6 The Maximum Entropy Model

Zhang Le <zhang.le.mail@gmail.com>

Abstract

This note describes the Maximum Entropy Model (ME or MaxEnt for short), an expressive statistical learning framework. The note can serve as a quick reference to the ME model and the related algorithms, which may be helpful to those who are interested in the theoretical aspect of ME model, or who wants to implement such a model on computer but are somewhat forgetful of technical details (such as the author). For those who want a gentle introduction to Maximum Entropy Model, Adwait Ratnaparkhi's paper [Rat97] provides a tender, painless starting point.

Disclaims: The author does not claim that the content is complete in any sense, nor are the technique details 100% free of bugs. Comments and criticism are always welcome.

The latest version of this note is available from the author's homepage

updated on 20060810:

1. This file has been on my hard disk for nearly two years and I think I'd rather put it online than wait it to be finished
2. There must be some errors so if you doubt something, very likely you are right. Your comments are mostly welcomed at my email address above
3. This is part of a longer file so I omit the references to avoid putting a 20 page reference. I apologies for any inconvenience caused. If you are fancy then check out the MaxEnt page on my website.

6.1 The Modelling Problem

Many problems in machine learning involves modelling a set of sample data with a statistical model. Given some training data $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|D|}\}$, where $\mathbf{x}_i \in \mathcal{X} \in \Re^n$ is a real-value vector[1],$\mathcal{X}$ is a finite alphabet. Without loss of generality, we further assume the data was generated i.i.d by an unknown distribution $q(\mathbf{x})$. Our task is to build a statistical model p that best accounts for the data in *some sense*. We call this process *learning from data* or *inference from data*.

For simplicity, we restrict our discussion to joint Maximum Entropy model. The reader can find the detail treatment on Conditional Maximum Entropy model in the appendixes, which receives a wider attention in Natural Language Processing tasks. Also, we only discuss discrete ME model. The continuous model can be derived similarly.

6.1.1 Entropy of a Probabilistic Distribution

Entropy is a measure of uncertainty of a probabilistic distribution introduced in the pioneering paper of CE Shannon [Sha48] (reprinted in [SW93]).

Given a probabilistic distribution $P(\mathbf{x}) = \{p(\mathbf{x}_1), p(\mathbf{x}_2), \ldots, p(\mathbf{x}_n)\}$,$\mathbf{x} \in \mathcal{X}$, the entropy of p is defined to be:

$$H(p) = -\sum_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}) \tag{6.1}$$

The higher $H(p)$ is, the more uncertainty p has, and the more information p contains. It is easy to see $H(p)$ achieves maximum when $p(\mathbf{x})$ is uniform [some simple proof here].

6.1.2 The Maximum Entropy Principle

In his famous 1957 paper [Jay57], Jaynes summarised the maximum entropy principle as:

Information theory provides a constructive criterion for setting up probability distributions on the basis of partial knowledge, and leads to a type of statistical inference which is called the maximum entropy estimate. It is least biased estimate possible on the given information; i.e., it is maximally noncommittal with regard to missing information.

That is to say, when characterising some unknown events with a statistical model, we should always choose the one that has maximum entropy subject to known constraints.

[1] Throughout this paper, we denote a vector in bold font like $\mathbf{x} \in \Re^n$,and a scalar in normal font such as $y \in \Re$.

6.2 Derive Parametric Form of Joint MaxEnt Model

Given a reference distribution: $\tilde{p}(\mathbf{x}) = \{\tilde{p}(\mathbf{x}_1), \tilde{p}(\mathbf{x}_2), \ldots, \tilde{p}(\mathbf{x}_n)\}$ and a set of constraints in the form of *potential functions* $f_i(\mathbf{x})$, we now derive the parametric form of a probabilistic model $p(\mathbf{x})$ satisfying certain constraints while maximising its entropy.

The constraint we are interested in is to require the expectation of a set of features $F = \{f_1, f_2, \ldots, f_k\}$ under the model equals to the expectation found in the data:

$$\sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) f_i(\mathbf{x}), 1 \leq i \leq k \tag{6.2}$$

The following equations are what we know, and therefore the model must satisfy:

$$p^* = \arg\max_p - \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}) \tag{6.3}$$

$$\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) = 1 \tag{6.4}$$

$$\sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) f_i(\mathbf{x}) \qquad 1 \leq i \leq k \tag{6.5}$$

Eq. (6.3) is required to form a probabilistic distribution, and eq. (6.3) is the maximum entropy criteria. Basically, we want to find a p^* maximises (6.3) given constraints (6.3) and (6.5). Equations (6.4), (6.5) and (6.3) together form a constrained optimisation problem. We can solve it using well established techniques like Lagrange methods.

Here we introduce Lagrange multiples λ_i and μ, and form the Lagrange function.

$$L(p, \lambda, \mu) = -\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i \left(\sum_{\mathbf{x}\in\mathcal{X}} (p(\mathbf{x}) f_i(\mathbf{x}) - \tilde{p}(\mathbf{x}) f_i(\mathbf{x})) \right) + \mu \left(\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) - 1 \right) \tag{6.6}$$

Taking partial difference with respect to $p(\mathbf{x})$ we get:

$$\frac{\partial L(p, \lambda, \mu)}{\partial p(\mathbf{x})} = -\log p(\mathbf{x}) - 1 + \sum_i \lambda_i f_i(\mathbf{x}) + \mu \tag{6.7}$$

Let (6.7) $= 0$ we have:

$$\log p(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) + \mu - 1 \tag{6.8}$$

taking $\exp(\cdot)$ on both sides:

$$p(\mathbf{x}) = \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) + \mu - 1 \right) \tag{6.9}$$

$$= \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \right) \cdot \exp(\mu - 1) \tag{6.10}$$

plug (6.10) into (6.4) we get:

$$\sum_{\mathbf{x}\in\mathcal{X}} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \right) \cdot \exp(\mu - 1) = 1 \tag{6.11}$$

dividing both sides by $\exp(\mu - 1)$:

$$\sum_{\mathbf{x}\in\mathcal{X}} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \right) = \frac{1}{\exp(\mu - 1)} \tag{6.12}$$

rearranging:

$$\exp(\mu - 1) = \frac{1}{\sum_{\mathbf{x}\in\mathcal{X}} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \right)} \tag{6.13}$$

substitute (6.13) into (6.10) we get:

$$p(\mathbf{x}) = \frac{1}{\sum_{\mathbf{x}\in\mathcal{X}} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})\right)} \cdot \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})\right) \tag{6.14}$$

Now introduce the normalisation term $Z(\mathbf{x}) = [\sum_{\mathbf{x}\in\mathcal{X}} \exp(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}))]^{-1}$ we can write $p(\mathbf{x})$ as:

$$\boxed{p(\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \cdot \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})\right)} \tag{6.15}$$

Eq. (6.15) is the parametric form of the joint Maximum Entropy model. From those steps we can see the parameter λ_i , which can be interpreted as the weight of feature f_i , is actually the Lagrange multiplier for the ith constraint equation in eq.(6.5)

If our objective criteria is not maximum entropy but minimum divergence between some *prior* distribution, e.g., $q_0(\mathbf{x})$, we can seek to minimise:

$$D(p||q_0) = \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q_0(\mathbf{x})} \tag{6.16}$$

$$= H(p, q_0) - H(p) \tag{6.17}$$

Here $D(p||q_0)$ is the Kullback-Leibler divergence between p and q_0. Using the same methods (Lagrange Multiples) we can get:

$$p(\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \cdot q_0(\mathbf{x}) \cdot \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})\right) \tag{6.18}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{X}} q_0(\mathbf{x}) \cdot \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})\right) \tag{6.19}$$

It is easy to see when $q_0(\mathbf{x})$ is uniform, we get maximum entropy solution (6.15).

6.3 Parameter Estimation (Extract Inference)

We first introduce the simple and fast gradient based methods. Then we discuss two iterative methods specially tailored for maximum entropy models, namely Generalized Iterative Scaling (GIS) and Improved Iterative Scaling (IIS). The iterative methods were once popular in the 1990s, but are a little out of date today.

6.3.1 Gradient Based Estimation

We start from Maximum Likelihood Estimation (MLE). Given data point $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, the likelihood[2] of model p_Λ on D given parameter Λ is:

$$L(D \mid \Lambda) = \prod_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x})^{C(\mathbf{x})} \tag{6.20}$$

Here $C(\mathbf{x})$ is the count of $\mathbf{x}$ in D.

It is usually more convenient to perform computation in log-space. And the log-likelihood of L is:

$$LL(D \mid \Lambda) = \sum_{\mathbf{x}\in\mathcal{X}} C(\mathbf{x}) \log p_\Lambda(\mathbf{x}) \tag{6.21}$$

$$\propto \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \log p_\Lambda(\mathbf{x}) \tag{6.22}$$

here $\tilde{p}(\mathbf{x}) = \frac{C(\mathbf{x})}{N}, N = \sum_{\mathbf{x}\in\mathcal{X}} C(\mathbf{x})$ Eq. (6.22) is the average per-symbol log-likelihood and differs from the real log-likelihood up to a constant N . In what follows, we define our objective function $L(\Lambda)$ to be:

$$L(\Lambda) = \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \log p_\Lambda(\mathbf{x}) \tag{6.23}$$

[2] Alternatively, if we index $\mathbf{x}$ over all instances in D we can write $L(D \mid \Lambda) = \prod_{\mathbf{x}'\in D} p_\Lambda(\mathbf{x}')$.

since this is the most widely form used in the literature.

We have:

$$L(\Lambda) = \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \log p_\Lambda(\mathbf{x}) \tag{6.24}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \log \left(\frac{1}{Z_\Lambda} \cdot \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \right) \right) \tag{6.25}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) - \log Z_\Lambda \right) \tag{6.26}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \cdot \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \log Z_\Lambda \tag{6.27}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) - \log Z_\Lambda \qquad \left(\text{because } \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \log Z_\Lambda = \log Z_\Lambda\right) \tag{6.28}$$

[the interpretation of the first and the second term]

Our goal now is to find a set of weights $\Lambda^* = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ that maximises eq (6.28). Fortunately, the objective function $L(\Lambda)$ is well behaved: it is convex in the k dimensional parameter space Λ. [to see why] So we can find the maximum point Λ^* by solving $\frac{\partial L(\Lambda)}{\partial \Lambda} = 0$.

Taking derivative of $L(\Lambda)$ with respect to λ_i

$$L'(\Lambda) = \frac{\partial L(\Lambda)}{\partial \lambda_i} = \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \frac{1}{Z_\Lambda} \sum_{\mathbf{x}\in\mathcal{X}} \left[\exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \right) \cdot f_i(\mathbf{x}) \right] \tag{6.29}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{X}} \left[\frac{1}{Z_\Lambda} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) \right) \cdot f_i(\mathbf{x}) \right] \tag{6.30}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) f_i(\mathbf{x}) \tag{6.31}$$

$$= E_{\tilde{p}}[f_i] - E_{p_\Lambda}[f_i] \tag{6.32}$$

Here $E_{\tilde{p}}[f_i] = \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x})$ denotes the expectation of feature f_i under "reference" distribution $\tilde{p}$ and $E_{p_\Lambda}[f_i] = \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) f_i(\mathbf{x})$ is the feature expectation under the model p_Λ.

Now we have objective function $L(\Lambda)$, its gradient $L'(\Lambda)$ with respect to λ_i, and a set of k parameters $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ to optimise. The problem of finding the optimal parameter Λ^* can be recast as a constraint optimisation problem: given L, G, and an initial guess of parameter $\Lambda^{(0)}$, find the optimised solution Λ^* in the parameter space that maximises L.

Several general gradient based optimise methods can be employed to find Λ^*, namely Conjugate Gradient (CG) and a quasi-Newton method called limited-memory BFGS algorithm (L-BFGS). These methods are general faster than the iterative alternative methods (see later sections) specially tailored for this problem.

6.3.2 Limited-memory BFGS Method

limited-memory BFGS algorithm (L-BFGS for short) is currently the most effective optimisation method for ME parameter estimation [Mal02]. The pseudo code of estimating ME parameter using L-BFGS is:

1. L = 0

- Initial Λ_0 to initial values (usually start from a uniform model (i.e. all $\lambda_i = 0$))
- repeat the following steps:
  - calculate $L_\Lambda^{(n)}$ and $L_\Lambda^{\prime(n)}$
  - $\Lambda^{(n+1)} = \text{L-BFGS}(L^{(n)}, L_\Lambda^{\prime(n)}, \Lambda^{(n)})$
- until: $|\ \frac{L^{(n+1)} - L^{(n)}}{L^{(n)}}\ | <$ predefined threshold

CG is very similar.

6.3.3 Iterative Scaling Methods

The idea behind iterative scaling methods is: given an initial model $\Lambda^{(0)}$, if we can find a serials of models $\Lambda^{(1)}, \Lambda^{(2)}, \ldots, \Lambda^{(n)}$ such that each new model $\Lambda^{(n+1)}$ is better than the previous model $\Lambda^{(n)}$ (in the sense of having a higher likelihood on the data), we are guaranteed to converge to the final model (since our model is convex).

We discuss IIS and GIS (and their variants) in this section.

6.3.4 Improved Iterative Scaling

Improved Iterative Scaling (IIS) was proposed by [PPL97]. The word "Improved" means IIS has a faster convergence rate than its ancestor GIS[3]. Suppose we have an initial model Λ and its successive model Λ' in a serials of iterations. The corresponding (averaged) log-likelihood objective functions are:

$$L(\Lambda) = \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \log p_\Lambda(\mathbf{x}) \tag{6.33}$$

and

$$L(\Lambda') = \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \log p_{\Lambda'}(\mathbf{x}) \tag{6.34}$$

We can write the difference $\Delta(\Lambda' \mid \Lambda)$ between $L(\Lambda')$ and $L(\Lambda)$ as:

$$\begin{aligned}
\Delta(\Lambda' \mid \Lambda) = L(\Lambda') - L(\Lambda) &= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \log \frac{p_{\Lambda'}(\mathbf{x})}{p_\Lambda(\mathbf{x})} && (6.35)\\
&= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \log \frac{\frac{1}{Z_{\Lambda'}} \exp\left(\sum_{i=1}^k \lambda_i f_i(\mathbf{x})\right)}{\frac{1}{Z_\Lambda} \exp\left(\sum_{i=1}^k \lambda_i f_i(\mathbf{x})\right)} && (6.36)\\
&= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \left(\sum_{i=1}^k \lambda_i f_i(\mathbf{x}) - \sum_{i=1}^k \lambda'_i f_i(\mathbf{x}) - \log \frac{Z_{\Lambda'}}{Z_\Lambda}\right) && (6.37)\\
&\quad \text{Let } \delta_i = \lambda'_i - \lambda_i, \text{we have} && (6.38)\\
\Delta(\Lambda' \mid \Lambda) = L(\Lambda') - L(\Lambda) &= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \left(\sum_{i=1}^k \delta_i f_i(\mathbf{x}) - \log \frac{Z_{\Lambda'}}{Z_\Lambda}\right) && (6.39)\\
&= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^k \delta_i f_i(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \cdot \log \frac{Z_{\Lambda'}}{Z_\Lambda} && (6.40)\\
&= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^k \delta_i f_i(\mathbf{x}) - \log \frac{Z_{\Lambda'}}{Z_\Lambda} && (6.41)
\end{aligned}$$

Applying $-\log \alpha \geq 1 - \alpha$ (true for all $\alpha > 0$) to establish a lower bound. (See appendix XXX for more on this

[3]Interestingly, in practice IIS was observed to be much slower than GIS. See [Mal02].

inequality). Here α is $Z_{\Lambda'}/Z_\Lambda$.

$$\begin{aligned}
\Delta(\Lambda' \mid \Lambda) = L(\Lambda') - L(\Lambda) &\geq \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \frac{Z_{\Lambda'}}{Z_\Lambda} && (6.42)\\
&= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \frac{\sum_{\mathbf{x}\in\mathcal{X}} \exp\left(\sum_{i=1}^{k} \lambda'_i f_i(\mathbf{x})\right)}{Z_\Lambda} && (6.43)\\
&\quad \text{now we replace } \lambda'_i \text{ with } \lambda_i + \delta_i \text{ in the last term} && (6.44)\\
&= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \frac{\sum_{\mathbf{x}\in\mathcal{X}} \exp\left(\sum_{i=1}^{k} (\lambda_i + \delta_i) f_i(\mathbf{x})\right)}{Z_\Lambda} && (6.45)\\
&= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \frac{\sum_{\mathbf{x}\in\mathcal{X}} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{k} \delta_i f_i(\mathbf{x})\right)}{Z_\Lambda} && (6.46)\\
&= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \sum_{\mathbf{x}\in\mathcal{X}} \frac{1}{Z_\Lambda} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})\right) \cdot \exp\left(\sum_{i=1}^{k} \delta_i f_i(\mathbf{x})\right) && (6.47)\\
&= \underbrace{\sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) \cdot \exp\left(\sum_{i=1}^{k} \delta_i f_i(\mathbf{x})\right)}_{\text{call this } A(\delta|\Lambda)} && (6.48)
\end{aligned}$$

$A(\delta \mid \Lambda)$ is the increase in log-likelihood of $p_{\Lambda'}$ over p_Λ.

If we can make $A(\delta \mid \Lambda) \geq 0$, we can be more close to the final Λ^* in each iteration. Unfortunately, $\frac{\partial A(\delta|\Lambda)}{\partial \delta_i}$ contains $\{\delta_1, \delta_2, \ldots, \delta_k\}$, so we can not get a close form solution.

The trick here is to rewrite $A(\delta \mid \Lambda)$ in a slightly different way to use Jensen's inequality. Let $f^\#(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x})$, i.e. the feature sum over $\mathbf{x}$. Thus $\frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})}$ is a probabilistic distribution over f_i. We have:

$$A(\delta \mid \Lambda) = \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) \cdot \exp\left(\sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})} \cdot f^\#(\mathbf{x}) \cdot \delta_i\right) \tag{6.49}$$

Since $\frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})}$ is a p.d.f., we can apply Jensen's inequality (see appendix XXX for more on this inequality), which states:

if $p(\mathbf{x})$ is a p.d.f. then:

$$\exp\left(\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) q(\mathbf{x})\right) \leq \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \exp\left(q(\mathbf{x})\right) \tag{6.50}$$

In the last term of eq. (6.49) we can think $\mathbf{x}$ as i, $p(\mathbf{x})$ as $\frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})}$, and $q(\mathbf{x})$ as $f^\#(\mathbf{x})\delta_i$. So:

$$\exp\left(\sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})} f^\#(\mathbf{x})\delta_i\right) \leq \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})} \exp\left(f^\#(\mathbf{x})\delta_i\right) \tag{6.51}$$

Therefore,

$$A(\delta \mid \Lambda) \geq \underbrace{\sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \cdot \sum_{i=1}^{k} \delta_i f_i(\mathbf{x}) + 1 - \sum_{\mathbf{x}\in\mathcal{X}} \cdot \sum_{i=1}^{k} \left(\frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})} \cdot \exp(f^\#(\mathbf{x})\delta_i)\right)}_{\text{call this } B(\delta|\Lambda)} \tag{6.52}$$

$B(\delta \mid \Lambda)$ is a new, but less tight, lower bound on the change in log-likelihood. See [PPL97] for extra conditions on the continuity and derivative of the lower bound, in order to guarantee convergence. taking derivative with respect to δ_i:

$$\begin{aligned}
\frac{\partial B(\delta \mid \Lambda)}{\partial \delta_i} &= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) \left(\frac{f_i(\mathbf{x})}{f^\#(\mathbf{x})} \cdot \exp\left(f^\#(\mathbf{x})\delta_i\right) \cdot f^\#(\mathbf{x})\right) && (6.53)\\
&= \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) \left(\exp(f^\#(\mathbf{x})\delta_i) f_i(\mathbf{x})\right) && (6.54)
\end{aligned}$$

Now δ_i appears alone, we can then solve the equation $\frac{\partial B(\delta|\lambda)}{\partial \delta_i} = 0$ using some one dimension root-finding procedure such as Newton-Raphson method. Once $\delta_i's$ are obtained, we can update Λ to $\Lambda' = \Lambda + \delta$. We can repeatedly perform the iteration until the model converge.

Here's the pseudo code for IIS based estimation:

- L = 0
- Initial Λ^0 to initial values (usually start from a uniform model (i.e. all $\lambda_i = 0, 1 \leq i \leq k$))
- repeat the following steps:
  - Solve $\frac{\partial B(\delta|\lambda)}{\partial \delta_i} = 0$
  - Update $\lambda_i^{(n+1)} = \lambda_i^{(n)} + \delta_i$
- until: $|\ \frac{L^{(n+1)} - L^{(n)}}{L^{(n)}}\ |<$ predefined threshold

6.3.5 Generalized Iterative Scaling

Now we consider an interesting case if IIS where $f^{\#}(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x}) = C$, and C is a constant, i.e., the sum of active features of each $\mathbf{x}$ is a constant.

Replacing $f^{\#}(\mathbf{x})$ with C in (6.54) we have:

$$\begin{aligned}
\frac{\partial B(\delta \mid \Lambda)}{\partial \delta_i} &= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{X}} p_\Lambda(\mathbf{x}) \exp(C\delta_i) f_i(\mathbf{x}) && (6.55)\\
&= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \exp(C\delta_i) \sum_{\mathbf{x} \in \mathcal{X}} p_\Lambda(\mathbf{x}) f_i(\mathbf{x}) && (6.56)\\
&= E_{\tilde{p}}[f_i] - \exp(C\delta_i) E_{p_\Lambda}[f_i] && (6.57)
\end{aligned}$$

Setting eq (6.57) to 0, we get:

$$\exp(C\delta_i) = \frac{E_{\tilde{p}}[f_i]}{E_{p_\Lambda}[f_i]} \tag{6.58}$$

and

$$\delta_i = \frac{1}{C} \log \frac{E_{\tilde{p}}[f_i]}{E_{p_\Lambda}[f_i]} \tag{6.59}$$

which is a close form solution. This is the famous GIS update rule.

So how about the case where $f^{\#}(\mathbf{x})$ is not a constant? In order to make use of (6.59), which requires $f^{\#}(\mathbf{x})$ to be a constant, we can let $C = max_{\mathbf{x} \in \mathcal{X}} \sum_{i=1}^{k} f_i(\mathbf{x})$ and add a *correction* feature:

$$f_c(\mathbf{x}) = C - \sum_{i=1}^{k} f_i(\mathbf{x}) \tag{6.60}$$

so that $f^{\#}(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x}) + f_c(\mathbf{x}) = C$ is still a constant. In practice, C is usually approximated from training data $\mathcal{D}$. Since it is infeasible to enumerate all $\mathbf{x} \in \mathcal{X}$ in many applications. And we can still use eq (6.59) to update δ_i. This is exactly the idea behind the Generalized Iterative Scaling algorithm [DR72]. From the above we can see that GIS is a specialised version of IIS, which appears approximately two decades later.

The ME model with a correction feature now looks like:

$$\begin{aligned}
p(\mathbf{x}) &= \frac{1}{Z} \cdot \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) + \lambda_c f_c(\mathbf{x})\right) && (6.61)\\
&= \frac{1}{Z} \cdot \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) + \lambda_c (C - \sum_{i=1}^{k} f_i(\mathbf{x}))\right) && (6.62)\\
& \text{where } Z \text{ is:} && (6.63)\\
Z &= \sum_{\mathbf{x} \in \mathcal{X}} \exp\left(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}) + \lambda_c (C - \sum_{i=1}^{k} f_i(\mathbf{x}))\right) && (6.64)
\end{aligned}$$

The expectation of the correction feature under $\tilde{p}$ and p_Λ are:

$$E_{\tilde{p}}[f_c] = \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) \left(C - \sum_{i=1}^{k} f_i(\mathbf{x}) \right) \tag{6.65}$$

$$E_{p_\Lambda}[f_c] = \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) \left(C - \sum_{i=1}^{k} f_i(\mathbf{x}) \right) \tag{6.66}$$

with these added complexity in mind, we now give the pseudo code for GIS algorithm:

- L = 0
- Set $C = \max_{\mathbf{x}\in\mathcal{D}} \sum_{i=1}^{k} f_i(\mathbf{x})$, and add the $(k+1)^{th}$ feature $f_c(\mathbf{x}) = C - \sum_{i=1}^{k} f_i(\mathbf{x})$ to the model. The corresponding parameter is λ_c (the $(k+1)^{th}$ parameter in Λ).
- Initial Λ^0 to initial values (usually start from a uniform model (i.e. all $\lambda_i = 0, 1 \le i \le k+1$))
- repeat the following steps:
  - For each λ_i in $\Lambda^{(n)}$:
    - Update $\delta_i^{(n+1)} = \lambda_i^{(n)} + \frac{1}{C} \log \frac{E_{\tilde{p}}[f_i]}{E_{p_{\Lambda^{(n)}}}[f_i]}$
- until: $| \frac{L^{(n+1)} - L^{(n)}}{L^{(n)}} |<$ predefined threshold (*)

It is worth mentioning that the update of the $(k+1)^{th}$ correction feature δ_c should be calculated with eq (6.66) and eq (6.61), which must be computed separately.

In theory, GIS has a slower converge rage compared to IIS, since its step coefficient (or step size) $1/C$ depends on the largest $f^{\#}(\mathbf{x})$, which is not as optimal as the update rule of IIS. Interestingly, in practice IIS was observed much slower than GIS [Mal02], mostly due to the computation of Newton-Rapson method involved in finding each update value δ_i.

6.3.6 Correction-Free GIS

Although the introduction of the correction feature f_c in GIS simplifies the update rule, it does not simplifies the implementation of the algorithm. Since the correction feature is different from all the other features in the model, it's expectation $E_{p_\Lambda}[f_c]$ and the corresponding parameter update δ_{f_c} must be computed separately using eq (6.66), which makes the implementation a bit more complex. Since no additional information is added to the model by the introduction of the correction feature (f_c only depends on the other features), can we get rid of it and make the already simple algorithm simpler?

The answer is "yes". In the footnote of the [Goo02] describing a faster GIS variant called SCGIS, Goodman noticed the correction feature of GIS is not necessary since we can think it as a special feature in the ME model but with zero weight, hence can be omitted safely. Later Curran and Clark concrete Goodman's idea and sketch a proof of the convergence of the correction-free GIS in the appendix of their paper [CC03].

The basic idea is to image adding a correction feature f_c as above with zero weight $\lambda_c = 0$, and never bother updating it (implying $\delta_i^{(i)} = 0$ all the time in training). Since $\lambda_c = 0$, we can remove it from eq (6.61). However, because $f^{\#}(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x}) + f_c(\mathbf{x}) = C$ is still a constant, the GIS update rule (6.59) can still converge!

So the pseudo code of the correction-free GIS looks like:

- L = 0
- Set $C = max_{\mathbf{x}\in\mathcal{D}} \sum_{i=1}^{k} f_i(\mathbf{x})$.
- Initial Λ^0 to initial values (usually start from a uniform model (i.e. all $\lambda_i = 0, 1 \le i \le k$))
- repeat the following steps:
  - For each λ_i in $\Lambda^{(n)}$:
    - Update $\delta_i^{(n+1)} = \lambda_i^{(n)} + \frac{1}{C} \log \frac{E_{\tilde{p}}[f_i]}{E_{p_{\Lambda^{(n)}}}[f_i]}$
- until: $| \frac{L^{(n+1)} - L^{(n)}}{L^{(n)}} |<$ predefined threshold (*)

Which is almost the same as the original GIS algorithm but without the need of updating correction feature.

6.4 Regularisation

Our parameter estimation methods described so far are are based on the Maximum Likelihood Estimate (MLE). One weakness of MLE training is that the trained model can be so faithful to the training data that it does not generalise well on unseen data. This is often called *over-fitting*. In practice over-fitting can occur when training a too complex model on a relatively small data set (e.g. the number of free parameters is larger than the size of the data). One way to avoid over-fitting is to use a separate held-out data and stop training when testing accuracy on the held-out data starts to drop. This solution is cumbersome. Since the testing accuracy on the held-out data may continue to increase after several temporary drops. If we stop training prematurely, we get a under-trained model, which is sub-optimal.

Another way to alleviate the over-training problem is to optimise a different objective function, which can incorporate some *prior* knowledge about the parameters. First, let's recall the Bayes rule:

$$P(\Lambda \mid D) = \frac{P(D \mid \Lambda) \cdot P(\Lambda)}{P(D)} \tag{6.67}$$

$P(\Lambda \mid D)$ is called the *posterior* of Λ given D, while $P(D \mid \Lambda)$ and $P(\Lambda)$ are called the *likelihood* of Λ on D and the *prior* of Λ respectively. Remembering that $P(D)$ (also called evidence) is independent of Λ and is just a normalisation term, we may write:

$$P(\Lambda \mid D) \quad \propto \quad P(D \mid \Lambda) \cdot P(\Lambda) \tag{6.68}$$

$$posterior \quad \propto \quad likelihood \cdot prior \tag{6.69}$$

From a Bayesian's point of view, although we do not know Λ before the observation of D, we can (nevertheless) assign a prior for the (so far) unknown Λ as $P(\Lambda)$. The prior $P(\Lambda)$ reflects our prior knowledge on Λ. It can be uniform, Gaussian, Gama distribution or any other (valid) probabilistic distribution. After observing D, we can use (6.67) to update our belief to obtain the posterior $P(\Lambda \mid D)$.

If we maximise the posterior $P(\Lambda \mid D)$ instead of the likelihood $P(D \mid \Lambda)$, we get a *Maximum A Posterior* estimate Λ_{MAP} that is different from the estimate obtained from MLE estimation Λ_{ML}. Interestingly, if the prior is uniform or flat (i.e. totally non-informative), then the MAP estimate is identical to the MLE estimator.

The use of a prior effectively limit the distribution of Λ to be certain form. It penalises more to those parameters that drift far away from the mean, hence prevent over-trained weights.

A common choice of prior distribution is the Gaussian distribution:

$$P(\lambda) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(\lambda - \mu)^2}{2\sigma^2}\right) \tag{6.70}$$

Here μ is mean and σ^2 is variance. μ determines the centre of the p.d.f. and σ^2 control the width of the distribution.

There are two primary reasons to use a Gaussian prior:

- In many cases it is a reasonable appropriate model of real data (by *Central Limit Theorem*).
- It is analytically very convenient.

Recall the original likelihood objective function is:

$$L(\Lambda \mid D) = \prod_{\mathbf{x} \in \mathcal{X}} p_\Lambda(\mathbf{x})^{C(\mathbf{x})} \tag{6.71}$$

With a Gaussian prior, our new posterior objective function is:

$$P(\Lambda \mid D) \propto P(D \mid \Lambda) \cdot P(\Lambda) \tag{6.72}$$

with

$$P(\lambda_i) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(\lambda_i - \mu)^2}{2\sigma^2}\right) \tag{6.73}$$

Assuming λ_i is i.i.d., the log MAP objective function is:

$$L_{MAP}(\Lambda \mid D) \quad = \quad L_\Lambda(D) + \log P(\Lambda) \tag{6.74}$$

$$= \quad L_\Lambda(D) - \sum_{i=1}^{k} \frac{(\lambda_i - \mu)^2}{2\sigma^2} - \sum_{i=1}^{k} \log(\sqrt{2\pi}\sigma) \tag{6.75}$$

$$= \quad L_\Lambda(D) - \sum_{i=1}^{k} \frac{(\lambda_i - \mu)^2}{2\sigma^2} - k \log \sqrt{2\pi}\sigma \tag{6.76}$$

Of course, the new objective function requires some changes to the parameter estimation methods, which we will discuss shortly.

6.4.1 Gradient-based Training with Regularisation

With the new MAP objective function in mind, the gradient function used in gradient-based training (Sec 6.3.1) becomes:

$$L'_{MAP}(\Lambda \mid D) = \frac{\partial L_{MAP}(\Lambda \mid D)}{\partial \lambda_i} \tag{6.77}$$

$$= \frac{\partial L_{\Lambda}(D)}{\partial \lambda_i} - \frac{\lambda_i - \mu}{\sigma^2} \tag{6.78}$$

$$= E_{\tilde{p}}[f_i] - E_{p_\Lambda}[f_i] - \frac{\lambda_i - \mu}{\sigma^2} \tag{6.79}$$

Applying regularisation to gradient-based training is a simple matter of replacing $L(\Lambda)$ (6.23) and $L'(\Lambda)$ (6.32) with the corresponding MAP functions $L_{MAP}(\Lambda)$ (6.76) and $L'_{MAP}(\Lambda)$ (6.79).

6.4.2 Iterative Scaling Training with Regularisation

In order to use the Gaussian prior in the Iterative Scaling training described in Section (6.3.3), recall the improvement in the log-likelihood objective function of two successive models $\Lambda$and Λ' can be written as:

$$\Delta(\Lambda' \mid \Lambda) = L(\Lambda') - L(\Lambda) = \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \log \frac{p_{\Lambda'}(\mathbf{x})}{p_{\Lambda}(\mathbf{x})} \tag{6.80}$$

Under the Gaussian MAP criteria, our new improvement is:

$$\Delta_{MAP}(\Lambda' \mid \Lambda) = L_{MAP}(\Lambda') - L_{MAP}(\Lambda) \tag{6.81}$$

$$= \sum_{\mathbf{x} \in \mathcal{X}} \tilde{p}(\mathbf{x}) \log \frac{p_{\Lambda'}(\mathbf{x})}{p_{\Lambda}(\mathbf{x})} - \sum_{i=1}^{k} \frac{(\lambda'_i - \mu)^2 - (\lambda_i - \mu)^2}{2\sigma^2} \tag{6.82}$$

$$= \Delta(\Lambda' \mid \Lambda) - \sum_{i=1}^{k} \frac{(\lambda'_i - \mu)^2 - (\lambda_i - \mu)^2}{2\sigma^2} \tag{6.83}$$

$$= \Delta(\Lambda' \mid \Lambda) - \sum_{i=1}^{k} \frac{(\lambda_i + \delta_i - \mu)^2 - (\lambda_i - \mu)^2}{2\sigma^2} \tag{6.84}$$

$$= \Delta(\Lambda' \mid \Lambda) - \sum_{i=1}^{k} \frac{\delta_i^2 + 2\delta_i\lambda_i - 2\delta_i\mu}{2\sigma^2} \tag{6.85}$$

Similarly, we can get new lower bounds of $\Delta_{MAP}(\Lambda' \mid \Lambda)$:

$$A_{MAP}(\delta \mid \Lambda) = A(\delta \mid \Lambda) - \sum_{i=1}^{k} \frac{\delta_i^2 + 2\delta_i\lambda_i - 2\delta_i\mu}{2\sigma^2} \tag{6.86}$$

then

$$B_{MAP}(\delta \mid \Lambda) = B(\delta \mid \Lambda) - \sum_{i=1}^{k} \frac{\delta_i^2 + 2\delta_i\lambda_i - 2\delta_i\mu}{2\sigma^2} \tag{6.87}$$

Differential $B_{MAP}(\delta \mid \Lambda)$ with respect to δ_i :

$$\frac{\partial B_{MAP}(\delta \mid \Lambda)}{\partial \delta_i} = \frac{\partial B(\delta \mid \Lambda)}{\partial \delta_i} - \frac{\delta_i + \lambda_i - \mu}{\sigma^2} \tag{6.88}$$

Setting eq (6.88) to 0 and solve for δ_i using Newon-Raphson method, we get the new IIS update under Gaussian prior.

For GIS update, we have:

$$\frac{\partial B_{MAP}(\delta \mid \Lambda)}{\partial \delta_i} = \sum_{\mathbf{x}\in\mathcal{X}} \tilde{p}(\mathbf{x}) f_i(\mathbf{x}) - \exp(C\delta_i) \sum_{\mathbf{x}\in\mathcal{X}} p_\Lambda(\mathbf{x}) f_i(\mathbf{x}) - \frac{\delta_i + \lambda_i - \mu}{\sigma^2} \tag{6.89}$$

$$= E_{\tilde{p}}[f_i] - \exp(C\delta_i) E_{p_\Lambda}[f_i] - \frac{\delta_i + \lambda_i - \mu}{\sigma^2} \tag{6.90}$$

Unfortunately, we can not get δ_i directly by setting eq (6.90) to 0, as we have done for eq (6.57). Instead we need to resort to Newton-Raphson method again to solve eq (6.90) $= 0$ for δ_i.

There is no principled way of choosing μ and σ. In practice the following prior with zero means work fairly well:

$$P(\lambda_i) = \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{\lambda_i^2}{2\sigma^2}\right) \tag{6.91}$$

6.5 Parameter Estimation with Stochastic Sampling (Approximate Inference)

In many real-world applications[4], the sample space $\mathcal{X} \in \Re^n$ is infinite or finite but too large to enumerate. Training such model using methods described in Section 6.3 is intractable. Because in each iteration of training we need to evaluate $E_p[f_i] = \sum_{x\in\mathcal{X}} p(x) f_i(x)$, which requires a summation of all $x's$ in $\mathcal{X}$, the computational cost is often too high to make naive training practical.

Fortunately, stochastic sampling provides an affordable means to solve this problem. The basic idea of sampling-based training is: in each round of training, if we can generate a set of R representative samples from p: $\{x_1, x_2, \cdots, x_R\}$, we can estimate $E_p[f_i]$ from those samples

$$E_p[f_i] \approx \frac{1}{R} \sum_{r=1}^{R} f_i(x_r) \tag{6.92}$$

Because when R goes to ∞ , by law of large number, the estimated $\hat{E}_p[f_i]$ will converge to the true expectation.

However, generating samples from an exponential distribution such as p is a difficult task. This problem can be circumvented by drawing samples from an easy-to-sample proposal distribution q in such a way that the samples mimic samples from p. Several sophisticated sampling methods can be used for this purpose, such as Gibbs Sampling, Metropolis-Hastings Sampling, and Importance Sampling. For the rest of this section, we discuss the use of importance sampling in ME training.

6.5.1 Importance Sampling

The core problem in training a ME model is the computation of $E_p[f_i]$. Notice that:

$$E_p[fi(\mathbf{x})] = \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) f_i(\mathbf{x}) \tag{6.93}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}} q(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} f_i(\mathbf{x}) \tag{6.94}$$

$$= E_q[\frac{p(\mathbf{x})}{q(\mathbf{x})} f_i(\mathbf{x})] \tag{6.95}$$

And it is easy to see this estimator is unbiased. [Some variance analysis of importance sampling]

Eq (6.95) still has one problem: we can not evaluate $p(\mathbf{x})$ since the normalisation constant Z is unknown in training. However, we can evaluate the un-normalised probability $p^*(\mathbf{x}) = \exp(\sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}))$. If there is a way of estimating Z, we can compute $p(\mathbf{x}) = p^*(\mathbf{x})/Z$.

A close look at the definition of Z shows Z can be written as an expectation with respect to a uniform distribution $(1/M)$:

$$Z = \sum_{\mathbf{x}\in\mathcal{X}} p^*(\mathbf{x}) = M \sum_{\mathbf{x}\in\mathcal{X}} \frac{1}{M} p^*(\mathbf{x}) = M E_{\frac{1}{M}}[p^*(\mathbf{x})] \tag{6.96}$$

[4] In statistical language Modelling task, $\mathbf{x}$ can denote a sentence in natural language. Hence $\mathcal{X}$ is an infinite set ranging over all possible sentences in a given language. In statistical parsing of natural language, $\mathbf{x}$ can be a valid parse licensed by some grammar. And $\mathcal{X}$ is a large set ranging over all possible parses that ban be generated by the grammar. Depend on the grammar used, $\mathcal{X}$ can be finite or infinite.

which implies we can estimate Z with importance sampling from the same samples we used to estimate $E_p[f_i]$:

$$\hat{Z} = M\frac{1}{R}\sum_{r=1}^{R}\frac{p^*(\mathbf{x})}{Mq(\mathbf{x})} = \frac{1}{R}\sum_{r=1}^{R}\frac{p^*(\mathbf{x})}{q(\mathbf{x})} \tag{6.97}$$

We now introduce *importance weight* w_r:

$$w_r = \frac{p^*(\mathbf{x})}{q(\mathbf{x})} \tag{6.98}$$

then,

$$\hat{Z} = \frac{1}{R}\sum_{r=1}^{R} w_r \tag{6.99}$$

and,

$$\begin{aligned}
\hat{E}_p[f_i] &= \frac{1}{R}\sum_{r=1}^{R}\frac{p(\mathbf{x})}{q(\mathbf{x})}f_i(\mathbf{x}) &(6.100)\\
&= \frac{1}{R}\sum_{r=1}^{R}\frac{p^*(\mathbf{x})}{\hat{Z}q(\mathbf{x})}f_i(\mathbf{x}) &(6.101)\\
&= \frac{1}{R}\frac{\sum_{r=1}^{R} w_r f_i(\mathbf{x})}{\hat{Z}} &(6.102)\\
&= \frac{\sum_{r=1}^{R} w_r f_i(\mathbf{x})}{\sum_{r=1}^{R} w_r} &(6.103)
\end{aligned}$$

The importance weight $w_r = p^*(\mathbf{x})/q(\mathbf{x})$ expresses the *importance* of a sample x_r generated from q. This is due to the fact that we are sampling from q rather than p. Obviously, if $q = p$, we have $w_r = 1$.

6.5.2 Adapting Sample Size

During training, it is unknown how many samples need to be generated in each iteration. Intuitively, in the first several iterations a small sample size is appropriate. As training goes on, the divergence between p and q will become larger and larger, hence more and more samples are needed to maintain a low variance of the estimators. Effective Sample Size (ESS) is a diagnostic measure for the importance sampling estimator [Logvineko,2001]

$$ESS = \frac{(\sum_{i=1}^{R} r_i)^2}{\sum_{i=1}^{R} r_i^2} \tag{6.104}$$

where r_i is the (normalised) *importance weight*:

$$r_i = \frac{w_i}{\sum_{i=1}^{R} w_i} \tag{6.105}$$

During training we can choose R so as to make sure the effective sample size is always greater than a minimum value R_0.This can be done by repeatedly adding a block of $R_b \geq 1$ samples until $ESS \geq R_0$. Below is the pseudo code for one iteration of importance-sampling-based training with adaptive sample size:

1. $ESS = 0$, $R = 0$
2. while $ESS < R_0$
   draw Rb samples from $q(\mathbf{x})$
   $R = R + Rb$
   update effective sample size:
   $ESS = \frac{(\sum_{i=1}^{R} r_i)^2}{\sum_{i=1}^{R} r_i^2}$
3. Estimate $\hat{E}_p[f_i]$ and $\hat{Z}$ with the R samples generated using eq (6.103) and eq (6.99).

6.6 Historical Remarks

The 1940s has witnessed significant progress in the theory of communication, marked by the influential 1948 paper of C.E. Shannon [Sha48]. In his pioneer work Shannon summarised almost all of the fundamental concept of modern information theory that is widely used today. [Shannon's work has sparked numerous applications]

One decade later, a mathematician named E.T. Jaynes proposed the maximum entropy principle in a paper published in *physical review* [Jay57]. Jaynes' 1957 paper was regarded by many as the first paper on maximum entropy principle. [TO BE CONFIRMED ON THIS]. Interestingly, Jaynes' paper was published over the objection of a reviewer[5]. [Historical remark: Although people have been practicing this principle/philosophy for many many years, it is the work of Ed. T. Jaynes that brought the concept of maximum entropy into a discipline of science.]

[KullBack's book]

The student of Kullback, Darroch, J.N. and Ratcliff, D. [TBC] invented a generalized iterative scaling (GIS) algorithm for estimating paramaters of ME model from data, which is a generalized of [XXX].

However, the principle of maximum entropy was not widely used in AI and NLP until early 1990s, when computer was fast enough to meet the computational requirement of such model. Researchers at IBM first applied maximum entropy model to machine translation task. Their work include the Conditional ME model [BDPDP96], Improved Iterative Scaling (IIS) algorithm, and a greedy random field induction algorithm [PPL97]. The technique was soon spread out by people visiting IBM. Rosenfeld applied conditional ME model to statistical language Modelling task [Ros96], and Ratnaparkhi attacked several NLP dis-ambiguity tasks with ME model [Rat98]. Both work represent the state-of-the-art of that time. ME model is so successful that by the end of 1990s, it has become a standard technique used in NLP. Several books have developed separate chapter discussing maximum entropy model. [Jel97][MS99], and [Jordan's book].

Various improvements over the standard ME model have been made in recent years. [CR99] investigated regularised ME model. They proposed to use a Gaussian prior to overcome the overfitting problem. This approach is effective and easy to implement and has become a standard component of common ME implementation. Prior other than Gaussian has also been proposed [Goo04]. [Mal02] performed a comparative study of different parameter estimating methods for ME model. His experiment showed that general-purpose gradient-based optimisation methods can be much faster than the iterative methods specially tailored for this problem. A second-order, limited-memory quasi-Newton method (also called L-BFGS) was observed significantly faster than all other methods evaluated, with IIS being the slowest one.

Dimensional reduction and model adaptation are two important practical issues when applying ME model to real-world data. [ZWWS03] proposed a faster dimensional reduction algorithm that considers a pool of candidate features in each round explicitly rather than selecting only one single best feature as was done in [BDPDP96]. MAP model adaptation has also been investivaged[CA04].

Currently on-going work focus on extending ME principle to sequential inference model such as Conditional Random Fields [LMP01] and latent variables. Scaling such models to large dataset is still an open issue.

6.7 Appendix: Jensen's Inequality

Jensen's inequality [Jen06] states that: let f be a function concave up on (a, b) i.e. for any $x_1, x_2 \in (a, b)$

$$f(\frac{x_1 + x_2}{2}) \leq \frac{f(x_1) + f(x_2)}{2} \tag{6.106}$$

then for any n numbers $x_i \in (a, b)$:

$$f(\frac{\sum_{i=1}^{n} x_i}{n}) \leq \frac{\sum_{i=1}^{n} f(x_i)}{n} \tag{6.107}$$

For concave down (convex) function, change all $\leq$to $\geq$ in (6.106) and (6.107).

The proof of this inequality can be found elsewhere[6]. We now derive several important inequalities from (6.107):

1. $f(x) = \ln(x), x > 0$. Since $\ln(x)$ is concave down (convex) we have:

$$\ln(\frac{1}{n}\sum_{i=1}^{n} x_i) \geq \frac{1}{n}\sum_{i=1}^{n} \ln x_i \tag{6.108}$$

[5] Jaynes kept that review, framed and hanging on the wall of his office for more than 40 years. See `http://bayes.wustl.edu/etj/report.html`

[6] http://www.math.udel.edu/~lazebnik/papers/jensen.ps.

2. $f(x) = e^x$. Since e^x is concave up we have:

$$\exp(\frac{1}{n}\sum_{i=1}^{n} x_i) \leq \frac{1}{n}\sum_{i=1}^{n} \exp(x_i) \tag{6.109}$$

3. If X is a real-valued random variable with $E(X)$ finite and the function $g(X)$ is convex, then $E(g(X) \geq g(E(X))$. For example: $E(X^2) \geq E(X)^2$.