{ML} = \mathrm{argmax}_{\Theta} L(\Theta), \tag{3.22}$$

thus ensuring that data values that were observed in the training data are assigned a high probability. In other words, the parameters that maximise the likelihood function will result in a model that is as close to the empirical distribution as is possible given the model framework. The product in (3.22) may prove difficult to deal with and so, in general, the value of Θ is chosen to maximise the logarithm of the likelihood function (which involves a more manageable sum rather than a product)

$$\mathcal{L}(\Theta) = \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) \log p(\mathbf{y}|\mathbf{x},\Theta) \tag{3.23}$$

rather than the likelihood itself. This does not alter the value of Θ chosen, since the logarithm is a monotonic function.

3.6.2 Maximum Likelihood Estimation for CRFs

The maximum likelihood parameter estimation problem for a CRF that defines the probability distribution

$$p(\mathbf{y}|\mathbf{x},\Theta) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i=1}^{n+1} \sum_{k} \lambda_k f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}) + \sum_{i=1}^{n} \sum_{k} \mu_k g_k(\mathbf{y}_i,\mathbf{x}) \right) \tag{3.24}$$

is the task of estimating the parameters $\Theta = (\lambda_1,\lambda_2,\dots;\mu_1,\mu_2,\dots)$ from a set of training data points $\mathcal{D} = \{(\mathbf{y}^{(1)},\mathbf{x}^{(1)}),\dots,(\mathbf{y}^{(N)},\mathbf{x}^{(N)})\}$ independently and identically generated from the empirical distribution $\tilde{p}(\mathbf{x},\mathbf{y})$, such that the log-likelihood of the training data is maximised. Substituting in the value of $p(\mathbf{y}|\mathbf{x},\Theta)$ for a conditional random field (Equation 3.24) into the definition of the log-likelihood (Equation 3.21) gives

$$\mathcal{L}(\Theta) = \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) \left[\sum_{i=1}^{n+1} \boldsymbol{\lambda}\cdot\mathbf{f} + \sum_{i=1}^{n} \boldsymbol{\mu}\cdot\mathbf{g} \right] - \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \log Z, \tag{3.25}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the parameter vectors $(\lambda_1,\lambda_2,\dots)$ and $(\mu_1,\mu_2,\dots)$, respectively; $\mathbf{f}$ is the feature vector $(f_1(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}), f_2(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}),\dots)$; and $\mathbf{g}$ is the feature vector $(g_1(\mathbf{y}_i,\mathbf{x}), g_2(\mathbf{y}_i,\mathbf{x}),\dots)$.

From a numerical optimisation point of view, the log-likelihood function for a CRF is well-behaved – it is smooth and concave over the entire parameter space. The concave nature of log-likelihood function means that Θ can be chosen to be to the value for which the global maximum is obtained and the gradient or vector of partial derivatives with respect to each parameter in Θ is zero. Differentiating the log-likelihood function with respect to parameter λ_k gives

$$\begin{aligned}
\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_k} &= \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) \sum_{i=1}^{n} f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}) && (3.26)\\
&\quad - \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x},\Theta) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}) && (3.27)\\
&= E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k] - E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k]. && (3.28)
\end{aligned}$$

Note that setting this to zero gives the maximum entropy model constraint: The expectation of feature f_k with respect to the distribution $\tilde{p}(\mathbf{x})p(\mathbf{y}|\mathbf{x}|\Theta)$ must be equal to the expected value of f_k with respect to the empirical distribution.

Unfortunately, it is not generally possible to find the value of Θ that maximises the log-likelihood analytically – setting the gradient of the log-likelihood function to zero and solving for Θ does not always yield a closed form solution. Instead, the parameters that maximise the log-likelihood must be chosen using some form of iterative technique. Currently literature on CRFs [31] covers two algorithms for parameter estimation, based on *Improved Iterative Scaling* (IIS) [42] *Generalised Iterative Scaling* (GIS) [17].

3.6.3 Iterative Scaling

Iterative scaling is a method of iteratively refining a joint [17] or conditional [13, 48] model distribution by updating the parameters of a model using the update rule:

$$\lambda_k \leftarrow \lambda_k + \delta\lambda_k \tag{3.29}$$

where the update $\delta\lambda_k$ is chosen such that the new value of λ_k is closer to the maximum likelihood solution than the previous value. Lafferty *et al.* [31] propose two iterative scaling algorithms for estimating the maximum likelihood parameters of a conditional random field – one based on *Generalised Iterative Scaling* (GIS) [17] and one based on *Improved Iterative Scaling* (IIS) [42]. In this section, an overview of these algorithms is presented, and theoretical problems with each algorithm are highlighted.

The basic iterative scaling framework assumes that we have a model $p(\mathbf{y}|\mathbf{x},\Theta)$ parameterised by $\Theta = (\lambda_1,\lambda_2,\ldots;\mu_1,\mu_2,\ldots)$. The aim is to find a new set of parameters $\Theta+\Delta$ where $\Delta = (\delta\lambda_1,\delta\lambda_2,\ldots;\delta\mu_1,\delta\mu_2,\ldots)$ which result in a model with higher log-likelihood. The aim of iterative scaling is to identify a growth transformation that updates the parameters of our model so as to increase the log-likelihood by much as possible. The growth transformation can then be applied iteratively, until convergence is reached. For a CRF, the change in log-likelihood can be bounded from below by an auxiliary function $\mathcal{A}(\Theta,\Delta)$ which is defined as

$$\begin{aligned}\mathcal{A}(\Theta,\Delta) \triangleq \sum_{\mathbf{x},\mathbf{y}} &\left[\sum_{i=1}^{n+1}\sum_k \lambda_k f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}) + \sum_{i=1}^{n}\sum_k \mu_k g_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x})\right] \\ &+ 1 - \sum_{\mathbf{x}} \tilde{p}(\mathbf{x})p(\mathbf{y}|\mathbf{x},\Theta)\left[\sum_{i=1}^{n+1}\sum_k \left(\frac{f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x})}{T(\mathbf{x},\mathbf{y})}\right)\exp\left(\delta\lambda_k T(\mathbf{x},\mathbf{y})\right)\right. \\ &\left. + \sum_{i=1}^{n}\sum_k \left(\frac{g_k(\mathbf{y}_i,\mathbf{x})}{T(\mathbf{x},\mathbf{y})}\right)\exp\left(\delta\mu_k T(\mathbf{x},\mathbf{y})\right)\right]\end{aligned} \tag{3.30}$$

Since $\mathcal{L}(\Theta+\Delta) - \mathcal{L}(\Theta) \geq \mathcal{A}(\Theta,\Delta)$, finding the Δ that maximises $\mathcal{A}(\Theta,\Delta)$ will maximise the change in log-likelihood also. This gives rise to an iterative procedure for calculating the maximum likelihood parameter set Θ_{ML}:

- `Initialise each` λ_k
- `Until converged:`

  `Solve` $\frac{\partial\mathcal{A}(\Theta,\Delta)}{\partial\delta\lambda_k} = 0$ `for each parameter` λ_k

  `Update each parameter using` $\lambda_k \leftarrow \lambda_k + \delta\lambda_k$

where setting the partial derivative of $\mathcal{A}(\Theta,\Delta)$ with respect to parameter λ_k to zero yields the equation

$$E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k] \triangleq \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1},\mathbf{y}_i\mathbf{x}) \quad (3.31)$$

$$= \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x},\Theta) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1},\mathbf{y}_i\mathbf{x}) \exp\left(\delta\lambda_k T(\mathbf{x},\mathbf{y})\right), \quad (3.32)$$

from which the update parameters $\delta\lambda_k$ and $\delta\mu_k$ may be calculated. GIS and IIS are variants on this basic principle, that employ slightly different techniques for identifying the update parameters from Equation 3.32.

3.6.3.1 Generalised Iterative Scaling for CRFs

Generalised Iterative Scaling (GIS) is form of iterative scaling that follows the basic principle outlined in the previous section to update the parameters of a model so that the model converges towards one in which the expected value of each feature with respect to the model is equal to the expectation of that feature with respect to the empirical distribution of the training data:

$$E_{\text{model}}[f_k] = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k]. \quad (3.33)$$

To ensure that the updates result in the convergence of parameter values to the global optimum, GIS constrains the feature set such that for each event in the training data $T(\mathbf{x},\mathbf{y}) = C$, where C is some constant to ensure the update vector can be calculated analytically and $T(\mathbf{x},\mathbf{y})$ is defined as the sum of the active feature values for observation and label sequence pair $(\mathbf{x},\mathbf{y})$:

$$T(\mathbf{x},\mathbf{y}) \triangleq \sum_{i=1}^{n+1} \sum_k f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}) + \sum_{i=1}^{n} \sum_k g_k(\mathbf{y}_i,\mathbf{x}). \quad (3.34)$$

Satisfaction of this constraint requires the definition of a global[2] correction feature, given by:

$$s(\mathbf{x},\mathbf{y}) \triangleq C - \sum_{i=1}^{n+1} \sum_k f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x}) - \sum_{i=1}^{n} \sum_k g_k(\mathbf{y}_i,\mathbf{x}), \quad (3.35)$$

[2]This feature is not specific to any particular edge or vertex.

where C is the maximum value of $T(\mathbf{x},\mathbf{y})$ for all $\mathbf{y}$ and $\mathbf{x}$ in the training data. Note that if the features used are binary valued, then $T(\mathbf{x},\mathbf{y})$ will simply be the number of active features for that event, and C will simply be the maximum possible number of active features. The addition of this feature to the feature set ensures that $T(\mathbf{x},\mathbf{y}) = C$ as desired. In general, adding new features to the feature set will alter the model distribution. However, in this case, the new feature is entirely dependent on the existing features in the feature set and therefore adds no additional information or constraints on the model. The trained model will therefore be unchanged by the definition of the correction feature.

Assuming the features chosen for a CRF sum to the constant C for all events, Lafferty *et al.* [31] assert that Equation 3.32 can be solved analytically as follows. Taking logs of both sides of Equation 3.32

$$\begin{aligned}\log E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k] &= \log\left[\sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x})p(\mathbf{y}|\mathbf{x},\Theta)\sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x})\exp(\delta\lambda_k C)\right] &(3.36)\\ &= \log E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k] + \delta_k C &(3.37)\end{aligned}$$

yields the update

$$\delta_k = \frac{1}{C}\log\left[\frac{E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k]}{E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k]}\right]. \tag{3.38}$$

The rate of convergence of the GIS algorithm is governed by the step size used for the updates which, in turn, is dictated by the magnitude of constant C: Large values of C give rise to a small step size and slow the rate of convergence, while smaller C values yield a larger step size and hence a faster rate of convergence.

In fact, careful analysis reveals that this algorithm is intractable. Specifically, the GIS algorithm is dependent on the addition of a global correction feature $s(\mathbf{y},\mathbf{x})$ to ensure the the active features values for each $(\mathbf{x},\mathbf{y})$ pair sum to a constant. Once added to the feature set, this correction feature is treated identically to all other features and its parameters are estimated using the update given in Equation 3.38. For any feature f_k, calculating this update requires computation of the expectation of f_k with respect to the product of the model distribution and the marginal distribution over observation sequences:

$$E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k] \triangleq \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x},\Theta) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1}, \mathbf{y}_i \mathbf{x}) \exp\left(\delta\lambda_k T(\mathbf{x},\mathbf{y})\right) \tag{3.39}$$

In the general case, this is intractable, since it requires the summation over all possible label sequences – a task that will be exponential in the number of possible labels. Lafferty *et al.* note that it is possible to get round this intractability for edge and vertex features, using a dynamic programming technique described here in Section 4.3.3. However, they neglect to mention that such a dynamic programming technique is *not* possible for global features and so calculating the expectation of the correction feature $s(\mathbf{x},\mathbf{y})$ with respect to the model distribution, and hence identifying the parameter update for the correction feature, is intractable. For the GIS algorithm to be applied correctly, the correction feature must be treated as any other feature in the model. Therefore, the intractability outlined here means it is not possible to use Lafferty *et al.*'s GIS-based algorithm for estimating the parameters of CRF.

3.6.3.2 Improved Iterative Scaling for CRFs

Improved Iterative Scaling [42] is a variant of GIS that eliminates the need for a correction feature and thus allows faster convergence than the basic GIS algorithm. Rather than attempting to solve Equation 3.32 for each parameter analytically, IIS is based on the observation that Equation 3.32 is a polynomial in $\exp(\delta\lambda_k)$ and can therefore be solved for $\delta\lambda_k$ using a simple technique such as the Newton-Raphson method.

To represent Equation 3.32 as a polynomial in $\exp(\delta\lambda_k)$ that may be tractably solved, Lafferty *et al.* use the mean field approximation:

$$T(\mathbf{x},\mathbf{y}) \approx T(\mathbf{x}) \triangleq \max_{\mathbf{y}} T(\mathbf{x},\mathbf{y}) \tag{3.40}$$

In other words, they approximate the the sum of the active feature values for each observation and label sequence pair $(\mathbf{x},\mathbf{y})$ with the maximum possible sum of observation features for that observation sequence $\mathbf{x}$. This enables Equation 3.32 to be rewritten as:

$$E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k] = \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x},\Theta) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1},\mathbf{y}_i\mathbf{x}) \exp\left(\delta\lambda_k T(\mathbf{x})\right) \tag{3.41}$$

This equation may be expressed as a polynomial in $\exp(\delta\lambda_k)$ by observing that any set of $(\mathbf{x},\mathbf{y})$ pairs can be partitioned into T_{max} non-overlapping subsets, where $T_{max} = \max T(\mathbf{x})$, according to their $T(\mathbf{x})$ values. Rewriting Equation 3.41 so that the sum over $(\mathbf{x},\mathbf{y})$ values are split according to $T(\mathbf{x})$ gives

$$\sum_{m=0}^{T_{max}} \sum_{\{\mathbf{x},\mathbf{y}|T(\mathbf{x})=m\}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x},\Theta) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1},\mathbf{y}_i\mathbf{x}) \left[\exp\left(\delta\lambda_k\right)\right]^m. \tag{3.42}$$

Assuming the standard interpretation of the delta function $\delta(\cdot)$, we now define $a_{k,m}$ to be the expectation of f_k given that $T(\mathbf{x}) = m$:

$$a_{k,m} = \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x},\Theta) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1},\mathbf{y}_i\mathbf{x}) \delta(m, T(\mathbf{x})) \tag{3.43}$$

which enables Equation 3.42 to be expressed as a polynomial in $\exp(\delta\lambda_k)$:

$$\sum_{m=0}^{T_{max}} a_{m,k} \exp\left(\delta\lambda_k\right)^m = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k]. \tag{3.44}$$

This polynomial may now being solved using the Newton-Raphson method.

3.6.4 Efficiency of IIS for CRFs

Although the mean field approximation required to make this IIS variant tractable merely serves to modify the lower bound on the change in log-likelihood, $\mathcal{A}(\Theta,\Delta)$, it is possible that this changed lower bound may result in slow convergence. Sure enough, Lafferty *et al.*'s experimental results involving CRFs for POS tagging indicate that convergence of their IIS variant is very slow. When estimating parameters for an exponential distribution of the form given in Equation 3.12, it is usual to initialise parameters so that the training commences with the model being the uniform distribution. However, Lafferty *et* *al.* were unable to train a CRF initialised to have a uniform distribution to convergence in 2000 iterations. Instead, they were only able to efficiently train CRFs by initialising the CRF parameter vector to the parameters of a MEMM trained to convergence in 100 iterations. When this set of initial parameters was used, the IIS-based algorithm for CRFs converged in 1000 iterations.

Although using trained MEMM parameters enabled Lafferty *et al.* to train CRFs in a reasonable time, this is not an ideal solution since it depends on the availability of trained MEMMs. In addition to this, recent work of Bancarz and Osborne [4] indicates that IIS can yield multiple globally optimal models that yield wildly differing performance levels, depending on the initial parameter vector. This observation may mean that the decision to start CRF training using the trained parameters of an MEMM is in fact biasing the performance of CRFs reported in the current literature.

3.7 Chapter Summary

This chapter introduced CRFs as an undirected model for labelling and segmenting data. The theoretical basis for the functional form of the distribution defined by a CRF was outlined, and an overview of CRFs as a solution to the label bias problem was presented. Finally, the parameter estimation algorithms covered in current CRF literature were described and their theoretical and practical limitations discussed.

Chapter 4

Numerical Optimisation for CRF Parameter Estimation

The primary justification behind the use of iterative scaling algorithms is considerable ease of implementation and the fact that, unlike other optimisation techniques, the gradient of the function being optimised (in this case the log-likelihood function) need not be calculated. Instead, the only computations required are the required are those necessary to evaluate the expectation of each feature value with respect to the new model distribution $E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k]$. This is highly advantageous for model distributions in which calculation of the gradient vector is computationally expensive, however, in the case of conditional random fields and other maximum entropy models [10], each element of the gradient vector is given by

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_k} = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f] - E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k]. \tag{4.1}$$

and so there is likely to be little computational advantage to using iterative scaling rather than techniques that utilise the gradient directly.

To investigate whether there is any computational advantage to using iterative scaling algorithms when training conditional maximum entropy models, Malouf [32] compared the performance of a number of algorithms for estimating parameters of conditional maximum entropy models using a range of NLP problems. Interestingly, Malouf observed that iterative scaling algorithms performed poorly in comparison with first- and second- order optimisation methods and, for all the NLP problems considered, a *limited memory variable metric* algorithm [7] performed substantially better than any of the other algorithms.

Malouf's findings for conditional maximum entropy models lend significant weight to the hypothesis that first- and second-order numerical optimisation techniques would result in good performance for CRF parameter estimation also. Lafferty *et al.*'s [31] choice of potential functions gives

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i=1}^{n+1}\sum_{k}\lambda_k f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}) + \sum_{i=1}^{n}\sum_{k}\mu_k g_k(\mathbf{y}_i, \mathbf{x})\right) \tag{4.2}$$

as the conditional distribution over label sequences given an observation sequence defined by a CRF. The functional form of this equation is very similar to that of a non-sequential conditional maximum entropy model:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{k}\lambda_k f_k(\mathbf{x}, \mathbf{y})\right). \tag{4.3}$$

Given this functional similarity and Malouf's observations that numerical techniques perform better than iterative scaling for parameter estimation of conditional maximum entropy models, I hypothesise that it is likely that such numerical techniques would also improve parameter estimation performance for conditional random fields. To experimentally investigate this hypothesis, three numerical optimisation techniques that Malouf found to perform well for estimating the parameters of conditional maximum entropy models were compared with with Lafferty *et al.*'s IIS-variant for CRF parameter estimation.

In this chapter, the first- and second-order numerical optimisation techniques applied to CRF parameter estimation in this thesis is are described, and implementation details are discussed. Finally, the performance of the numerical techniques implemented are compared with that of Lafferty *et al.*'s IIS-based algorithm on a toy problem based on the NLP task of shallow parsing.

4.1 First-order Numerical Optimisation Techniques

The numerical optimisation techniques applied to CRF parameter estimation in the work described in this thesis fall into two categories – first- and second-order techniques. First-order techniques use the information contained within the gradient vector $G(\Theta)$ of the function being optimised to repeatedly shift estimates of the parameters towards the point at which the gradient is zero and the function is at an optimum. Here, two first-order methods were applied to the task of CRF parameter estimation, both of which are variants of the *non-linear conjugate gradient* algorithm.

4.1.1 Non-Linear Conjugate Gradient

Unlike steepest ascent methods, which consider the same search direction several times when maximising a function, conjugate direction methods generate a set of non-zero vectors known as the *conjugate set* and successively maximise the function along each of these directions. *Non-linear conjugate gradient* methods are a particular form of conjugate direction technique in which each conjugate vector, or search direction, is generated from the previous search direction alone, rather than all previous elements of the conjugate set. Specifically, each successive search direction $\mathbf{p}_j$ is selected to be a linear combination of the steepest ascent direction, or gradient of the function to be maximised, and the previous search direction $\mathbf{p}_{j-1}$. Each iteration of the conjugate gradient update algorithm shifts the parameters of the function to be maximised in the direction of the current conjugate vector $\mathbf{p}_j$ using the update rule

$$\lambda_k^{(j+1)} = \lambda_k^j + \alpha^{(j)}\mathbf{p}_j \tag{4.4}$$

where $\alpha^{(j)}$ is the optimal step size, selected using an approximate line search.

There are several conjugate gradient methods that are appropriate for maximising a general convex function such as the log-likelihood of $\tilde{p}(\mathbf{x},\mathbf{y})$ according to a conditional model of the form given in Equation 4.2. Here, we consider the *Fletcher-Reeves* and the *Polak-Ribière-Positive* algorithms. These algorithms are theoretically equivalent, but may exhibit different numerical properties due to different methods for choosing the search direction and step size. A detailed discussion of both algorithms may be found in [37].

4.2 Second-Order Numerical Optimisation Techniques

Second-order optimisation techniques, such as Newton's method, improve over first-order techniques such as conjugate gradient, by augmenting the gradient values used in calculating the parameter updates with information regarding the curvature, or second order derivatives, of the function to be optimised.

The general second-order update rule is calculated from the second-order Taylor series approximation of $L(\Theta+\Delta)$, given by:

$$L(\Theta+\Delta) \approx L(\Theta) + \Delta^T G(\Theta) + \frac{1}{2}\Delta^T H(\Theta)\Delta \tag{4.5}$$

where $H(\Theta)$ is the matrix of second partial derivatives with respect to Θ of the log-likelihood function, or the Hessian matrix. Setting the derivative of this approximation to zero, results in the update rule:

$$\Delta^{(k)} = H^{-1}(\Theta^{(k)})G(\Theta^{(k)}). \tag{4.6}$$

Although this update rule results in very fast convergence, computation of the inverse of the Hessian matrix may be prohibitively expensive for large-scale problems such as those encountered in NLP. Therefore, use of second-order methods that make direct use of the Hessian when estimating the parameters of large models is highly impractical.

Variable-metric or *quasi-Newton* methods are a form of second-order technique, similar to Newton's method, but rather than explicitly calculating the inverse Hessian they rely entirely on information contained within the gradient objective function. At each iteration, variable-metric methods avoid the use of second-derivatives and, instead, build a model of the Hessian by measuring the change in gradient. This local approximation of the Hessian is empirically found to be sufficiently good that variable-metric methods often exhibit super-linear convergence.

The basic principle behind variable-metric methods is to replace the Hessian matrix in the second order Taylor approximation of $L(\Theta+\Delta)$ with $B(\Theta)$, a symmetric positive definite matrix that approximates the Hessian. This results in the revised update rule:

$$\Delta^{(k)} = B^{-1}(\Theta^{(k)})G(\Theta^{(k)}). \tag{4.7}$$

Every iteration, $B(\Theta)^{-1}$ is updated to reflect the parameter changes that result from the previous iteration. However, rather than calculating $B(\Theta)^{-1}$ afresh, it is simply updated to account for the curvature measured during the previous iteration – a task which relies only on the current gradient $G(\Theta^{(k)})$ and the gradient from the previous step $G(\Theta^{(k-1)})$:

$$B(\Theta^{(k)})^{-1}(G(\Theta^{(k)}) - G(\Theta^{(k-1)}) = \Delta^{(k-1)}. \tag{4.8}$$

Approximating the Hessian matrix by $B(\Theta)$ enables variable-metric methods to exhibit improved convergence over the traditional Newton method.

4.2.1 Limited-Memory Variable-Metric Methods

Despite the computational improvements obtained by approximating the Hessian by $B(\Theta)$, the approximate Hessian and its inverse prove to be sufficiently dense that their storage is infeasible for large-scale problems. In the case of NLP tasks, the number n of parameters to be estimated may be millions, yet storage of an $n \times n$ dense matrix for such tasks is currently impossible. However, it is possible to modify variable-metric methods to use implicit representations of Hessian approximations that only require storage of a small number of vectors of length n, where n is the number of parameters to be estimated. Such methods are called *limited-memory variable-metric* methods.

Limited-memory variable-metric methods make use of the fact that, for iteration k, calculation of the product:

$$B(\Theta^{(k)})^{-1}G(\Theta^{(k)}. \tag{4.9}$$

may be performed using a sequence of inner products and vector summations involving $G(\Theta^{(k-1)}$, and the set of pairs:

$$\{\Delta^{(i)}, G(\Theta^{(i)}) - G(\Theta^{(i-1)})|i = k-m, \ldots, k-1\}. \tag{4.10}$$

At each iteration, the oldest pair in the set $\{\Delta^{(i)}, G(\Theta^{(i)}) - G(\Theta^{(i-1)})\}$ is, therefore, deleted and replaced with the pair obtained from the current step, ensuring that only most recent m pairs are stored at any given point in time. In practice, values of m between 3 and 20 are sufficient to obtain good performance, and so the reduction in storage space over variable-metric methods is significant.

The large-scale nature of the problems for which CRFs may prove useful means that use of standard variable metric methods for parameter estimation is infeasible. However, the space reduction exhibited by limited memory variable metric methods results in storage requirements that are practical, even for very large-scale tasks such as those found in NLP. For this reason, we do not attempt to apply standard variable metric methods to the task of CRF parameter estimation, but consider limited memory variable metric methods instead.

4.3 Implementation

To enable efficient performance, PETSc (the Portable, Extensible Toolkit for Scientific Computation) [2, 3] was used as the implementation basis. PETSc is a software library that assists development of scientific applications modelled by partial differential equations by providing a variety of data structures and routines for storing, manipulating and visualising very large sparse matrices. All operations required for training CRFs may be expressed in terms of matrix calculations. Framing parameter estimation in this way enables us take advantage of utilities offered by the PETSc framework and therefore improve efficiency.

4.3.1 Representation of Training Data

As we stated in Section 3.4, every potential function in a chain structured CRF is function of the entire observation sequence being labelled, and the labels of two adjacent elements in the corresponding label sequence:

$$\psi_{Y_e}(\mathbf{y}_e) = \exp\left(\sum_k \lambda_k f_k(i, i-1, \mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}) + \sum_k \mu_k g_k(i, \mathbf{y}_i, \mathbf{x})\right) \tag{4.11}$$

This observation, combined with the nature of the dynamic programming techniques used by Lafferty *et al.* to enable efficient calculation of feature expectations (outlined in Section 4.3.3), means that the most appropriate method of representing the data used for training a chain structured CRF is to define a sparse matrix F, with rows corresponding to particular $(\mathbf{x}^j, \mathbf{y}_i^j, \mathbf{y}_{i-1}^j)$ tuples and columns to f and g features.

4.3.2 Model Probability as Matrix Calculations

Lafferty *et al.* observe that for a chain-structured CRF in which each label sequence is augmented with a start $\mathbf{y}_0$ and stop $\mathbf{y}_{n+1}$ state, the conditional probability $p(\mathbf{y}|\mathbf{x})$ of a label sequence $\mathbf{y}$ given an observation sequence $\mathbf{x}$ may be expressed in terms of matrices. Specifically, if $\mathcal{Y}$ is the alphabet from which labels are drawn, then we define $M_i(\mathbf{x})$ to be a $|\mathcal{Y} \times \mathcal{Y}|$ matrix random variable, where each element $M_i(y', y|\mathbf{x})$ of $M_i(\mathbf{x})$ is given by:

$$M_i(y', y|\mathbf{x}) = \exp\left(\sum_k f_k(\mathbf{y}_{i-1} = y', \mathbf{y}_i = y, \mathbf{x}) + \sum_k g_k(\mathbf{y}_i = y, \mathbf{x})\right) \tag{4.12}$$

Recalling the view of a CRF as a finite state model with unnormalised probabilities mentioned in Section 3.5, each matrix $M_i(\mathbf{x})$ can be considered to represent the weights on each arc of the model at time step i. This matrix formulation means that the unnormalised conditional probability $p^*(\mathbf{y}|\mathbf{x})$ of the label sequence can be expressed as the product of the appropriate entries drawn from the $n+1$ matrices for this sequence:

$$p^*(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n+1} M_i(\mathbf{y}_{i-1}, \mathbf{y}_i|\mathbf{x}) \tag{4.13}$$

The normalisation factor $Z(\mathbf{x})$, which depends only on the observation sequence $\mathbf{x}$, may be calculated from the set of $M_i(\mathbf{x})$ matrices by using *closed semirings*, an algebraic structure that results in a general framework for solving path problems within graphs [16]. Specifically, $Z(\mathbf{x})$ is given by the **(start,stop)** entry of the product of the $M_i(\mathbf{x})$ matrices:

$$Z(\mathbf{x}) = \left[\prod_{i=1}^{n+1} M_i(\mathbf{x})\right]_{\mathbf{start,stop}} . \tag{4.14}$$

4.3.3 Dynamic Programming for Feature Expectations

The IIS-based training algorithm for CRFs and the numerical estimation techniques described in Sections 4.1 and 4.2 all involve calculation of feature expectations with respect to both the empirical distribution $\tilde{p}(\mathbf{x}, \mathbf{y})$, given by

$$E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k] = \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}, \mathbf{y}) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}) \tag{4.15}$$

for an edge feature $f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x})$, and the feature expectations with respect to the product of the model distribution $p(\mathbf{y}|\mathbf{x})$ and the marginal $p(\mathbf{x})$:

$$E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k] = \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x}) \sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}). \tag{4.16}$$

for an edge feature $f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x})$. However, calculating the expectation of each feature with respect to the CRF model distribution in a naïve fashion is intractable due to the required sum over all possible label sequences. For instance, if a given observation sequence $\mathbf{x}$ has a length n then there will be

$n^{|\mathcal{Y}|}$ possible label sequences for this observation sequence. Clearly evaluating $\tilde{p}(\mathbf{x})p(\mathbf{y}|\mathbf{x})\sum_{i=1}^{n+1} f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x})$ for all of these possible label sequences is infeasible. Fortunately, Lafferty *et al.* propose a dynamic programming method for avoiding this intractability when calculating $E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k]$.

Lafferty *et al.*'s dynamic programming technique makes use of the fact that the model probability $p(\mathbf{y}|\mathbf{x})$ may be expressed in terms of the matrices $M_i(\mathbf{x})$. Starting with the definition of the expectation of an edge feature $f_k(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x})$, we note that Equation 4.16 may be rewritten as:

$$\begin{aligned} E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k] = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{n+1} \sum_{y',y} & f_k(\mathbf{y}_{i-1} = y', \mathbf{y}_i = y, \mathbf{x}) \\ & \times \frac{\alpha_i(y'|\mathbf{x}) M_i(y', y|\mathbf{x}) \beta_{i+1}(y|\mathbf{x})}{Z(\mathbf{x})} \end{aligned} \tag{4.17}$$

where $\alpha_i(\mathbf{x})$ and $\beta_i(\mathbf{x})$ are forward and backward vectors, and are defined respectively as by the base cases:

$$\alpha_0(y|\mathbf{x}) = \begin{cases} 1 & \text{if } y = \textbf{start} \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

and

$$\beta_{n+1}(y|\mathbf{x}) = \begin{cases} 1 & \text{if } y = \textbf{stop} \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

and the recurrence relations:

$$\alpha_i(\mathbf{x})^T = \alpha_{i-1}(\mathbf{x})^T M_i(\mathbf{x}) \tag{4.20}$$

and

$$\beta_i(\mathbf{x}) = M_{i+1}(\mathbf{x}) \beta_{i+1}(\mathbf{x}) \tag{4.21}$$

Similarly, Lafferty *et al.* state that the expectation of a vertex feature $g_k(\mathbf{y}_i, \mathbf{x})$ with respect to the model distribution may be expressed as:

$$\begin{aligned} E_{p(\mathbf{y}|\mathbf{x},\Theta)}[g_k] = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{n+1} \sum_{y} & g_k(\mathbf{y}_i = y, \mathbf{x}) \\ & \times \frac{\alpha_i(y|\mathbf{x}) \beta_i(y|\mathbf{x})}{Z(\mathbf{x})} \end{aligned} \tag{4.22}$$

However, the representation of the training data used in this project, in which each row of the training data matrix corresponds to a particular $(\mathbf{x}, \mathbf{y}_{i-1}, \mathbf{y}_i)$ tuple, means that in fact it is more efficient to represent the expectation of an edge feature g_k as

$$E_{p(\mathbf{y}|\mathbf{x},\Theta)}[g_k] = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{n+1} \sum_{y',y} g_k(\mathbf{y}_i = y, \mathbf{x}) \times \frac{\alpha_i(y'|\mathbf{x}) M_i(y', y|\mathbf{x}) \beta_{i+1}(y|\mathbf{x})}{Z(\mathbf{x})} \tag{4.23}$$

Using these expressions for feature expectations means that rather than calculating the entire model distribution $p(\mathbf{y}_1, \ldots, \mathbf{y}_n|x)$ we need only calculate each possible $p(\mathbf{y}_{i-1}, \mathbf{y}_i|\mathbf{x})$ value using the dynamic programming technique described above. In addition to this, we observe that rewriting the definitions for $E_{\tilde{p}(\mathbf{x},\mathbf{y})}[f_k]$ and $E_{p(\mathbf{y}|\mathbf{x},\Theta)}[f_k]$ in a similar fashion means that all the information we need from the empirical distribution $\tilde{p}(\mathbf{x}, \mathbf{y})$ may be stored using two vectors – $p(\mathbf{y}_{i-1}, \mathbf{y}_i|\mathbf{x})$ for all possible $(\mathbf{y}_{i-1}, \mathbf{y}_i)$ values and p $(\mathbf{x})$, the marginal distribution of observation sequences in the training data.

All information required for estimating parameters, both by the IIS-based algorithm and the numerical techniques, is contained within the distribution vectors $\tilde{p}(\mathbf{y}_1, \ldots, \mathbf{y}_n|x)$ and $\tilde{p}(\mathbf{x})$, the $M_i(\mathbf{x})$ matrices, the forward and backwards vectors, and the training data matrix. Using these data structures, all computations required for parameter estimation may be performed using matrix calculations which may be efficiently performed using the PETSc library.

4.3.4 Optimisation Techniques

The IIS-based parameter estimation algorithm was implemented in C++ using data structures and routines provided by PETSc. The other estimation algorithms were also implemented in C++ and made use of TAO (the Toolkit for Advanced Optimisation) [6, 5] for the implementation of the numerical optimisation methods. TAO is a library, built on top of PETSc, designed to assist with the implementation of tasks that involve non-linear optimisation problems. TAO provides routines for line searches and convergence tests as well as implementations of standard optimisation algorithms, such as conjugate gradient and variable metric methods.

4.3.5 Stopping Criterion

To ensure fair comparison, the same stopping criterion was used for all parameter estimation algorithms. Malouf [32] found that judging convergence to be reached when the relative change in log-likelihood between iterations:

$$\frac{L(\Theta^{(k)}) - L(\Theta^{(k-1)})}{L(\Theta^{k})} \tag{4.24}$$

fell below a predetermined threshold of 10^{-7} provided a good basis for comparing parameter estimation techniques for conditional maximum entropy models. Since this criterion was used by Malouf to successfully compare parameter estimation algorithms for conditional maximum entropy models, it would seem an appropriate choice to use for CRFs which have a similar functional form to the models investigated by Malouf.

4.4 Experiments

To perform a comparison of the numerical optimisation techniques described in Sections 4.1 and 4.2 and Lafferty *et al.*'s IIS-based algorithm, the implementation described in the previous section was applied to a well-known sequential data labelling task found in statistical natural language processing – text chunking or shallow parsing [28]. This task was chosen because it is representative of the sorts of sequential data found in NLP problems, and has been previously studied using many different methods ranging from adaptations of non-sequential techniques maximum entropy [38, 29] to HMM- and FSA-based methods [43, 51, 34].

4.4.1 Shallow Parsing

Shallow parsing is the task of diving text into non overlapping chunks or phrases such that syntactically related words are grouped together. For example, consider the labelled sentence:

- [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September] [O .].

Text chunking of this nature can be very useful – the resultant chunks can be used to assist the process of full parsing, as originally proposed by Abney [1]. Here, the CoNLL-2000 shared task (a supervised learning problem involving annotating part-of-speech (POS) tagged sentences with non-overlapping phrases) is used. An overview of this task and detailed description of the chunk types may be found in Tjong Kim Sang and Buchholz [28]. Each word in a given sentence is labelled with a chunk label which can be one of three forms – an *O* label (indicating that the word is not part of any phrase), a *B* label (indicating that the word is the first of a particular phrase) or an *I* label (indicating that the word is inside a phrase). The *B* and *I* labels are further subdivided to indicate what type of minor phrasal category a given word word is part of, giving rise to a total of 22 chunks labels (see Table 4.1).

The corpus from which the sentences I used were drawn consists of material from the Penn Treebank II (WSJ sections 15-18 and 20). The material has been split into sentences (or sequences) and each sentence split into tokens (words, punctuation). Tokens are additionally annotated with a part-of-speech tag (generated by a Brill tagger).

4.4.2 Features

Previous work involving log-linear models for text chunking [38, 29, 39] has resulted in the identification of a set of informative features that capture salient aspects of the tagging task. Specifically, when performing text chunking, there

Count	Chunk Type
55081	NP (noun phrase)
21467	VP (very phrase)
21281	PP (prepositional phrase)
4227	ADVP (adverb phrase)
2207	SBAR (subordinated clause)
2060	ADJP (adjective phrase)
556	PRT (particles)
56	CONJP (conjunction phrase)
31	INTJ (interjection)
10	LST (list marker)
2	UCP (unlike coordinated phrase)

Table 4.1: Number of chunks per phrase type in training data (211727 tokens). [28]

is a tradeoff between context and accuracy. The more context (words and POS tags) used, the greater the accuracy of the model will be, however more context vastly increases the number and sparsity of features. Osborne [39] found that good results can be achieved by considering the POS tag of the current word t_i , the POS tags of the next two words t_{i+1}, t_{i+2} and the last four letters of the current word w_i. To encode context of this nature as CRF features, a set of features is defined for each previous and current state pair (y', y) and each state-observation pair (y, x). Each state-state feature is of the form:

$$f_{<y',y>}(\mathbf{y}_{i-1}, \mathbf{y}_i) = \begin{cases} 1 & \text{if label } \mathbf{y}_{i=1} = y' \text{ and label } \mathbf{y}_i = y \\ 0 & \text{otherwise.} \end{cases} \tag{4.25}$$

and each state-observation feature is defined as

$$g_{<y,b>}(\mathbf{y}_i, \mathbf{x}) = \begin{cases} 1 & \text{if } b(\mathbf{x}_i) \text{ and label } \mathbf{y}_i = y \\ 0 & \text{otherwise.} \end{cases} \tag{4.26}$$

where $b(\mathbf{x}_i)$ is a boolean predicate regarding the nature of the i^{th} word and its context $t_i, t_{i+1}, t_{i+2}, w_i$. An example of such a boolean predicate is

$$b(\mathbf{x}_i) = \begin{cases} 1 & \text{if the POS tag of the current word is } NN \\ 0 & \text{otherwise.} \end{cases} \tag{4.27}$$

Unfortunately, implementation problems (outlined below) meant that it was not possible to use the context described above when CRF parameter estimation techniques. Instead a greatly reduced context consisting of the current POS tag t_i and the last four letters of the current word w_i were used instead. The method for constructing CRF features from this limited context is identical to the process described above.

4.4.3 Performance of Parameter Estimation Algorithms

My original intention was to compare the performance of Lafferty *et al.*'s IIS variant and the parameter estimation algorithms described in Sections 4.1 and 4.2 using the training data from the CoNLL-2000 shared task to train the CRF models. However problems with memory leaks in the CRF implementation, which appeared to be occurring in the PETSc library routines for matrix creation, manipulation and destruction, meant that the training process used increasingly large amounts of memory as it ran, even though no additional memory needed to be allocated. These memory leaks severely limited the size of data set and amount of context that could feasibly be used when training the CRF models. I therefore chose to restrict the training data to a toy problem consisting of five sentences drawn from the CoNLL-2000 shared task corpus (see Table 4.2). These sentences are not special in any way, and the accuracy of any model trained on these sentences is not indicative of the accuracy that would be achieved, were the models trained on a much larger data set. However, these five sentences do provide a common, albeit tiny, data set that may be used to compare the relative performance of the parameter estimation algorithms in question. To provide a more detailed and thorough comparison of the parameter estimation algorithms, all experiments used to compare the training methods will be repeated with the full CoNLL-2000 training data set once the memory leaks mentioned above have been eliminated[1]

Labels	Contexts	Features	Non-zeros	$(\mathbf{y}_{i-1} = y', \mathbf{y}_i = y, \mathbf{x})$ events
10	5	69	4572	23760[2]

Table 4.2: Characteristics of the toy dataset used to compare algorithm performance.

The results of applying each of the parameter estimation algorithms to the five sentences drawn from the CoNLL-2000 training data set are summarised in Table 4.3. For each training algorithm, this table indicates the number of iterations required to train the model to convergence, the number of log-likelihood and gradient evaluations required (some optimisation techniques require multiple evaluations per iteration) and the total elapsed time in seconds.[3] Despite

Method	Iterations	LL Evaluations	Time (s)
IIS	>150	>150	>188.65
Conjugate gradient (FR)	19	99	124.67
Conjugate gradient (PRP)	27	140	176.55
Limited memory variable metric	22	22	29.72

Table 4.3: Results of comparison.

the highly trivial nature of the data set used, these results clearly highlight performance differences between the algorithms. Lafferty *et al.*'s IIS-based algorithm is the slowest of all of the techniques, taking more than n iterations and n seconds to reach a relative change in log-likelihood of 10^{-7} between iterations. The conjugate gradient methods are both faster than the IIS variant, requiring fewer iterations and log-likelihood calculations. Of the

[1]The updated results will be at `http://www.cogsci.ed.ac.uk/~osborne/msc-projects/wallach.ps.gz`.

[3]All experiments were performed using a single CPU of a dual processor Intel(R) Xeon(TM) CPU 1700MHz machine with 2GB of RAM.

two conjugate gradient-based methods, Fletcher-Reeves exhibits faster convergence. The fastest technique out of all the methods investigated was the limited memory variable metric algorithm [7]. This technique trained the CRF to convergence in 29.72 seconds using 22 iterations and performed only 22 log-likelihood and gradient calculations. Even though the results obtained in these experiments seem to be highly encouraging, to fully confirm these results, a comparison of the four techniques will need to be rerun on a much larger, non-trivial dataset once issues regarding memory leaks in the CRF implementation have been sorted out.

Most importantly, these results echo Malouf's [32] findings for conditional maximum entropy models, even though the data set used here was not in any way representative of the kind of dataset encountered in NLP classification tasks. This re-confirmation of Malouf's experimental observations using a different theoretical framework has significant implications not only for training of CRFs, but for training of other maximum entropy and minimum divergence models. In particular, Malouf's findings combined with the work in this thesis show, using two independent experimental scenarios involving two different log-linear models, that general gradient-based numerical optimisation techniques outperform iterative scaling by a considerable margin both in terms of log-likelihood evaluations and total elapsed time. Additionally, in both Malouf's experiments and the work outlined in this thesis, a limited memory variable metric method that takes into account the curvature of the log-likelihood function when calculating updates results in significantly faster convergence than the first-order techniques considered.

4.5 Chapter Summary

In this chapter, a number of first- and second-order numerical optimisation techniques applied to CRF parameter estimation were described and implementation details of a program to compare CRF training algorithms were given.

Finally, the performance of the numerical techniques implemented were compared with that of Lafferty *et al.*'s IIS-based algorithm on a toy problem based on the NLP task of shallow parsing. These experimental results indicated that numerical optimisation techniques were faster at training CRFs than iterative scaling.

Chapter 5

Conclusions

The aim of the work described in this thesis was to compare the performance of general numerical optimisation techniques with that of iterative scaling algorithms for the task of estimating the parameters of conditional random fields. As an initial step, theoretical analysis of CRFs and the iterative scaling algorithms presented in current literature [31] was carried out. This analysis (see Chapter 3 for details) revealed that the GIS-based algorithm for CRFs [31] is in fact intractable due to the sequential nature of the data being modelled. Additionally, Lafferty *et al.*'s IIS-based algorithm requires the use of a mean field approximation which may slow convergence.

In addition to these theoretical disadvantages of iterative scaling for CRFs, recent experimental work of Malouf [32] also suggested that iterative scaling may not be the most practical method of training CRFs. Malouf's work demonstrated that first- and second-order general numerical optimisation techniques significantly reduced the time taken to train (non-sequential) conditional maximum entropy models – a type of log-linear model with a very similar functional form to the distribution defined by a conditional random field. To investigate whether these numerical optimisation techniques would enable efficient training of CRFs, a CRF implementation was developed using matrix and vector primitives provided by the PETSc software library. The IIS-based parameter estimation algorithm and the framework for the numerical optimisation algorithms were implemented in C++ using data structures and routines provided by PETSc. The actual numerical optimisation routines used were two conjugate gradient based algorithms (Fletcher-Reeves and Polak-Ribière-Positive) and the limited memory variable metric algorithm of Benson and Moré [7]. The implementations of these algorithms used were those provided by TAO (the Toolkit for Advanced Optimisation), a framework for implementing software involving optimisation routines.

The original intention was to compare the performance of iterative scaling and the numerical optimisation algorithms using the text chunking data from the CoNLL-2000 shared task to train CRF models. However implementation issues meant that using a dataset of this size was not possible during the time frame of the project. Instead, the algorithms were compared using a tiny subset of this data which, despite the trivial nature, clearly highlighted the performance differences between the parameter estimation techniques compared.

The results obtained for CRF parameter estimation using this naïve dataset echo Malouf's findings for non-sequential conditional maximum entropy models, thus confirming his findings using a similar, but non-identical, theoretical framework. The most striking feature of the results presented in this thesis is that all of the numerical optimisation techniques outperformed Lafferty *et al.*'s IIS-based method. Additionally, Benson and More's limited-memory variable metric algorithm performed much better than the first-order conjugate gradient algorithms also investigated.

In conclusion, the work described in this thesis indicates that the iterative scaling techniques presented in current literature on CRF parameter estimation suffer from theoretical and practical disadvantages, making CRFs an impractical choice for labelling real-world sequential data. Although the experiments performed will need to be re-run using much larger data sets, the experimental results presented here reveal that it is possible to train CRFs using general first- and second-order numerical optimisation techniques with a much better convergence rate than that exhibited by iterative scaling. This is a highly promising finding as it provides a means for CRFs to be trained without relying on other models to provide initial parameter values.

Bibliography

[1] Steven P. Abney. *Principle-Based Parsing: Computation and Psycholinguistics*, chapter Parsing by Chunks, pages 257–278. Kluwer Academic Publishers, Boston, 1991.

[2] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. PETSc web page. 2001.

[3] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. PETSc 2.0 users manual. Technical Report ANL-95/11 - Revision 2.1.2, Argonne National Laboratory, 2002.

[4] Iain Bancarz and Miles Osborne. Improved Iterative Scaling can yield multiple globally optimal models with radically differing performance levels. In *Coling 2002*, Taipei, Taiwan, 2002.

[5] Steve Benson, Lois Curfman McInnes, and Jorge More. Toolkit for Advanced Optimization (TAO) web page.

[6] Steve Benson, Lois Curfman McInnes, Jorge More, and Jason Sarich. TAO users manual. Technical Report ANL/MCS-TM-242 - Revision 1.4, Argonne National Laboratory, 2002.

[7] Steven J. Benson and Jorge J. Moré. A limited memory variable metric method for bound constrained optimisation. Technical Report ANL/ACS-P909-0901, Argonne National Laboratory, 2001.

[8] A. Berger and R. Miller. Just-in-time language modelling. In *Proceedings of the ICASSP-98*, Seattle, WA, 1998.

[9] Adam L. Berger. The improved iterative scaling algorithm: A gentle introduction, 1997.

[10] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[11] Jeff A. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report ICSI-TR-97-021, International Computer Science Institute, Berkeley, 1997.

[12] Léon Bottou. *Une approache théoretique de l'apprentissage connexionniste: Applications à la reconnaissance de la parole*. PhD thesis, Université de Paris XI, Paris, 1991.

[13] P. Brown, S. Della Pietra, V. Della Pietra, , R. Mercer, A. Nadas, and S. Roukos. Translation models using learned features and a generalized Csiszar algorithm. IBM Technical Report.

[14] Stanley F. Chen and Ronald Rosenberg. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, School of Computer Science, Carnegie Mellon University, February 1999.

[15] Peter Clifford. Markov random fields in statistics. In Geoffrey Grimmett and Dominic Welsh, editors, *Disorder in Physical Systems: A Volume in Honour of John M. Hammersley*, pages 19–32. Oxford University Press, 1990.

[16] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[17] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[18] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[19] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[20] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478, 1996.

[21] J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

[22] John E. Hopcroft, Rajeev Motwani, and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.

[23] E.T. Jaynes. Information theory and statistical mechanics. *The Physical Review*, 106(4):620–630, May 1957.

[24] E.T. Jaynes. Information theory and statistical mechanics II. *The Physical Review*, 108(2):171–190, October 1957.

[25] Frederick Jelinek. *Statistical methods for speech recognition*. MIT Press, 1997.

[26] M. Jordan and C. Bishop. An introduction to graphical models.

[27] Sham Kakade, Yee Whye Teh, and Sam Roweis. An alternative objective function for Markovian fields. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[28] E. Kim-Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000*, Lisbon, Portugal, 2000.

[29] Rob Koeling. Chunking with maximum entropy models. In *Proceedings of CoNLL-2000*, Lisbon, Portugal, 2000.

[30] J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242, 1992.

[31] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML 2001*, 2001.

[32] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, 2002.

[33] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. ICML 2000*, 2000.

[34] Antonio Molina and Ferran Pla. Shallow parsing using specialized hmms. *Journal of Machine Learning Research*, 2:595–613, March 2002.

[35] K. Murphy and M. Paskin. Linear time inference in hierarchichal HMMs. *Neural Information Processing Systems*, December 2001.

[36] Jorge Nocedal. Large scale unconstrained optimization. In A. Watson and I. Duff, editors, *The State of the Art in Numerical Analysis*, pages 311–338. Oxford University Press, 1997.

[37] Jorge Nocedal and Steve Wright. *Numerical Optimization*. Springer, New York, 1999.

[38] Miles Osborne. Shallow parsing as part-of-speech tagging. In *Proceedings of CoNLL-2000*, Lisbon, Portugal, 2000.

[39] Miles Osborne. Shallow parsing using noisy and non-stationary training material. *Journal of Machine Learning Research*, 2:695–719, 2002.

[40] A. Paz. *Introduction to Probabilistic Automata*. Academic Press Inc., 1971.

[41] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. Technical Report CMU-CS-95-144, Carnegie Mellon University, 1995.

[42] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):380–393, 1997.

[43] Ferran Pla, Antonio Molina, and Natividad Prieto. Improving text chunking by means of lexical-contextual information in statistical language models. In *Proceedings of CoNLL-2000*, Lisbon, Portugal, 2000.

[44] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series. Prentice-Hall, Inc., 1993.

[45] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[46] Adwait Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*. University of Pennsylvania, May 1996.

[47] Adwait Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania, 1997.

[48] Ronald Rosenfeld. *Adaptive Statistical Language Modelling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, April 1994.

[49] C.E. Shannon. A mathematical theory of communication. *Bell System Tech. Journal*, 27:379–423 and 623–656, 1948.

[50] Jun Wu and Sanjeev Khudanpur. Efficient training methods for maximum entropy language modelling. In *Proceedings of the ICSLP2000*, volume 3, pages 114–117, Beijing, 2000.

[51] GuoDong Zhou, Jian Su, and TongGuan Tey. Hybrid text chunking. In *Proceedings of CoNLL-2000*, Lisbon, Portugal, 2000.