ITERATIVE ALGORITHMS

Ramasubramanian H. Sundaram

Department for Electrical and Computer Engineering
Mississippi State University, Mississippi State, MS 39762
sundaram@isip.msstate.edu

ABSTRACT

Optimization techniques have helped in achieving optimal solutions to many a problems in various fields.Optimization methods using differential calculus can be applied to solve certain problems but as the problem becomes too cumbersome then classical methods get replaced by Iterative Techniques. This process of finding solution iteratively involves extensive computations and there are several algorithms on iteration. This paper will focus on various Iterative algorithms and their applications.

1. INTRODUCTION

Iteration is a systematic method for generating successive approximations to estimate one or more unknowns, starting with assumed values for the unknowns. The process of finding the successive approximations to estimate an unknown is called an *Iterative Process.* Each step in an iterative process is called an *Iteration* and each successive approximation is called an *Iterate.* Iteration is done by choosing x(0) as initial solution to the problem. Then one finds a better solution, say x(1),from x(0) and so on. The solution x(k) is the best possible solution if it minimizes the objective function. This objective function can be a cost function or error function around which the whole problem is based. If the successive approximations seem to converge to a value then the iteration is said to be *convergent*. Iteration can also be terminated if there are indications that no solution to the problem exists i.e. if the iteration process is divergent.

Though it is difficult to state a general rule as to when iteration is preferable to other methods, one major factor that limits the choice of method is the accuracy required and the computational cost. If accuracy is not a constraint then iterative methods can be used as they give an approximate solution sooner than other methods in less computational time.

This paper will explore in detail various iterative algorithms and their application. The mathematics involved in these algorithms is illustrated with suitable examples.

2. NEWTON-RAPHSON METHOD

Newton-Raphson method is an iterative method for finding the roots of a non-linear and transcendental equation [1]. It is by no means uncommon for these types of equation to occur in practice. There are no simple methods available for solving these equations and hence, iterative methods are frequently employed for solving these equations.

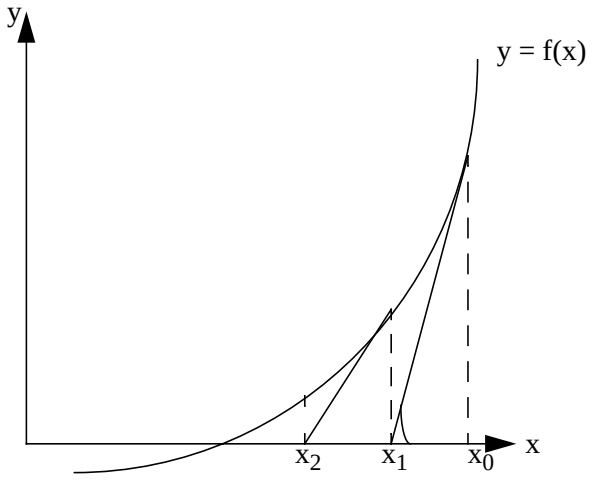

Figure 1: Illustration of Newton-Raphson Method.

2.1. Basics

Newton-Raphson is mainly used because of its simplicity and speed of convergence. Let us consider a function y=f(x) as shown in Figure 1. Let the function have a continuous derivative $f'(x)$.

The roots of the non-linear equation are the values of the variable x such that y = f(x) = 0.

To find the roots of the equation one chooses the initial estimate of x as x_0 and find $f(x_0)$. The successive approximate of x_0 is chosen as

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{1}$$

Proof: If β be the angle made by the tangent with the x-axis then

$$\tan\beta = \frac{f(x_0)}{x_0 - x_1} \tag{2}$$

From this we can find the successive approximate x_1 as in (1). The above formula can be generalized to find the $(n+1)^{th}$ approximation given the n^{th} estimate.

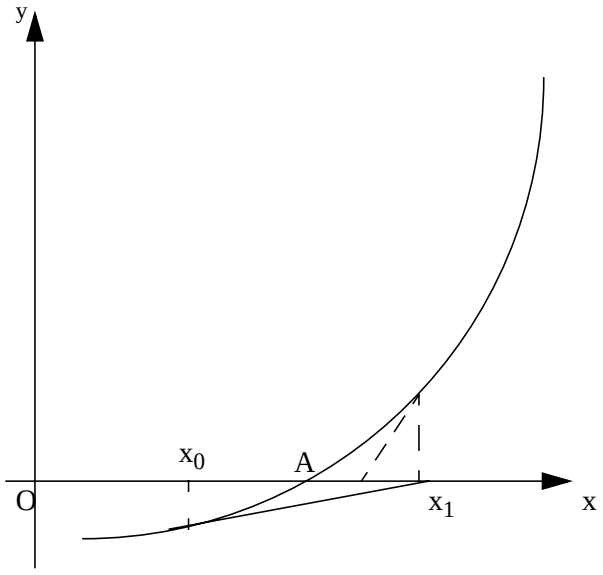

Figure 2: Newton-Raphson method for a different choice of initial estimate

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3}$$

In Figure 1, the initial estimate x_0 was chosen such that the successive estimates x_1, x_2, etc. were tending to the actual solution. Let us consider a case where the initial estimate $x_0$is chosen as in Figure 2.

As can be seen from the Figure 2, x_1 is in the opposite side of A from x_0. It may be even further from A than x_0 itself. After x_1 is obtained the iteration starts to converge to the exact solution. Thus one can infer that it is advisable to do the iterative process for two to three iterates before deciding whether it is convergent or divergent, as a process which seems to be divergent after the first iterate will be convergent when proceeded further.

2.2. Order of convergence

Let us characterize the quality of the Newton-Raphson method by judging the speed of convergence[1]. Let $x_{n+1} = g(x_n)$ define the iteration method where

$$g(x) = x - \frac{f(x)}{f'(x)} \tag{4}$$

Also let *s* be the actual solution. Then $x_n = s - e_n$, where e_n is the error of x_n. If g is differentiable a number of times then by Taylor's series[1] one can write

$$x_{n+1} = g(x_n) = g(s) + g'(s)(x_n - s) + \frac{1}{2}g''(s)(x_n - s)^2 + \ldots \tag{5}$$

$$x_{n+1} = g(s) - g'(s)e_n + \frac{1}{2}g''(s)e_n^2 \tag{6}$$

The exponent of e in the first non-vanishing term after g(s) is called the order of convergence of the Newton-Raphson method defined by g. The order measures the speed of convergence. Also, we find

$$g'(x) = \frac{f(x)f''(x)}{(f'(x))^2} \tag{7}$$

Equation (7) implies that $g'(s) = 0$. Subtracting g(s)

from both sides of (6) implies

$$-e_{n+1} = \frac{1}{2}g''(s)e_n^2 + \dots \quad (8)$$

$$e_{n+1} = -\frac{1}{2}g''(s)e_n^2 + \dots \quad (9)$$

It can be seen that if the iteration is convergent i.e. if e_n is less than one, then the Newton-Raphson method is of second order. The error at one iteration is a multiple of the square of the error at the previous iteration. As we near the actual solution the error term is small and hence convergence is quite rapid.

2.3. Example

Let us consider an example to illustrate the Newton-Raphson method. Consider the following equation

$$f(x) = e^x + x - 10 \quad (10)$$

To find a suitable we will find the value of the function given by (10) at various points and choose the x-value where the function changes sign as our initial estimate.

As can be seen from Table 1, since the function changes sign between x=2 and x=3, the roots lie between 2 and 3. Let the initial estimate of the root be x_0 =2. Differentiating f(x) with respect to x we get,

$$f'(x) = e^x + 1 \quad (11)$$

and x_{n+1} is given by

$$x_{n+1} = x_n - \frac{e^{x_n} + x_n - 10}{e^{x_n} + 1} \quad (12)$$

Substituting the values of the successive approximates, starting with $x_0 = 2$ in (12), we get the successive iterates as shown in Table 2.

For an accuracy of up to 3 decimal places we find the solution to be x = 2.071 which we get at the end of 4^{th} iteration.

2.4. Issues

Although Newton-Raphson method gives a faster convergence, care should be taken while considering the initial estimate for the root. A good initial estimate of the root can reduce the number of iterations drastically. A bad estimate may even lead to a divergent iterative process.

If an estimate of the root be a minimum point of the function, then the successive estimate will be at infinity[2]. In this case the whole process should be started with a different initial estimate.

3. STEEPEST DESCENT METHOD

Iterative methods are applied to optimization problems to find the maximum or minimum, given the objective function. The basic idea behind the iterative methods of optimization is to produce a sequence of improved approximations to the optimum. One such method to optimize an objective function is the Steepest Descent Method. Given the objective function M(x) a value of the variable vector $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ is sought that minimizes the objective function M(x). The constraint imposed on

x	f(x)
0	-9
1	-6.28
2	-0.61
3	13.09

Table 1: Choice of Initial estimate

x_n	Value
x_0	2
x_1	2.073
x_2	2.071
x_3	2.071

Table 2: Iterative Estimates to find the root

the objective function M(x) is that the function be a convex function [3].

3.1. Basics

The descent algorithms involve iterations which consist of essentially of three parts. First, a direction of descent is found, then a descent steplength is determined and finally the descent step is calculated. The iteration can be explained as:

1. Start with initial trial point X_i

2. Find a suitable direction S_i which points in the general direction of the minimum.

3. Find an appropriate step length λ_i for the movement along the direction of S_i

4. Obtain the new approximation $X_{i+1} = X_i + \lambda_i S_i$

5. Test whether $X_{i+1}$is optimum.

3.2. Theory

The descent direction is an n-dimensional vector $\mathbf{s} = (s_1, s_2,...,s_n)^T$. At the k^{th} iteration the direction vector s^k originates at the current point x^k. It points in a descent or "downhill" direction. That is, the value of the objective function decreases from x^k to a point at some distance in that direction.A vector s^k is said to be in *descent direction* with respect to the objective function M(x) at x^k, if it satisfies

$$M(x^{k+1}) = M(x^k + \lambda s^k) < M(x^k) \tag{13}$$

The descent steplength λ_i is a scalar. It is a measure of the distance along the descent direction s^k between two successive iteration points x^k and x^{k+1}. In other words, at the k^{th} iteration a step of length λ^k is taken from point x^k in the direction s^k to the point x^{k+1}.

Steepest descent method is based on the premise that the overall efficiency of the descent process is best if each of the iteration is optimal. Optimal descent iterations are those resulting in maximum reductions in the value of the objective function M(x).

3.3. Gradient of an objective function

The partial derivatives of an objective function, M, with respect to each of the *n* variables are collectively called the gradient of the function and is denoted by ∇M

$$\nabla M_{n\times 1} = \begin{bmatrix} \frac{\partial M}{\partial x_1} \\ \frac{dM}{dx_2} \\ \cdots \\ \frac{dM}{dx_n} \end{bmatrix} \tag{14}$$

The gradient is a n-component vector and it has a very important property. If we move along the gradient direction from any point in n-dimensional space, the functional value increases at the fastest rate. Hence the gradient direction is called the direction of steepest ascent. Since the gradient vector represents the direction of steepest ascent, the negative of the gradient vector denotes the direction of the steepest descent [3].

3.4. Rate of change of a function along a direction

Next we need to find the optimal steplength along the gradient direction that will minimize the objective function M(x)[4]. The next iterate will be at a distance λ_i along the direction S_i from the current iterate X_i. To find the rate of change of the objective function along the direction S_i(characterized by the parameter λ), that is,

$$\frac{dM}{d\lambda} = \sum_{j=1}^{n} \frac{\partial x_j}{\partial \lambda}\left(\frac{\partial M}{\partial x_j}\right) \tag{15}$$

where x_j is the j^{th} component of X. But

$$\frac{\partial x_j}{\partial \lambda} = \frac{\partial}{\partial \lambda}(x_{ij} + \lambda s_{ij}) = s_{ij} \tag{16}$$

where x_{ij} and s_{ij} are the j^{th} components X_i and S_i respectively. Hence,

$$\frac{dM}{d\lambda} = \sum_{j=1}^{n} \frac{\partial M}{\partial x_j} s_{ij} = \nabla M^T S_i \quad (17)$$

If λ minimizes M(x) in the direction $\mathbf{S_i}$, we have

$$\left.\frac{dM}{d\lambda}\right|_{\lambda = \dot{\lambda}} = \left(\nabla M\big|_{\dot{\lambda}}\right) S_i = 0 \quad (18)$$

at the point $\mathbf{X_i} + \lambda \mathbf{S_i}$

3.5. Example

To illustrate the steepest gradient algorithm let us consider an objective function involving two variables.

$$M(x, y) = x - y + 2x^2 + 2xy + xy^2 \quad (19)$$

with initial estimate

$$X_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (20)$$

Iteration 1:

The gradient of M is given by

$$\nabla M = \begin{bmatrix} \frac{\partial M}{\partial x} \\ \frac{\partial M}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 + 4x + 2y \\ -1 + 2x + 2y \end{bmatrix} \quad (21)$$

and

$$\nabla f_1 = \nabla f(X_1) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (22)$$

therefore,

$$S_1 = -\nabla M_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad (23)$$

To find the next estimate X_2, we need to find the optimal steplength λ_1 . For this, we minimize with respect to λ_1

$$M(X_1 + \lambda_1 S_1) = M(-\lambda_1, \lambda_1) = \lambda_1^2 - 2\lambda_1 \quad (24)$$

Equating (23) to zero we get

$$\lambda_1 = 1 \quad (25)$$

to be the optimal steplength. Hence, the successive estimate is given by

$$X_2 = X_1 + \lambda_1 S_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad (26)$$

As

$$\nabla M_2 = \nabla M(X_2) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (27)$$

X_2 is not the optimum.

Iteration 2:

$$S_2 = \nabla M_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (28)$$

To find the next iterate we need to find the optimal steplength λ_2

$$M(X_2 + \lambda_2 S_2) = M(-1 + \lambda_2, 1 + \lambda_2) \quad (29)$$

$$M(X_2 + \lambda_2 S_2) = 5\lambda_2^2 - 2\lambda_2 - 1 \quad (30)$$

Equating (29) to zero we get

$$\lambda_2 = \frac{1}{5} \quad (31)$$

Then the next estimate is given by

$$X_3 = X_2 + \lambda_2 S_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \frac{1}{5} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} \quad (32)$$

Since the components of the gradient at X_3,

$$\nabla M_3 = \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix} \tag{33}$$

are not zero, we proceed to the next iteration.

Iteration 3:

$$S_3 = -\nabla M_3 = \begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix} \tag{34}$$

Optimal steplength is found to be

$$\lambda_3 = 1.0 \tag{35}$$

The next iterate is

$$X_4 = \begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} + 1 \begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix} = \begin{bmatrix} -1.0 \\ 1.4 \end{bmatrix} \tag{36}$$

Since

$$\nabla M_4 \neq 0 \tag{37}$$

X_4 is not the optimum and one needs to proceed to the next iteration. This process is continued till the optimum point is found.

The iterative process can be terminated if two successive iterates are very close together[3] i.e.

$$\left| X_{i+1} - X_i \right| \leq \varepsilon \tag{38}$$

where ε is a very small number.

3.6. Issues

Though the steepest descent method converges faster, its convergence is slow when compared with conjugate gradient techniques. Also, for the this method to converge at the global minimum the objective function needs to be a convex function [4]. If the objective function is not convex, the steepest descent method will converge at the global minimum depending on the initial estimate. However, there are acceleration techniques being used to speed up the convergence of the steepest descent method.

4. THE EM ALGORITHM

Iterative algorithms also finds application in stochastic models. One such algorithm is the Expectation-Maximization (EM) algorithm which estimates the parameters of the a probability function such that it produces maximum-likelihood (ML) estimates of the parameters. EM algorithm is used when direct access necessary to estimate the parameters is impossible, or some of the data is missing.

4.1. Basics

The EM algorithm consists of two major steps: an expectation step, followed by a maximization step. The expectation is with respect to the unknown underlying variables, using the current estimate of the parameters and conditioned upon observations. The maximization step then provides a new estimate of the parameters. These two steps are iterated until convergence as in Figure 3.

4.2. General Statement of the EM algorithm

Let Y denote the sample space of the observations, and let $y \in R^m$ denote an observation from Y. Let χ denote the underlying space and let $x \in R^n$ be an outcome from χ, with m < n. The data x is referred to as the complete data. The complete data x is not observed directly, but only by means of y, where y(x), and y(x) is a many-to-one mapping.

The probability density function (pdf) of the complete data is $f_X(x|_\theta) = f(x|_\theta)$, where θ is the set of parameters of the density. The pdf f is assumed to be a continuos function of θ and appropriately differentiable. The pdf of the incomplete data is

$$g(y|_\theta) = \int_{\chi(y)} f(x|_\theta) dx \tag{39}$$

Let $l_y(\theta) = g(y|_\theta)$ denote the likelihood function and let

$$L_y(\theta) = \log g(y|_\theta) \tag{40}$$

denote the log-likelihood function.

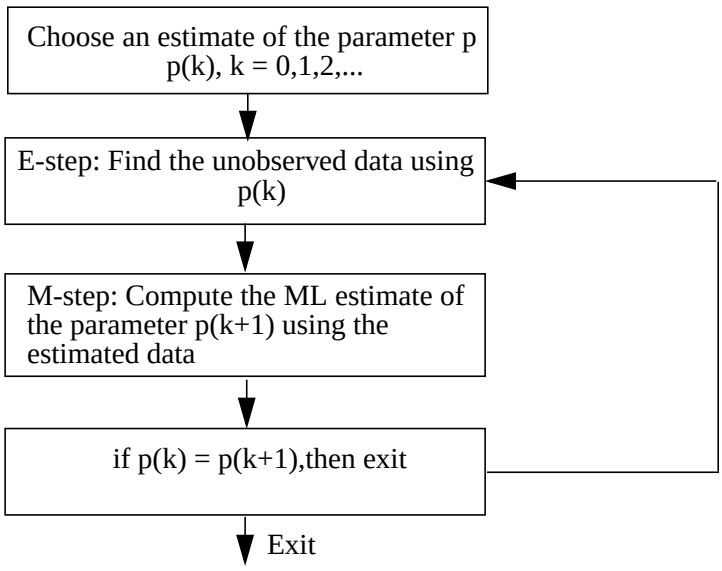

Figure 3: Flow diagram of an Expectation-Maximization algorithm.

The basic idea behind the EM algorithm is that we would like to find θ to maximize $\log f(x|_{\theta})$, but we do not have the data x to compute the log-likelihood. So instead, we maximize the expectation of $\log f(x|_{\theta})$ given the data y and our current estimate of θ. This can be expressed in two steps. Let θ^k be our estimate of the parameters at the k^{th} iteration.

For the Expectation step compute:

$$Q\left(\theta\Big|_{\theta^k}\right) = E\left[\log f(x|_{\theta})\Big|_{y,\,\theta^k}\right] \tag{41}$$

For the Maximization step let θ^{k+1} be that value of θ which maximizes $Q(\theta|_{\theta^k})$:

$$\theta^{k+1} = \arg max_{\theta} Q\left(\theta\Big|_{\theta^k}\right) \tag{42}$$

4.3. The EM Theorem

The theorem is based on a special case of Jensen's inequality[5]. If p(x) and q(x) are two discrete probability distributions, then

$$\sum_x p(x)\log p(x) \geq \sum_x p(x)\log q(x) \tag{43}$$

with the equality if and only if p(x) = q(x) for all x.

Let y denote observable data. Let $P_{\theta'}(y)$ be the probability distribution of y under some model whose parameters are denoted by θ'. Let $P_{\theta}(y)$ be the corresponding distribution under a different setting θ of the same parameters. By EM theorem, we prove that y is more likely to occur under θ than under θ'.

Let t be another variable whose value is determined in the same process that generates y. Then because $P_{\theta'}(t|_y)$ is a probability distribution that sums to 1,

$$\log P_{\theta}(y) - \log P_{\theta'}(y) =$$

$$\sum_t P_{\theta'}(t|_y)\log P_{\theta}(y) - \sum_t P_{\theta'}(t|_y)\log P_{\theta'}(y) \tag{44}$$

Since we can multiply by 1 without altering the equation, (44) becomes

$$= \sum_t P_{\theta'}(t|_y)\log P_\theta(y)\frac{P_\theta(t,y)}{P_\theta(t,y)} - \sum_t P_{\theta'}(t|_y)\log P_{\theta'}(y)\frac{P_\theta(t,y)}{P_\theta(t,y)} \quad (45)$$

$$= \sum_t P_{\theta'}(t|_y)\log\frac{P_\theta(t,y)}{P_\theta(t|_y)} - \sum_t P_{\theta'}(t|_y)\log\frac{P_{\theta'}(t,y)}{P_{\theta'}(t|_y)} \quad (46)$$

$$= \sum_t P_{\theta'}(t|_y)\log P_\theta(t,y) - \sum_t P_{\theta'}(t|_y)\log P_{\theta'}(t,y) + \sum_t P_{\theta'}(t|_y)\log P_{\theta'}(t|_y) - \sum_t P_{\theta'}(t|_y)\log P_\theta(t|_y) \quad (47)$$

Hence,

$$\log P_\theta(y) - \log P_{\theta'}(y) \geq \sum_t P_{\theta'}(t|_y)\log P_\theta(t,y) - \sum_t P_{\theta'}(t|_y)\log P_{\theta'}(t,y) \quad (48)$$

We have thus proven that if the right hand side of (48) is positive then,

$$P_\theta(y) > P_{\theta'}(y) \quad (49)$$

which is the basic EM theorem. It states that if we start with the parameter setting θ' and find a parameter setting θ such that right hand side of (48) is greater than zero, then the observed data y will be more probable under the regime of θ than they were under θ'. The reason for the name Expectation-Maximization is that we take the expectation of the random variable $\log P_\theta(t,y)$ with respect to the old distribution $P_{\theta'}(t|_y)$ and then maximize that expectation as a function of the argument θ

4.4. Convergence of the EM Algorithm

For the EM algorithm, at every iteration of the algorithm, a value of the parameter is computed so that the likelihood function does not decrease. That is, at every iteration the estimated parameter provides an increase in the likelihood function until a local maximum is achieved, at which the likelihood function cannot increase.

The rate of convergence of the EM algorithm is slower than the Newton's method. However, convergence near the maximum is rapid. Even with potentially slow convergence there are advantages of EM over other Newton's algorithms. There is no chance of "overshooting" the target or diverging away from the maximum. The EM algorithm is guaranteed to be stable and to converge to an ML estimate.

4.5. Example

Let us consider an practical problem [6] to illustrate the application of the EM algorithm. Suppose that in an image pattern-recognition problem, there are two classes to be distinguished: a class of dark objects and a class of light objects. The class of dark objects may be further subdivided into two shapes: round and square.

Let us assume the objects are known to be trinomially distributed. Let X_1 be the random variable that represents the number of round dark objects, X_2 represent the number of square dark objects, X_3 represent the number of light objects and let $[x_1, x_2, x_3]^T = x$ be the vector of values the random variables take for a particular image. Therefore

$$P\left(X_1 = x_1, X_2 = x_2, X_3 = x_3\Big|_p\right) = \left(\frac{n!}{x_1!x_2!x_3!}\right)\left(\frac{1}{4}\right)^{x_1}\left(\frac{1}{4}+\frac{p}{4}\right)^{x_2}\left(\frac{1}{2}-\frac{p}{4}\right)^{x_3} = f\left(x_1, x_2, x_3\Big|_p\right) \quad (50)$$

where p is an unknown parameter of the distribution and $n = x_1 + x_2 + x_3$

Let $[y1, y2]^T = y$ be the number of dark objects and number of light objects detected, respectively, so that

$$y_1 = x_1 + x_2 \tag{51}$$

and

$$y_2 = x_3 \tag{52}$$

As already discussed, there is a many-to-one mapping between $\{x_1, x_2\}$ and y_1. If $y_1 = 3$, then there are more than one possibility for x_1 and x_2 like $x_1=1$ and $x_2 = 2$ or $x_1 = 2$ and $x_2 = 1$. The EM algorithm is specifically designed for problems with such many-to-one mappings. Then

$$P\left(Y_1 = y_1 \middle|_p\right) = \left(\frac{n!}{y_1!}\right)\left(\frac{1}{2} + \frac{p}{4}\right)^{y_1}\left(\frac{1}{2} - \frac{p}{4}\right)^{n-y_1} \tag{53}$$

$$= g\left(y_1 \middle|_p\right)$$

The idea behind the EM algorithm is that, even though we do not know x_1 and x_2 directly, knowledge of the underlying distribution $f(x_1,x_2,x_3/p)$ can be used to determine an estimate for p. This is done by first estimating the underlying data, then using these data to update the estimate of the parameter. Let p^k indicate the estimate after k^{th} iteration, k = 1,2,.... An initial parameter value p^0 is assumed.

First the expectation step is performed to compute the expected value of the x data using the current estimate of the parameter. The expected value of x_1, given the measurement y_1 and current estimate of the parameter [6]

$$x_1^{[k+1]} = E\left[x_1 / \left(y_1, p^{[k]}\right)\right] \tag{54}$$

$$x_1^{[k+1]} = y_1 \frac{\left(\frac{1}{4}\right)}{\frac{1}{2} + \frac{p^k}{2}} \tag{55}$$

Similarly,

$$x_2^{[k+1]} = E\left[x_2 / \left(y_1, p^{[k]}\right)\right] \tag{56}$$

$$x_2^{[k+1]} = y_1 \frac{\frac{1}{4} + \frac{p^k}{2}}{\frac{1}{2} + \frac{p^k}{2}} \tag{57}$$

x_3 need not be computed as it is uniquely known from y_2.

Next, perform the maximization step using the data from the expectation step to find the new estimate of the parameter. This is done by taking the derivative of $\log f(x_1^{k+1},x_2^{k+1},x_3/p)$ with respect to p, equating it to zero, and solving for p,

$$\frac{d}{dp}\log f\left(x_1^{k+1}, x_2^{k+1}, x_3 / p\right) = 0 \tag{58}$$

Solving for p we get the next estimate $p^{[k+1]}$ as

$$p^{k+1} = \frac{2x_2^{k+1} - x_3}{x_2^{k+1} + x_3} \tag{59}$$

Expressing the next estimate of the parameter in terms of the current estimate of the parameter we get [6],

$$p^{k+1} = \frac{p^k(4y_1 - 2x_1) + 2y_1 - 2x_1}{p^k(2y_1 + 2x_3) + y_1 + 2x_3} \tag{60}$$

If the actual values of p, x_1, x_2 are 0.5, 25, 38 respectively and if we start with an initial estimate of p i.e. $p^0 = 0$, then the algorithm proceeds as shown in the Table 3.

4.6. Remarks

The EM algorithm can be employed when there is an underlying set with a known distribution function that is observed by means of a many-to-one mapping [6]. Analytically, the most difficult portion of the EM algorithm is the E-step. This also expensive in

k	$x_1^{[k]}$	$x_2^{[k]}$	$p^{[k]}$
1	31.500000	31.500000	0.379562
2	26.475460	36.524540	0.490300
3	25.298157	37.701843	0.514093
4	25.058740	37.941260	0.518840
5	25.011514	37.988486	0.519773
6	25.002255	37.997745	0.519956
7	25.000441	37.999559	0.519991
8	25.000086	37.999914	0.519998
9	25.000017	37.999983	0.520000
10	25.000003	37.999997	0.520000

Table 3: Calculations involved in estimating an unknown parameter using the EM algorithm

computation as the expectation must be computed over all values of the unobserved variables.In most instances, where the EM algorithm applies, there are other algorithms like gradient descent which can also be applied. Though the EM algorithm is slower in convergence, it is easy to compute when compared with other algorithms as it does not require higher order derivatives to be calculated.

5. SUMMARY

In this paper, we reviewed various iterative algorithms and their application to various problems. While a particular problem can be solved using different algorithms, a proper choice of algorithm is required so that computations are minimized. Care should be taken so that the iteration does not diverge or overshoot the required value. The choice of an algorithm for a particular application also depends upon the accuracy required and the rate of convergence.

6. REFERENCES

[1] Erwin Kreyszig, “Advanced Engineering Mathem atics,” 7th edition,John Wiley & Sons, Inc.,1993

[2] A. C. Bajpai, I. M. Calus, J. A. Fairley, “Numeric al Methods for Engineers and Scientists,” John Wiley & Sons, 1978

[3] S.S.Rao,“Optimization:TheoryandApplications,” 2nd Edition, John Wiley and Sons, 1978.

[4] S.L.S. Jacoby, J.S. Kowalik, J.T. Pizzo,“Iterative Methods for Nonlinear Optimization Problems,” Prentice-Hall Series in Automatic Computation, 1972.

[5] Frederick Jelinek,“Statistical Methods for Speech Recognition,” The MIT Press Cambridge, Massachusetts, 1997.

[6] T. K. Moon, “The Expectation-Maximization Algorithm,” *Signal Processing*, vol. 13, no. 6, pp.47 - 60, November 1996.