Chapter 2

Maximum Entropy Modeling

In this chapter, we introduce the maximum entropy principle and its application to language modeling. We will show a concrete example of building a trigram model using the maximum entropy method. We also address the computational issues and challenges in estimating model parameters.

2.1 The Maximum Entropy Principle

When we try to model the behavior of a random variable or a stochastic process from data, we should choose through intuition a model that satisfies all observed properties of this variable or process while making no “unwarranted” assumptions. This results in a maximum entropy model in the sense of information theory. We introduce the maximum entropy principle by providing simple examples of dice. Readers will see that the maximum entropy results meet our intuitions.

2.1.1 Examples: Dice

Let \( p_i \) be the probability of the event that the facet with \( i \) dots faces up, where \( i = 1, \cdots, 6 \). We want to estimate \( p_i \) with no empirical knowledge. The intuitively appealing model is the uniform model, in which \( p_i = \frac{1}{6} \) for \( i = 1, 2, \cdots, 6 \). Actually, it is the maximum entropy model! Now let us find the maximum entropy distribution
mathematically and check whether it matches our intuitive uniform model.

**Problem 1:**

Find the probability distribution \( P = (p_1, p_2, \cdots , p_6) \), which maximizes the entropy

\[
H(P) = -\sum_{i=1}^{6} p_i \log(p_i).
\]

**Solution:**

An implicit constraint for a probability distribution is \( \sum_{i=1}^{6} p_i = 1 \).
We use the method of *undetermined Lagrangian multipliers*.
Let

\[
\mathcal{L}(P, \alpha) = \alpha \left( \sum_{i=1}^{6} p_i - 1 \right) + (-\sum_{i=1}^{6} p_i \log(p_i)).
\]

Setting the partial derivatives with respect to \( p_j \) to zero, we get

\[
\frac{\partial \mathcal{L}}{\partial p_j} = -1 - \log p_j + \alpha = 0.
\]

Solving the above equations we obtain the following solution:

\[
p_i = \frac{1}{6}, \text{ for } i = 1, \cdots , 6.
\]

Next, we assume that the die is loaded, and we observe that the facet with five dots has a \( \frac{1}{3} \) chance of facing up. We want to find the values of \( p_i \) that satisfy this empirical constraint. Our intuition tells us to choose \( \frac{1}{3} \) for \( p_5 \) and an equal value for the rest of the \( p_i \). Once again, we use the maximum entropy method to solve this problem.

**Problem 2:**

Find the probability distribution \( p_i \), for \( i = 1, 2, \cdots , 6 \), which maximizes \( H(P) \) subject to the constraint

\[
P[f] = \sum_{i=1}^{6} p_i f(i) = \frac{1}{3}.
\]
where \( f \), defined as

\[
f(i) = \begin{cases} 
1 & \text{if } i = 5, \\
0 & \text{otherwise},
\end{cases}
\]

is an indicator function describing the specific constraint for the facet with five dots.

**Solution:**

Creating the Lagrangian

\[
\mathcal{L}(P, \alpha) = \alpha_1 \left( \sum_{i=1}^{6} p_i - 1 \right) + \alpha_2 \left( \sum_{i=1}^{6} f(i) \cdot p_i - \frac{1}{3} \right) + \left( - \sum_{i=1}^{6} p_i \log(p_i) \right),
\]

and setting its partial derivatives \( \frac{\partial \mathcal{L}}{\partial p_j} \) to 0, we obtain the following equations:

\[
\alpha_1 + \alpha_2 - \log p_5 - 1 = 0,
\]
\[
\alpha_1 - \log p_i - 1 = 0, \quad \text{for } i = 1, 2, 3, 4, 6.
\]

Solving the above equations, we obtain the solution

\[
p_i = \frac{2}{15}, \quad \text{for } i = 1, 2, 3, 4, 6, \quad \text{and } p_5 = \frac{1}{3}.
\]

From the two examples above, we can see that maximum entropy distributions match our intuitions. Shore & Johnson (1980), Jaynes (1982) and Csiszár (1991) showed why the maximum entropy distribution is the optimal among all probability distributions obtained from (incomplete) data. The examples above also show a way of solving ME problems using Lagrangian multipliers. Unfortunately, the method of Lagrangian multipliers is not practical for most applications since the number of constraints is huge, resulting in an enormously large number of nonlinear equations to be solved. The analytical solution is either non-existent or hard to find. However, maximum entropy models can be estimated by numerical methods, which will be introduced in the following sections.
2.1.2 The General Problem

Problem 2 in the previous section is a concrete example of inferring a probability distribution from some observations. A typical problem of statistical modeling is to estimate a probability measure \( p(s) \) on \( \mathcal{S} \) from independent and identically distributed observations \( s_1, s_2, \ldots, s_L \), where \( L \) is the number of samples. In the maximum entropy framework, a distribution \( p^* \) is chosen to maximize the entropy \( H(p) = -\sum_{s \in \mathcal{S}} p(s) \log p(s) \) from a linear family of probability distributions

\[
\mathcal{P} = \{ p : \sum_{s \in \mathcal{S}} p(s) g_k(s) = a_k, 1 \leq k \leq K \} \tag{2.1}
\]

for a given set of real-valued functions \( G = \{ g_1, g_2, \ldots, g_K \} \) defined on \( \mathcal{S} \). It is easy to check that \( \mathcal{P} \) is closed and convex.

The problem above is equivalent to finding a distribution \( p^* \) to minimize the I-divergence

\[
D(p||q) = \sum_{s \in \mathcal{S}} p(s) \log \frac{p(s)}{q(s)}
\]

between \( p \) and the uniform distribution \( q(s) = \frac{1}{|\mathcal{S}|}, \forall s, \text{i.e.,} \)

\[
p^* = \arg \min_{p \in \mathcal{P}} D(p||q).
\]

We define an exponential family

\[
\mathcal{R} = \left\{ p : p(s) = \frac{1}{z} q(s) \exp \left[ \sum_{k=1}^{K} \lambda_k g_k(s) \right], \text{for some } \lambda_1, \lambda_2, \ldots, \lambda_K \right\} \tag{2.2}
\]

for the given functions \( g_1, g_2, \ldots, g_K \) and the reference distribution \( q \), where

\[
z = \sum_{s \in \mathcal{S}} q(s) e^{\sum_{k=1}^{K} \lambda_k g_k(s)}
\]

is a normalization constant.

2.1.3 The Maximum Entropy Solution

Csiszár (1975) showed that intersection \( \mathcal{P} \cap \bar{\mathcal{R}} = \{ p^* \} \) where \( \bar{\mathcal{R}} \) is the closure of \( \mathcal{R} \). This means that the maximum entropy distribution \( p^* \) of the linear family \( \mathcal{P} \) is
unique and has an exponential form, and conversely, if a \( pmf \, p^* \) has the exponential form and satisfies the linear constraints \( \{ \sum_s p(s) g_k(s) = a_k, 1 \leq k \leq K \} \), then it has maximum entropy.

Therefore, instead of finding the maximum entropy distribution from the linear family \( \mathcal{P} \), we look for the model satisfying these linear constraints from the exponential family \( \mathcal{R} \), since the latter can be solved by the generalized iterative scaling method (Darroch & Ratcliff, 1972; Csiszár, 1989).

**Generalized Iterative Scaling**

Darroch & Ratcliff (1972) proposed the generalized iterative scaling (GIS) algorithm for seeking the probability distribution with an exponential form,

\[
p(s) = c \cdot q(s) \prod_{k=1}^{K} \alpha_k^{g_k(s)}, \tag{2.3}
\]

which satisfies linear constraints of the form

\[
\sum_{s \in S} p(s) \cdot g_k(s) = a_k, \text{ for } k = 1, 2, \ldots, K, \tag{2.4}
\]

where the \( g_k \)'s are real-valued functions. They proved that starting with \( p_0 = q \), the estimate \( p_{n+1}(s) \) of \( p \) in the \((n + 1)^{th}\) iteration computed by

\[
p_{n+1}(s) = p_n(s) \prod_{k=1}^{K} \left( \frac{\alpha_k}{a_{i,n}} \right)^{g_k(s)}, \tag{2.5}
\]

where

\[
a_{i,n} = \sum_s p_n(s) g_k(s) \tag{2.6}
\]

is the expectation of the \( i^{th}\) feature function in the \( n^{th}\) iteration, converges, i.e., \( p_n(s) \to p^*(s) \)

\[
p^*(s) = \frac{1}{z} \exp \{ \sum_{k=1}^{K} \lambda_k^* g_k(s) \}.
\]

It is worth emphasizing that there exists a unique solution to (2.3) and (2.4).
Starting from some arbitrary \( p_0 = q^1 \), \( p_n \) converges to a unique solution for maximum entropy models by the GIS algorithm. However, GIS has some practical problems. First, the convergence is quite slow for large models. Furthermore, if the linear constraints are not consistent\(^2\), intermediate values of model parameters may even overflow/underflow the range of double precision representation before they would converge. An improved iterative scaling algorithm has been proposed by Della Pietra, Della Pietra & Lafferty (1997) to overcome these computational issues.

**Improved Iterative Scaling**

We need some additional notation

\[
g_\#(s) = \sum_{k=1}^{K} g_k(s)
\]

to describe the IIS algorithm. If \( g_k(s) \) is binary, \( g_\#(s) \) is the total number of features that are active on the \( s \).

Compared to Equation (2.5), a more general iterative solution for \( p^* \) has the following form:

\[
p_{n+1}(s) = p_n(s) \cdot u
\]

(2.7)

where \( u = u_1, u_2, \ldots, u_K \) is a \( K \) dimensional vector of positive constants. Della Pietra, Della Pietra & Lafferty (1997) showed that if \( u \) is the unique solution \(^3\) of

\[
\sum_{s \in \mathcal{S}} p_n(s) \cdot u_k \#(s) \cdot g_k(s) = a_k
\]

for all \( k \), then the iterative equation (2.7) provides a solution for \( p_n \rightarrow p^* \). In practice, the IIS algorithm converges faster than GIS \(^4\).

\(^1\) Usually initialized as a uniform distribution.

\(^2\) Very likely in real applications.

\(^3\) The computational cost of solving these equations numerically using Newton's method may be ignored when compared to the total training time.

\(^4\) If \( g_\#(s) \) is a constant for all \( s \), IIS is the same as GIS.
2.2 Maximum Entropy Language Modeling

In this section, we apply the maximum entropy principle to language modeling. In order to formalize the computation of maximum entropy language models that satisfy constraints we set, we need to define some concise notation. We use \( x \) and \( y \), respectively, to represent the history and the word following that history in language modeling. For instance, \( x = w_{i-2}, w_{i-1} \) and \( y = w_i \) in a trigram model \( p(w_i|w_{i-2}, w_{i-1}) \). We use \( \mathcal{X} \) and \( \mathcal{Y} \) to represent the set of histories and the set of (future) words, respectively. A typical problem in language modeling and in many other empirical natural language processing applications is to estimate a conditional probability measure \( p(y|x) \) on a probability space \( \mathcal{X} \times \mathcal{Y} \). Here, \( \mathcal{Y} = V \) where \( V \) is the vocabulary. In the trigram model \( p(w_i|w_{i-2}, w_{i-1}) \), \( \mathcal{X} = \mathcal{Y} \times \mathcal{Y} \).

In the maximum entropy framework, a set of \( K \) real-valued feature functions \( \{g_1, \cdots, g_K\} \) need to be defined on \( \mathcal{X} \times \mathcal{Y} \). However, for convenience, only binary feature functions

\[
G = \{g_k : (\mathcal{X} \times \mathcal{Y}) \to \{0, 1\}, k = 1, \ldots, K\}
\]

are considered in language modeling. \(^5\)

![Figure 2.1: A binary feature \( g_k \) partitions \( \mathcal{X} \times \mathcal{Y} \).](image)

Each binary feature \( g_k \) for \( k = 1, \cdots, K \), partitions the domain \( \mathcal{X} \times \mathcal{Y} \) into two sets \( A_k \) and \( (\mathcal{X} \times \mathcal{Y}) \setminus A_k \): those \( \langle x, y \rangle \in \mathcal{X} \times \mathcal{Y} \) for which \( g_k(x, y) \) is active and those

\(^5\)We will address how to choose these features for language models in subsequent chapters.
for which it is not (Figure 2.1), or more formally,

\[
g_k(x, y) = g_k(x, y; A_k) = \begin{cases} 
1 & (x, y) \in A_k, \\
0 & \text{otherwise},
\end{cases}
\]  

(2.8)

where \(x\) and \(y\) are variables and \(A_k\) can be regarded as a coefficient.

For instance, for words \(a, b, c \in V\), unigram, bigram and trigram feature functions are defined respectively as

\[
g_1(x, y; c) = \begin{cases} 
1 & y = c, \\
0 & \text{otherwise},
\end{cases}
\]  

(2.9)

\[
g_2(x, y; b, c) = \begin{cases} 
1 & x = (\ast, b) \text{ and } y = c, \\
0 & \text{otherwise},
\end{cases}
\]

and

\[
g_3(x, y; a, b, c) = \begin{cases} 
1 & x = (a, b) \text{ and } y = c, \\
0 & \text{otherwise}.
\end{cases}
\]

A vector \(a = (a_1, ..., a_K)\) is chosen to be the target expectation\(^6\) for these features based on the observed samples \((x_1, y_1), ..., (x_L, y_L)\). Typically, \(a_k = \frac{\#_{(x_i, y_i) \in A_k}}{L}\).

The set of features \(G\) and their target expectations \(a\) define a linear family of probability distributions

\[
\mathcal{P} = \{p : p[g_k] = a_k, \ \forall \ k = 1, ..., K\},
\]  

(2.10)

where \(p[g_k]\) is the expectation of \(g_k(x, y; A_k)\) with respect to the distribution \(p(x, y)\), as

\[
p[g_k] = \sum_{x \in X, y \in Y} p(x, y) \cdot g_k(x, y; A_k).
\]  

(2.11)

Note that (2.10) has the same form as (2.1). The only difference is that \(g_k\) in (2.1) is a real-valued function. It is binary-valued in (2.10).

\(^6\)E.g., empirical expectation.
Given such a class $\mathcal{P}$ of probability distributions, we want to find the distribution $p^*(x, y)$ in $\mathcal{P}$ that maximizes the entropy $H(p)$. Since we have shown in Section 2.1.3 that the maximum entropy solution has the exponential form, we consider the alternative class $\mathcal{R}$ of exponential models $m(x, y)$ defined via $G$ as

$$\mathcal{R} = \{m : m(x, y) = \frac{1}{z} e^{\sum_{k=1}^{K} \lambda_k g_k(x, y; A_k)}\},$$

or, in the simpler notation of Ristad (1997),

$$\mathcal{R} = \left\{ m : m(x, y) = \frac{r(x, y)}{z} \right\},$$

where

$$r(x, y) = \prod_{k=1}^{K} \alpha_k^{g_k(x, y; A_k)}, \quad z = \sum_{x \in X, y \in Y} r(x, y)$$

and the $\alpha$’s are nonnegative real numbers. Here $\lambda_k = \ln(\alpha_k)$.

In language modeling, we are more interested in the conditional probability $m(y|x)$ than in $m(x, y)$ where

$$m(y|x) = \frac{1}{z(x)} \prod_{k=1}^{K} e^{\lambda_k g_k(x, y; A_k)}, \quad z(x) = \sum_{y \in Y} \prod_{k=1}^{K} e^{\lambda_k g_k(x, y; A_k)}.$$  

Given the set $G$ of feature functions and their target expectations $a$, parameters $\alpha$ (or $\lambda$) are computed using either GIS or IIS. The art of maximum entropy language modeling is in choosing the most “useful” feature functions and setting proper target expectations for them.

### 2.3 An Example: Building a Maximum Entropy Trigram Model

In this section, we show how to build an ME language model by a concrete example of a trigram model. For a triple $(w_{i-2}, w_{i-1}, w_i)$, the model estimating $p(w_i|w_{i-2}, w_{i-1})$ should satisfy the following unigram, bigram and trigram constraints that are copied
from (1.3)-(1.5):
\[
\sum_{w_{i-2}, w_{i-1}} p(w_i|w_{i-1}, w_{i-2}) \cdot p(w_{i-2}, w_{i-1}) = \frac{\#[w_i]}{\#[\text{training data}]}, \quad (2.16)
\]
\[
\sum_{w_{i-2}} p(w_i|w_{i-2}, w_{i-1}) \cdot p(w_{i-2}, w_{i-1}) = \frac{\#[w_{i-1}, w_i]}{\#[\text{training data}]}, \quad (2.17)
\]
\[
p(w_i|w_{i-2}, w_{i-1}) \cdot p(w_{i-2}, w_{i-1}) = \frac{\#[w_{i-2}, w_{i-1}, w_i]}{\#[\text{training data}]}. \quad (2.18)
\]
We define a binary function
\[
g_1(w_i) \triangleq g(x, y; w_i) = \begin{cases} 1 & y = w_i, \\ 0 & \text{otherwise} \end{cases} \quad (2.19)
\]
corresponding to the first (unigram) constraint. The expectation of this feature under a model \( p \),
\[
p[g_1] = \sum_{(x,y):y=w_i} g(x, y; w_i) p(y|x) p(x) = \sum_{w_{i-2}, w_{i-1}} p(w_i|w_{i-2}, w_{i-1}) p(w_{i-2}, w_{i-1})
\]
is identical to the left hand side of (2.16). Finding a model satisfying the unigram constraint (2.16) is equivalent to finding a model \( p \) under which the expectation of the feature function \( g_1(w_i) \) satisfies
\[
\sum_{(x,y):y=w_i} g(x, y; w_i) p(y|x) p(x) = \frac{\#[w_i]}{\#[\text{training data}]}. \quad (2.20)
\]
Similarly we can define the bigram feature function
\[
g_2(w_{i-1}, w_i) \triangleq g(x, y; w_{i-1}, w_i) = \begin{cases} 1 & y = w_i \text{ and } x = * \text{ w}_{i-1}, \\ 0 & \text{otherwise} \end{cases} \quad (2.21)
\]
corresponding to the bigram constraint (2.17), where * designates any prefix word string, and the trigram feature function
\[
g_3(w_{i-2}, w_{i-1}, w_i) \triangleq g(x, y; w_{i-2}, w_{i-1}, w_i) = \begin{cases} 1 & y = w_i \text{ and } x = * \text{ w}_{i-2}, w_{i-1}, \\ 0 & \text{otherwise} \end{cases} \quad (2.22)
\]
corresponding to the trigram constraint (2.18). The model satisfying the N-gram constraints (2.16)-(2.18) is the model under which the expectations of the feature functions (2.19), (2.21) and (2.22) equal their empirical expectations. According to the conclusion in Section 2.2, the model has the exponential form of (2.15).

Equation (2.15) involves K factors for each conditional probability. However, given any specific 3-tuple \( w_{i-2}, w_{i-1}, w_i \), the trigram model has at most three features \( g_1(w_i) \), \( g_2(w_{i-1}, w_i) \) and \( g_3(w_{i-2}, w_{i-1}, w_i) \) active simultaneously; therefore, subscripting the \( \alpha \)'s with the corresponding words instead of a generic index \( k \), we get

\[
p(w_i|w_{i-2}, w_{i-1}) = \frac{\alpha_{w_i}^1(w_i) \cdot \alpha_{w_{i-1}, w_i}^2(w_{i-1}, w_i) \cdot \alpha_{w_{i-2}, w_{i-1}, w_i}^3(w_{i-2}, w_{i-1}, w_i)}{z(w_{i-2}, w_{i-1})},
\]

where

\[
z(w_{i-2}, w_{i-1}) = \sum_{w_i \in V} \alpha_{w_i}^1(w_i) \cdot \alpha_{w_{i-1}, w_i}^2(w_{i-1}, w_i) \cdot \alpha_{w_{i-2}, w_{i-1}, w_i}^3(w_{i-2}, w_{i-1}, w_i).
\]

Substituting all \( \lambda = \ln(\alpha) \) for \( i = 1, 2 \) and 3 in (2.23), we obtain (1.6). Now we have answered the remaining questions in Section 1.4, namely, "why ME models have the exponential form, how to define feature functions and how to obtain feature parameters."

### 2.4 Features of Maximum Entropy Language Models

Features of maximum entropy models can come from any kind of information sources besides the N-grams shown in the previous section. In this section, we study several kinds of feature functions used in language models in this dissertation.

According to definition (2.8), the feature function \( g_k(x, y; A_k) \) is identified by the subset \( A_k \), and thus the subscript \( k \) may be omitted without causing any confusion. Even though \( A_k \) could be any subset of \( \mathcal{X} \times \mathcal{Y} \) in principle, it may be assumed to contain a subset of \( \mathcal{X} \) and only one element in \( \mathcal{Y} \) in language modeling\(^7\) for simplicity.

\(^7\)Because a language model is used to predict the probability of a word given the context
A meaningful constraint in a maximum entropy model should reflect the dependencies between histories and future words obtained from some sources of information. In order to organize the feature set properly, we categorize feature functions by the information source from which these features come and according to their order.

### 2.4.1 Feature Categorization By Information Source

#### Collocation Dependencies

We use the collocation dependencies, i.e., the dependencies of the current word and several preceding words, in all our ME models because they are reliable and have been shown to be effective in all kinds of language models. In this dissertation, we use $w_i$ to represent the current word and $w_{i-N+1}^{i-1} \doteq w_{i-N+1}, \cdots, w_{i-1}$ for N-1 preceding words. The collocation N-gram features have the form of

$$g(w_{i-N+1}, \cdots, w_{i-1}, w_i) = g(x, y; w_{i-N+1}, \cdots, w_{i-1}, w_i)$$

$$= \begin{cases} 
1 & \text{if } x \text{ ends with the suffix } w_{i-N+1}^{i-1}, \text{ and } y = w_i, \\
0 & \text{otherwise}.
\end{cases}$$

#### Topic Dependencies

The topic $t_i$ of the discussion strongly influences the use of the current word $w_i$. An ME language model may use this topic dependency. A topic feature has the form of

$$g(t_i, w_i) = g(x, y; t_i, w_i) = \begin{cases} 
1 & \text{if } t_i \text{ is the topic of } x \text{ and } y = w_i, \\
0 & \text{otherwise}.
\end{cases}$$

The short-hand notation $g(t_i, w_i)$ is used for topic unigram features in this dissertation.

#### Syntactic Dependencies

We will show in Chapter 5 the benefit of syntactic dependencies in improving the language model performance. Two kinds of syntactic dependencies are explored in
this dissertation. The first kind involves syntactic head words $hw_{i-2}, hw_{i-1}$ and the current word $w_i$. The feature function can depend on only one head word $hw_{i-1}$ or on both head words, with the forms

$$g(hw_{i-1}, w_i) \doteq g(x, y; hw_{i-1}, w_i)$$

$$= \begin{cases} 
1 & hw_{i-1} \text{ is the preceding head word in } x \text{ and } y = w_i, \\
0 & \text{otherwise}
\end{cases}$$

and

$$g(hw_{i-2}, hw_{i-1}, w_i) \doteq g(x, y; hw_{i-2}, hw_{i-1}, w_i)$$

$$= \begin{cases} 
1 & hw_{i-2}, hw_{i-1} \text{ are the two preceding head words in } x \text{ and } y = w_i, \\
0 & \text{otherwise},
\end{cases}$$

respectively.

Another kind of syntactic dependency (already mentioned in Chapter 1) is that between syntactic non-terminal (NT) labels $nt_{i-2}, nt_{i-1}$ of the preceding heads and the current word $w_i$, with the corresponding feature functions

$$g(nt_{i-1}, w_i) \doteq g(x, y; nt_{i-1}, w_i)$$

$$= \begin{cases} 
1 & nt_{i-1} \text{ is the NT of preceding head in } x \text{ and } y = w_i, \\
0 & \text{otherwise}
\end{cases}$$

and

$$g(nt_{i-2}, nt_{i-1}, w_i) \doteq g(x, y; nt_{i-2}, nt_{i-1}, w_i)$$

$$= \begin{cases} 
1 & nt_{i-2}, nt_{i-1} \text{ are the two preceding NT labels in } x \text{ and } y = w_i, \\
0 & \text{otherwise}.
\end{cases}$$

\(^8\)Briefly introduced in Chapter 1 and to be discussed in detail in Chapter 5
Word-Class Based Dependencies
When the vocabulary size is large and the training data is relatively sparse, the
trigram dependence may not be reliable. One widely used approach to avoid this
problem is to cluster words into word classes, and to accumulate statistics according
to word classes instead of individual words. The most common word classification
is based upon the part-of-speech (POS) tag of words. In this dissertation, we also
explore the dependencies between the part-of-speech tags of the two preceding words,
$pos_{i-2}, pos_{i-1}$, and the current word $w_i$, and we create feature functions

$$g(pos_{i-1}, w_i) = g(x, y; pos_{i-1}, w_i) = \begin{cases} 1 & pos_{i-1} \text{ is the preceding POS tag in } x \text{ and } y = w_i, \\ 0 & \text{otherwise} \end{cases}$$

and

$$g(pos_{i-2}, pos_{i-1}, w_i) = g(x, y; pos_{i-2}, pos_{i-1}, w_i) = \begin{cases} 1 & pos_{i-2}, pos_{i-1} \text{ are the two preceding POS tags in } x \text{ and } y = w_i, \\ 0 & \text{otherwise.} \end{cases}$$

2.4.2 Feature Categorization By Order

We call the unigram feature $g(w_i)$ an order 1 feature because its value depends on
one variable $y$. Similarly, we call the bigram feature $g(w_{i-1}, w_i)$ an order 2 feature and
the trigram feature $g(w_{i-2}, w_{i-1}, w_i)$ an order 3 feature because they are determined
by 2 and 3 variables, respectively. In general, we call the N-gram feature an order $N$
feature.

Since the value of the unigram feature function $g(w_i)$ depends only on whether
$y = w_i$ and is independent of $x$, the unigram feature is also called a marginal feature.
All other features do depend on $x$ and are thus called conditional features.

Even though features of ME models can come from other information sources than
collocation dependence, they still look like N-gram features. For example, the head
word features \( g(hw_{i-2}, hw_{i-1}, w_i) \) look like a trigram model and thus can be regarded as head word trigram features. We can assign an order to all features as we have done for N-gram features.

If a history \( x \) is an \( m \)-vector \((x_1, x_2, \cdots, x_m)\) where \( x_1, x_2, \cdots, x_m \) are symbols, a feature function is called an order \( N \) feature function if its value depends on \( N - 1 \) symbol in \((x_1, x_2, \cdots, x_m)\) and on \( y \). For example, \( g(nt_{i-2}, nt_{i-1}, w_i) \) is an order 3 feature function.

In this dissertation, we only consider features of order 3 or lower because the estimate of higher order feature parameters is not very reliable due to the data sparseness problem.

### 2.4.3 Feature Patterns

Each kind of feature described in Section 2.4.1 may be called a feature pattern (or a feature type) in later chapters. In summary, we have defined some N-gram feature patterns denoted as \( g(w_{i-N+1}, \cdots, w_i) \), one feature pattern for topic constraints, two patterns for head word constraints and two for non-terminal constraints in Section 2.4.1. Two features with the same order but from different information sources will not be confusing since they belong to different patterns, even though they are in a similar form. For instance, both \( g(t_i, w_i) \) and \( g(w_{i-1}, w_i) \) have two coefficients, but they are clearly corresponding to the topic feature and the regular bigram feature, respectively.

In the rest of this section, we will give a formal definition for a feature pattern, which is based on the concept of subsequence of a history.

**Definition (Subsequence and Supersequence):**

\( s' = s_{k_1}, s_{k_2}, \cdots, s_{k_l} \) is a subsequence of a string (or a vector) \( s = s_1, s_2, \cdots, s_d \) if \( 1 \leq k_1 < k_2 < \cdots < k_l \leq d \). \( s \) is then a supersequence of \( s' \).

A history \( x \) can be treated as a vector \( x_1, x_2, \cdots, x_d \) where \( x_i \) is a symbol at position \( i \), and the number of components of \( x \) is fixed. A subsequence \( x' = x_{k_1}, x_{k_2}, \cdots, x_{k_l} \) of \( x \) can thus be treated as a class of histories with the same symbols in positions \( k_1, k_2, \cdots, k_l \). For example, if \( x = x_1, x_2, x_3, x_4 = hw_{i-2}, hw_{i-1}, nt_{i-2}, nt_{i-1} \) denotes
the history of two preceding head words and two preceding non-terminal labels, 
\( x' = x_3, x_4 = nt_{i-2}, nt_{i-1} \) denotes all histories with the same non-terminal labels 
\( nt_{i-2}, nt_{i-1} \) no matter what the head words are.

**Definition (Feature Pattern):**
Let \( x = x_1, x_2, \cdots, x_d \) be a history and \( x_{k_1}, x_{k_2}, \cdots, x_{k_l} \) be a subsequence of history \( x \) where \( k_l \leq d \). All features that depend on symbols at position \( k_1, k_2, \cdots, k_l \) are regarded as belonging to one feature pattern denoted as \( g(x_{k_1}, x_{k_2}, \cdots, x_{k_l}, y) \).

The order \( l + 1 \) and the information source of a feature are clearly indicated by the feature pattern.

### 2.5 Training Maximum Entropy Language Models

We have introduced the basic ideas of iterative scaling for training maximum entropy models in general in Section 2.1.3. In this section we show the state-of-the-art training methods for language models in particular.

In the language model of Equation(2.15),

\[
m(y|x) = \frac{1}{z_\lambda(x)} \prod_{k=1}^{K} e^{\lambda_k \cdot g_k(x, y; A_k)}, \quad z_\lambda(x) = \sum_{y} \prod_{k=1}^{K} e^{\lambda_k \cdot g_k(x, y; A_k)}.
\]

To train \( \lambda \)'s by either GIS or IIS, we need to compute the expectations of all features \( g_k \in G \) under a model \( m \) as

\[
m[g_k] = \sum_{x, y} m(x, y) \cdot g_k(x, y) = \sum_{\langle x, y \rangle: g_k(x, y) = 1} m(x) \cdot m(y|x), \quad \text{for} \ k = 1, \cdots, K, (2.24)
\]

or the partial counts of the expectations as:

\[
m_{j}[g_k] = \sum_{\langle x, y \rangle: g_k#(x, y) = j} m(x) \cdot m(y|x) \cdot g_k(x, y), \quad \text{for} \ k = 1, \cdots, K \quad (2.25)
\]

where \( j = 1, \cdots, \max g_k#(x, y) \) and when we use the abbreviated notation \( g_k(x, y) = g_k(x, y; A_k) \) for simplicity.

For each feature \( g_k \), we must find all \( \langle x, y \rangle \) for which \( g_k \) is active and \( m(x)m(y|x) \) is non-zero. If that feature is a marginal one whose value is independent of \( x \), it
has $|\mathcal{X}|$ nonzero terms in the summation. Since there are $|\mathcal{Y}|$ marginal features, the training time per iteration is at least $O(|\mathcal{X}| \cdot |\mathcal{Y}|)$. For a simple trigram model, this complexity is $O(|\mathcal{Y}|^3)$ when $\mathcal{X} = \mathcal{Y} \times \mathcal{Y}$, which is intractable even for a small task such as Switchboard with $|\mathcal{Y}| \approx 20,000$.

Berger, Della Pietra & Della Pietra (1996) made the observation that the empirical distribution $\hat{p}(x)$ may be used as our marginal $m(x)$ on $\mathcal{X}$ to simplify the computation if we are only interested in the conditional probability $m(y|x)$ rather than the joint one $m(x, y)$. Equations (2.24) and (2.25) can thus be approximated as

$$m[g_k] = \sum_{(x,y)} m(x, y) \cdot g_k(x, y) \approx \sum_{(x,y): y_k(x,y)=1} \hat{p}(x) \cdot m(y|x), \text{ for } k = 1, \cdots, K$$

and

$$m_j[g_k] = \sum_{(x,y): y_k(x,y)=j} m(x, y) \cdot g_k(x, y) \approx \sum_{(x,y): y_k(x,y)=j} \hat{p}(x) \cdot m(y|x), \text{ for } k = 1, \cdots, K,$$

respectively. This implementation takes only $O(|\mathcal{X}| \cdot |\mathcal{Y}|)$ time where the set of “seen” histories $\mathcal{X}$ is much smaller than $\mathcal{X}$, and its size $|\mathcal{X}|$ is strictly bounded by the amount of training data.

In equations (2.13) and (2.14), $m(y|x)$ is uniquely decided by $\alpha$’s that are the only unknown parameters of the model to be trained. We take advantage of the fact that $g_k(x, y)$ is either 0 or 1, and we obtain the GIS equation for updating $\alpha_k(x, y)$ as

$$\alpha_k^{(n+1)} = \alpha_k^{(n)} \cdot \frac{a_k}{m[g_k]}$$

where $m[g_k]$ is the expectation of $g_k$.

Applying the IIS algorithm, the update equation (2.26) for individual $\alpha$’s is replaced by

$$\alpha_k^{(n+1)} = \alpha_k^{(n)} \cdot u_i$$

where $u_i$ is the solution to the simultaneous equations

$$\sum_{j=0}^{\max g_k} m_j[g_k] \cdot u_i^j = a_k,$$  \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} (2.28)$$

and

$$m_j[g_k] = \sum_{(x,y): y_k(x,y)=j} \hat{p}(x) \cdot m(y|x) \cdot g_k(x, y).$$

(2.29)
Since IIS converges faster than GIS, we use IIS to update $\alpha$’s in our work.

### 2.5.1 Computing the Normalization Factors

We need first to calculate the normalization factor $z(x)$ for each history according to (2.15) in order to compute $m(y|x)$, and then compute all the feature expectations $m[g_k]$. We introduce the state-of-the-art approach (as used before this dissertation) for computing $z$ in this section and for computing $m[g_k]$ in the next section.

In the straightforward implementation (2.15), it takes $O(|Y|)$ time to compute the normalization factor $z(x)$ for each $x \in \hat{X}$. IBM researchers treat the marginal features and other features in different ways, and take advantage of marginal features (Jelinek, 1997). Feature functions in $G$ are partitioned into two subsets: the marginal-feature subset $G_m$ that contains all features independent of history, and the conditional-feature subset $G_c$ that contains the rest of features. We also write marginal features $g_k(x, y)$ as $g_k(y)$ for the sake of simplicity. For each “history” $x$, the vocabulary $V$ is split into subsets $Y_x$ and $V - Y_x$ where

$$Y_x = \{ y : g_k(x, y) = 1 \text{ for some } g_k \in G_c \},$$

where $Y_x$ contains all $y$’s that have a conditional constraint in the given context $x$. Let

$$I(y) = \{ k : g_k(y) = 1 \}$$

be the indices corresponding to marginal (unigram) constraints\(^9\) on $y$. Setting

$$r(y) = \prod_{k \in I(y)} \alpha_k$$

to be the factor corresponding to unigram constraints on the word $y$, the calculation of $z(x)$ is simplified as follows

$$z(x) = \sum_{y \in Y} r(y) + \sum_{y \in Y_x} (r(x, y) - r(y)).$$

\(^9\)In the case of the basic N-gram model, there is only one such $k$ for each $y$. However, in some language models, such as the topic-sensitive model described later, there can be more than one such $k$. 
It should be noted that the first term in (2.30) can be pre-calculated once in $O(|\mathcal{Y}|) = O(|V|)$ time for all seen $x$’s, and the second term requires $O(|\hat{X}| \cdot C)$ time for all $x$, where $C$ is the average size of $Y_x$, i.e.,

$$C = \frac{1}{|\hat{X}|} \sum_{x \in \hat{X}} |Y_x|.$$  

The total running time for all $z(x)$’s is $O(|\mathcal{Y}| + |\hat{X}| \cdot C)$. Typically, $C$ is two orders of magnitude smaller than the vocabulary size $|\mathcal{Y}|$, resulting in a considerable computational speed-up relative to $O(|\hat{X}| \cdot |\mathcal{Y}|)$. This implementation has been called unigram-caching in the literature.

Unigram-caching, however, is still impractical for a large corpus, especially when constraints other than N-grams are considered, where the number of distinct histories increases by several orders of magnitude and $C$, which is actually a function of the vocabulary size $|V|$ and the size $L$ of the training corpus, also becomes extremely large. Obviously, the most computationally expensive part in (2.30) is in the second sum.

### 2.5.2 Computing Feature Expectations

The other half of computation required for maximum entropy modeling is to calculate the expectations of our features with respect to the joint model

$$m[g_k] = \sum_{\langle x, y \rangle; g_k(x, y) = 1} \hat{p}(x) \cdot m(y|x).$$

(2.31)

For each feature $g_k$ we need to list all $< x, y >$ for which $g_k$ is active and $m(y|x)$ is nonzero. If the feature $g_j$ is a marginal one, it has $|\hat{X}|$ nonzero $m(y|x)$’s. The total calculation for all $m[g_k(y)]$’s is $O(|\hat{X}| \cdot |\mathcal{Y}|)$ since the number of marginal features is $|\mathcal{Y}|$.

It should be noted that for each marginal $g_j$, there exists a $y$ such that $g_j(x, y) = 1$ for all $x \in \hat{X}$. But for the same $y$, only a few $x$’s have nonzero conditional features $g_k(x, y)$’s. We can take advantage of this to reduce the computation as described
\[ m[g_j] = \sum_{x \in \hat{X}} \hat{p}(x) \cdot m(y|x) \]
\[ = \alpha(y) \sum_{x \in \hat{X}} \alpha(x, y) \cdot \frac{\hat{p}(x)}{z(x)} \]
\[ = \alpha(y) \sum_{x \in \hat{X}} (\alpha(x, y) - 1) \cdot \frac{\hat{p}(x)}{z(x)} + \alpha(y) \sum_{x \in \hat{X}} \frac{\hat{p}(x)}{z(x)} \]  
(2.32)

where \( y \) can be regarded as fixed in the above equations. Now the running time is \( O(|\hat{X}| \cdot \mathcal{C} + |\mathcal{Y}|) \)\(^{10}\) instead of \( O(|\hat{X}| \cdot |\mathcal{Y}|) \). It should be noted that amount of the computation for the normalization factors and that for the estimate of parameters is the same. The former is a summation for each seen history over all words following that history in the training data, while the latter is a summation for each word over all histories seen preceding it.

More efficient training methods for computing both will be discussed in the next chapter.

### 2.5.3 Updating \( \alpha \) by Newton’s Method

The last step in training is to find the solution \( u_k \) to equations (2.28) and then to update the corresponding \( \alpha' \) by (2.27). We use Newton’s method to find the root of (2.28). We rewrite Equation (2.28) as

\[ f_k(u) = \sum_{j=0}^{\max g_k} m_j[g_k] \cdot u^j - a_k, \text{ for } k = 1, \cdots, K. \]

The following algorithm gives a numerical solution for \( f(u) = 0 \).

**Algorithm (Newton’s Method)**

**Initial Step:** Set initial value \( u_0 \) for \( u \) and a small number \( \epsilon > 0 \).

**Iteration Steps:** While \( |f(u)| \geq \epsilon \) do

\(^{10}\sum_{x \in \hat{X}} \frac{p(x)}{z(x)} \) can be pre-calculated in time of \( O(|\hat{X}|) \) so the total running time for \( \alpha(y) \sum_{x \in \hat{X}} \frac{\hat{p}(x)}{z(x)} \) is \( O(|\hat{X}| + |\mathcal{Y}|) \). Furthermore \( g_k(x, y) = 1 \) only for \( <x, y> \) active, so the computation for all \( \alpha(y) \sum_{x \in \hat{X}} (\alpha_k^{g_k(x, y)} - 1) \cdot \frac{\hat{p}(x)}{z(x)} \) is \( O(|\hat{X}| \cdot \mathcal{C}) \).
\[ \delta = -\frac{f'(u)}{f(u)}, \]
\[ u = u - \delta. \]

Readers may refer to Numerical Recipes (2002) (or any textbook for numerical analysis) for details about Newton’s method.