Adaptive Learning Rate Algorithms for Stochastic Optimization and Variational Bayesian Inference

Author: Ioannis Papantonis

Supervisor: Dr. Michalis Titsias
## Contents

1 Introduction  
1.1 Numerical analysis in general .................................. 4  
1.2 Convex optimization ............................................ 5  
1.3 Examples of convex problems .................................... 7  
1.4 Limitations ....................................................... 8  
1.5 Stochastic optimization .......................................... 9  
1.6 Robbins Monro conditions ....................................... 10  
1.7 Review of some popular stochastic optimization algorithms .. 11  
1.8 Connections to Bayesian inference ............................... 13  
1.9 Variational inference ............................................ 15  
1.10 Formulation of the problem ...................................... 17  
1.11 Applications of Variational inference ........................... 19  

2 Learning rate analysis  
2.1 Introduction ..................................................... 23  
2.2 On the selection of metric ....................................... 24  
2.3 One step variance reduction ..................................... 25  
2.4 Derivation of optimal $\eta$ ....................................... 26  
2.5 Remarks ........................................................... 27  
2.6 Multi step variance reduction ................................... 28  
2.7 Derivation of optimal $\eta$ ....................................... 29  
2.8 Remarks ........................................................... 30  
2.9 Online interpretation ............................................ 31  

3 Gaussian approximations  
3.1 Introduction ..................................................... 34  
3.2 The model ....................................................... 35  
3.3 Reparameterization .............................................. 36  
3.4 The problem ..................................................... 38  
3.5 Intuition and a simple example .................................. 40  
3.6 Taylor approximation ............................................ 41  
3.7 Remarks ........................................................... 43  

4 Experiments  
4.1 The proposed algorithm ......................................... 44  
4.2 Logistic Regression ............................................... 46
4.3 Neural Network ............................................. 48

References ....................................................... 50
1 Introduction

1.1 Numerical analysis in general

Numerical analysis is a field of mathematics that studies ways to obtain numerical approximations of the solutions of problems, and are usually employed when the exact solutions are either analytically intractable or computationally intensive. There are many problems, for example, in Differential equations or in finding the roots of a polynomial, where we have theoretical results stating that the problem has solutions, but not a way to acquire them, so approximating them is a way to overcome this problem.

Perhaps the earliest example of approximation can be found in a Babylonian tablet, where they attempted to measure the length of the diagonal of a unit square, which nowadays, we just call $\sqrt{2}$, using what was, later, named Heron’s method [1]. One more famous example is Archimedes’ approximation of pi, using the method of exhaustion. It is interesting to notice the similarities between Archimedes’ method and the definition of Riemann sums, used in integration of a function [2].

Contemporary numerical analysis focuses mainly on questions regarding ways to solve equations or systems of equations, interpolate a function, evaluate integrals, solve differential equations and optimize a function. In the following chapters, we are going to be concerned with the last area of study, although this statement is not entirely true, since there is some overlap between these categories and problems can be expressed in various forms, so tools from more than one of them can be used.

An example of this interchangeability can be found in Newton’s method [3]. This is a standard, iterative, algorithm used in order to find the roots of a function. It is a well known result that for, a differentiable, function, the derivative equals zero at the points where it attains an optimal value [4]. This allows us to transform the problem of minimizing or maximizing a function into a problem of finding the zeros of another function, so we can apply Newton’s method to the derivative to obtain an optimization algorithm.
1.2 Convex optimization

The problem of optimizing a function [5] is pretty much solved for linear functions, but the general case still poses a challenge. Thankfully, there is a class of functions with a lot of useful properties, where we can derive optimization algorithms with theoretical guarantees about their convergence, so let’s proceed with defining it.

Let \( f \) be a function, \( f : X \to R \). A function is called convex if:

\[
\forall x_1, x_2 \in X, \forall t \in [0, 1]: f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).
\]

The above condition implies that the line segment connecting \( x_1, x_2 \) is above the function, in the interval \([x_1, x_2]\), as seen in figure 1(a). There are equivalent ways to define a convex function, each one providing a different insight. As one could suspect from the name alone, there is a connection between convex functions and convex sets, specifically, a \( f \) is convex if and only if the epigraph of \( f \) is a convex set, where the epigraph \((f) = \{(x, t) : t \geq f(x)\}\) [6]. Using our example, this means that the parabola along its interior forms a convex set, which is clearly true. Another equivalent definition is that a function is convex if and only if its second derivative is non negative, or if the Hessian matrix is positive semidefinite [6]. The following is the most important definition for the rest of the chapter, because it makes use of the tangent of the function and provides us with a useful inequality: \( f \) is convex if and only if it lies on or above all its tangents [6], as shown in figure 1 (b).
The corresponding inequality is the following:

\[ \forall x, y \in X : f(x) \geq f(y) + f'(y)(x - y) \] (1)

A first remark we can make by inspecting this inequality, is that if for a \( a \in X \) we have that \( f'(a) = 0 \), then plugging it in the above expression we have that:

\[ \forall x \in X : f(x) \geq f(a) \]

so \( a \) is a global minimum.

Interestingly, if we choose to use an, iterative, descent method of the following scheme:

\[ x_{t+1} = x_t - \eta_t \Delta x_t \]

where \( \eta_t \) is called the step size or learning rate at iteration \( t \) and \( \Delta x_t \) is the search direction for the same iteration, then eq (1) implies that all we have to do is to enforce that:

\[ f'(x_t) \cdot \Delta x_t \geq 0 \]

or

\[ \nabla f(x_t) ^\top \cdot \Delta x_t \geq 0 \] (2)

for the multi dimensional case.

An obvious choice of \( \Delta x_t \), in order to satisfy the above condition, is to use the derivative at \( x_t \), \( f'(x_t) \) or \( \nabla f(x_t) \), for the multi dimensional case. The resulting update is:

\[ x_{t+1} = x_t - \eta_t f'(x_t) \]

and this algorithm is known as Gradient Descent and is one of the most popular ways to optimize a function, mainly for problems with a moderate amount of data, because it can be computationally intensive and time consuming when used in very high dimensionality problems. We will return to this discussion later, as this limitation has given rise to a whole new class of stochastic algorithms.
Another way to choose a descent direction can be found by inspecting the equivalent definitions for a convex function. One of them states that the second derivative is non-negative or that the Hessian matrix is positive semidefinite. Following this observation, we could satisfy eq (2) by setting \( \Delta x_t \) equal to \( f'(x_t) \cdot f''(x_t)^{-1} \) or \( H(x_t)^{-1} \cdot \nabla f(x_t) \), for the multi-variable case, where we have denoted by \( H(x_t) \) the Hessian matrix of \( f \) at \( x_t \). Thus, the update rule becomes:

\[
x_{t+1} = x_t - \eta_t f'(x_t) \cdot f''(x_t)^{-1}
\]

this algorithm is known as Newton’s method and it can be interpreted as approximating \( f \) with its second order Taylor expansion [3].

This is also a widely used algorithm for optimizing a function, because it is fast in general and has some nice properties, such as it converges to the solution in just one iteration, for quadratic functions, since, in that case, the second order Taylor approximation and the actual function are identical. On the negative side, it requires the computation of the inverse of the Hessian, which is \( O(N^3) \), so it is infeasible in problems with very high dimensionality. It is worth noting that, for both algorithms, we have only specified the search direction, so we still need to adjust the learning rate. This question will be addressed in the following chapter.

### 1.3 Examples of convex problems

We will now give two examples of convex problems that find application in the field of Machine Learning. We will start with the well-studied problem of regression [7]. Here, we want to find the solution of an overdetermined system of equations, meaning that there are more equations than unknowns, so instead of finding an exact solution, we project it into a lower dimensional space. Formally, using ordinary least squares, we want the solution of the following problem:

\[
\min_{x \in \mathbb{R}^{M \times 1}} \| Ax - b \|_2^2
\]

where \( A \in \mathbb{R}^{N \times M}, \ b \in \mathbb{R}^{N \times 1} \). The above function is quadratic in \( x \), so it is convex. It is worth noting that this problem can be solved analytically, by setting the derivative equal to zero, but in practice it is solved using one of the algorithms that we have mentioned, in order to avoid solving a linear system with a large number of equations.
Another popular category of classifiers in supervised learning consists of maximum margin classifiers, more specifically Support Vector Machines [8]. Here, the problem is to find the hyperplane with the maximum margin, in order to separate data with different labels. Formally, we want to solve the following:

$$\min \frac{1}{2} \|w\|_2^2$$

s.t.:

$$y_i \cdot (w^\top \cdot x_i + b) \geq 1, \; i = 1, \ldots, n$$

where $y_i$ is the label of data point $i$, $w$ is the parameters’ vector, $x_i$ is the $i$ data point and $b$ is a number that can be interpreted as a bias parameter. This is a constrained convex problem and, in practice, we don’t attain the solution by solving it, but by solving its dual.

1.4 Limitations

The presented algorithms work very well on convex problems, by incorporating information about the curvature of the objective function, using either the first derivative, as in Gradient descent, or the second derivative, such as Newton’s method. In fields like Machine Learning, usually the objective function corresponds to the cost function of the underlying model, where each training data point contributes to it, so it is the sum of partial cost functions.

In contemporary problems it is not unusual to have massive data sets. In the context of having a cost function made by partial costs, one would have to calculate the derivative on each data point, at every iteration. When speaking about data sets that cannot even be loaded into RAM, it means that the calculations will be dramatically slow.

In addition to the above, the dimensionality of the problem has an effect on the efficiency of such algorithms. Take, for example, Deep Learning models [9], that may have millions of parameters to be learnt. An algorithm like Newton’s method, that requires the inverse of the Hessian matrix, cannot even be used, for various reasons, since the computations would be extremely costly and unstable, while the matrix itself would be impossible to be stored.
1.5 Stochastic optimization

If we take a closer look at the presented limitations, we can see that they mostly rise due to our choice to perform exact calculations. It is reasonable to question what happens if we relax this requirement and if this relaxation will result in a more efficient algorithm. An obvious way to proceed, would be to somehow approximate the desired quantities.

We can now look at an example to demonstrate this idea, following the set up of a cost function that is the sum of individual, partial costs. We will also use Gradient descent to perform the optimization, so the quantity we have to estimate is the gradient of the function. To this end, an approach would be to select a subset of the whole data set, carry out the necessary computations on this new set and use the result as an estimate of the true gradient.

Formally, we want to minimize a function of the form:

\[ F(w) = \frac{1}{N} \sum_{i=1}^{N} f_i(w, x_i) \]

where \( N \) is the number of training examples, \( w \) is the parameter, \( x_i \) is the \( i \)-th data point and \( f_i \) is the corresponding cost for \( x_i \).

The gradient of \( F \) is equal to:

\[ \nabla_w F(w) = \frac{1}{N} \sum_{i=1}^{N} \nabla_w f_i(w, x_i) \]

We introduce stochasticity in the following manner: Let \( g \) be a random variable uniformly distributed on the set \( \{ \nabla_w f_1(w, x_1), \ldots, \nabla_w f_N(w, x_N) \} \). We notice that the expectation of \( g \) is:

\[ E(g) = \frac{1}{N} \sum_{i=1}^{N} \nabla_w f_i(w, x_i) = \nabla_w F(w) \]

Supposing we use \( M \) data points to approximate \( \nabla_w F(w) \), we consider random variables \( g_1, \ldots, g_M \) following the same distributions as \( g \). Each \( g_i \) is equal to \( \nabla_w f_j(w, x_j) \), for some \( j \), so there is a subset, \( I \), of the subscripts \( \{1, \ldots, N\} \), such that \( \{g_1, \ldots, g_M\} = \bigcup_{i \in I} \{ \nabla_w f_i(w, x_i) \} \).
We are going to use $\frac{1}{M} \sum_{j=1}^{M} g_j$ to estimate the true gradient. We can use the last remark we made, to rewrite this as:

$$\frac{1}{M} \sum_{j=1}^{M} g_j = \frac{1}{M} \sum_{i \in I} \nabla_{w} f_i(w, x_i)$$

so it just the gradient of the function calculated on a subset of the training data set.

Finally, we show that this estimator is unbiased, since

$$E(\frac{1}{M} \sum_{j=1}^{M} g_j) = \frac{1}{M} \sum_{j=1}^{M} E(g_j) = \frac{1}{M} M E(g) = \nabla_{w} F(w)$$

The above procedure proves that we can select a subset or mini batch of the data set in order to obtain an unbiased estimator of the true gradient of the objective function. Furthermore, the update rule, using this estimator, takes the form:

$$w_{t+1} = w_t - \eta_t \frac{1}{M} \sum_{i \in I} \nabla_{w} f_i(w, x_i)$$

This algorithm is called Stochastic Gradient Descent with mini batch of size $M$.

1.6 Robbins Monro conditions

In the previous section, we examined away to construct an unbiased estimator of the gradient of a function and how to use it to get a stochastic optimization algorithm. For deterministic algorithms, like Gradient Descent, we have theoretical results stating useful properties regarding their behaviour, so it is logical to ask if stochastic algorithms share the same properties. As it turns out, the learning rate plays a crucial role in the convergence of such algorithms. Perhaps this should be expected, since even for their deterministic counterparts, a bad choice of learning rate can cause the algorithm to either oscillate around the solution or never reach it. Given the fact that stochastic algorithms make use of noisy estimates, instead of exact values, a bad learning rate can be disastrous.
The following result is due to Robbins and Monro [10], and is extremely useful, since it provides us with sufficient conditions for the algorithm to converge.

Let $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ be the function we are interested in minimizing, using a scheme of the form $x_{t+1} = x_t - \eta_t (\nabla f(x_t) + D_{t+1})$, where $\nabla f(x_t)$ is the exact gradient and $D_{t+1}$ is a zero mean, random noise, so $\nabla f(x_t) + D_{t+1}$ is an unbiased estimator of $\nabla f(x_t)$. Then this scheme converges to the minimum, if the following are satisfied:

(i) $\lim_{t \to \infty} \eta_t = 0$,  
(ii) $\sum_{t=1}^{\infty} \eta_t = \infty$,  
(iii) $\sum_{t=1}^{\infty} \eta_t^2 < \infty$

In practice, even a small, constant learning rate results in a convergent algorithm.

1.7 Review of some popular stochastic optimization algorithms

We saw that by using an appropriate schedule for the learning rate, we can have a convergent stochastic algorithm with fewer computational and data storage requirements. Of course, on the down side, Stochastic Gradient Descent (SGD) performs more iterations to converge, as well as it exhibits random walk behaviour. The latter, has led to different variants of SGD that deal with this problem.

One of the first attempts was to add an additional term, to end up with a scheme of the following form:

$$
\begin{align*}
    u_{t+1} &= \gamma u_t + \gamma \nabla w g(w_t) \\
    w_{t+1} &= w_t - u_{t+1}
\end{align*}
$$

where $\gamma$ is the momentum term, usually set at around 0.9, and $g(w_t)$ is a stochastic estimate of the gradient of the objective function [11].

Intuition about the update can be gained from Physics [12]. In this view, we see $w_t$ as particle moving on the graph of the objective function, with $w_0$ as its initial position. The force acting on this particle is equal to the negative gradient of the potential energy. In addition, since $F = ma$, the acceleration is proportional the force on the particle. Combining this, with the above update rule, $u_t$ can be interpreted as a velocity term, so at each
iteration we first update the velocity of the particle, using the momentum term to ensure that the particle will stop when it reaches the bottom of the landscape, and use this value to update the position.

Next, we take a look at slightly different version of the momentum SGD, called Nesterov accelerated gradient [13]. This time, the update has the form:

\[
\begin{align*}
  u_{t+1} &= \gamma u_t + \gamma \nabla_w g(w_t - \gamma u_t) \\
  w_{t+1} &= w_t - u_{t+1}
\end{align*}
\]

Taking a closer look at this, the only difference with the momentum SGD, is that instead of computing the gradient at \(w_t\), we use \(w_t - \gamma u_t\). A way to, intuitively, justify this alteration, is that by looking at the updates of the momentum SGD, we know that \(w_t\) will move by \(\gamma u_t\), plus the gradient term, so \(w_t - \gamma u_t\) can be used as an approximation for \(w_{t+1}\). Supposing that this estimate is not far from the complete update, it makes sense to compute the gradient at it, since we incorporate information about the future position, and use this gradient as our search direction.

Next we are going to present the Adagrad algorithm [14]. One of the main differences between this and the previous algorithms is that Adagrad uses adaptive learning rates, different ones per feature. The update rule is as follows:

\[
  w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t} + \epsilon} \cdot g(w_t)
\]

where \(\eta\) is the learning rate, \(\epsilon\) is a small number to avoid division with zero, and \(G_t\) is a vector where the i-th element is the sum of the squares of the partial derivatives of the objective function w.r.t. \(w_i\), up to iteration \(t\), so \(G_t = \sum_{i=1}^{t} g_i(w_t)^2\), where \(g_i\) is the estimate of the gradient at iteration \(i\). All the operations are element wise, so, for example, the i-th element of \(\frac{g(w_t)}{\sqrt{G_t}}\), is equal to the product of the i-th element of \(g(w_t)\) times the square root of the inverse of the i-th element of \(G_t\).

One of the main advantages of Adagrad is that it is no longer needed to manually adjust the learning rate, so in most applications we just choose a constant value for \(\eta\) and leave it like that, throughout all the iterations. In addition, it is worth noticing that parameters with high gradient will have a small learning rate, while those with a small gradient will have a big learning rate, so it acts as a way to put more weight on infrequent features.
On the other hand, the fact that the denominator increases after each iteration, which in turn causes the learning rate to diminish, is one of the main disadvantages. So after a certain point the algorithm will no longer be able to learn, and if this happens early then it might not converge.

The last algorithm we will review is called Adam and is one of the most widely used in Machine Learning [15]. The update rule is:

\[
\begin{align*}
    m_t &= \beta_1 m_{t-1} + (1 - \beta_1)g(w_t) \\
    u_t &= \beta_2 u_{t-1} + (1 - \beta_2)g(w_t)^2 \\
    m_t &= \frac{m_t}{1 - \beta_1^t} \\
    u_t &= \frac{u_t}{1 - \beta_2^t} \\
    w_{t+1} &= w_t - \frac{\eta}{\sqrt{u_t} + \epsilon} \cdot m_t
\end{align*}
\]

where \( \beta_1, \beta_2 \in (0, 1) \), \( \epsilon \) is a small number to avoid division with zero, and all the vector operations are element wise. The quantities \( m_t, u_t \) can be seen as decaying average estimators of \( \text{E}(g(w_t)), \text{E}(g(w_t)^2) \), respectively. This algorithm shares similar theoretical bounds to Adagrad, but it also tries to overcome the problem of the monotonically decreasing learning rate, using a moving average of the sum of the squares of the gradients, instead of the whole sum, as Adagrad does.

### 1.8 Connections to Bayesian inference

Bayesian Statistics is a field of Mathematics, with applications to various areas, including Machine Learning. In recent years there has been an increase to the available computational power, and this has led to the rise of Bayesian methods, since previously intractable models can be brought to life, using methods such as Markov Chain Monte Carlo (MCMC) sampling [16]. It is interesting that contemporary research in stochastic optimization employs such ideas. There have been different approaches on how to connect these fields, like considering the sequence of the optimization updates, \( \{w_t\}_{t=1}^T \), as sampling from the posterior distribution of a random variable. Other researchers have employed approximate Bayesian methods as a tool in developing stochastic algorithms. The idea that there is some interchangeability between these two fields is not entirely new, since even the Gibbs
sampler was first introduced as a maximum a posteriori (MAP) inference method [17].

As we mentioned previously, MCMC methods have been used to produce stochastic optimization algorithms. To give an example, we will briefly discuss a certain application. In general, MCMC algorithms are used in order to sample from an intractable posterior distribution.

Stochastic Gradient Langevin Dynamics (SGLD) is such a sampling algorithm. The update for the parameter, \( w \), is:

\[
\begin{align*}
    w_{t+1} &= w_t - \frac{\epsilon}{2} \nabla f(w_t) + \eta_t \\
    \eta_t &\sim N(0, \epsilon)
\end{align*}
\]

where \( f(w_t) \) corresponds to the potential energy of the model, usually defined in terms of the posterior distribution or the cost function. The above scheme will end up drawing samples from the true posterior distribution of the parameters.

The idea is to combine this with the Robbins Monro conditions, so instead of using a constant value for \( \epsilon \), the authors use a sequence \( \epsilon_t \), satisfying the necessary conditions [18]. The result is an algorithm that initially behaves as an optimizer, after some point it enters a posterior sampling phase, and it transitions smoothly between the two.

We close the chapter of stochastic optimization with an algorithm that makes of an approximate Bayesian method. In a recent paper it was shown that, under some assumptions, the update rule of SGD can be seen as the discretization of a Stochastic Differential Equation, which in turn defines an Ornstein-Uhlenbeck process [19]. This means that the aforementioned equation has an analytic, stationary, Gaussian distribution that depends on the choice of the learning rate. By assumption, the posterior distribution is also Gaussian, at a region around the mode. The rest of the analysis is concerned with different ways to adjust the value of the learning rate, in such a way, that these two distributions get as close as possible. The derivation of this algorithm highlights an important concept of approximate Bayesian Statistics, the distance between distributions and how to minimize it in order to obtain a meaningful approximation. We are going to examine this concept in greater detail in the next section.
1.9 Variational inference

The main tool in Bayesian inference is the posterior distribution of the parameters, given the observed data. This distribution is essential in Bayesian analysis, from performing MAP inference to computing the expectation of the complete data log-likelihood in EM. There are several algorithms that allow for exact inference, such as the Junction Tree one, for discrete graphical models [16]. However, their time complexity renders them impractical for high dimensional models, like those used in real world applications.

In cases like these, some sort of approximation is needed. Generally speaking, there are two main approaches on developing such methods, the stochastic and the deterministic ones. Stochastic techniques, like MCMC, have made Bayesian methods widely applicable, since they have nice theoretical properties, the approximations they generate would be equal to the exact value, given infinite computational power, as well as they act as a black box tool for researchers in non Mathematical fields. On the downside, these methods require a substantial amount of computational power, so they are not very efficient for high dimensional problems.

On the other hand, deterministic techniques rely on analytical approximations of the posterior distribution. Before going on with the calculus of variations, we will give an example of one of the simplest deterministic methods, the Laplace approximation. To this end, consider a distribution function of the form:

\[ p(x) = \frac{1}{Z} e^{-E(x)} \]

where the normalization constant, \( Z \), is not known. The first step is to find the mode of \( p(x) \), so we need to calculate \( x^* = \arg \min_{x} E(x) \). This can be done using an optimization algorithm. Once we have \( x^* \), we use the second order Taylor expansion of \( E(x) \) around it:

\[ E(x) \approx E(x^*) + (x - x^*) \nabla E(x)|_{x=x^*} + \frac{1}{2} (x - x^*)^T H(x^*)(x - x^*) \]

where \( H(x^*) \) is the Hessian matrix at \( x^* \), \( H(x^*) = \nabla \nabla E(x)|_{x=x^*} \). Combining the above with the fact that \( \nabla E(x)|_{x=x^*} = 0 \), since \( x^* \) is the mode, result in the following approximation:

\[ p(x) \approx \frac{1}{Z^*} e^{-E(x^*)-\frac{1}{2}(x-x^*)^T H(x^*)(x-x^*)} = \frac{1}{Z^*} e^{-\frac{1}{2}(x-x^*)^T H(x^*)(x-x^*)} \]

15
The above is the form of a Gaussian distribution, more specifically, we deduce that \( x \sim N(x^*, H^{-1}(x^*)) \). Now we can use this distribution to calculate any desired statistical functions, such as expected values, instead of the true posterior. For example, an estimate for \( Z \) can be obtained from the fact that \( Z^* \) is the normalizing constant of a Normal distribution, so:

\[
Z^* = \sqrt{\det 2\pi H^{-1}(x^*)} \Rightarrow Ze^{E(x^*)} = \sqrt{\det 2\pi H^{-1}(x^*)}
\]

meaning that the approximated value is:

\[
Z = e^{-E(x^*)} \sqrt{\det 2\pi H^{-1}(x^*)}
\]

There are some things that have to be taken under consideration when using this approximation. First of all, the result depends on the mode, so for multi-modal distributions, different choices of the mode produce different Laplace approximations. In addition, this whole process is applicable only to continuous random variables. Another thing is that it focuses around the mode, so it may fail to capture global properties of the distribution. It is also worth noting that this approximation is basis dependent, meaning that if we use a non linear transformation of \( x \), then, in general, the resulting estimate for \( Z \) will be different [20]. This is not desired behaviour, since the true value of \( Z \) is not dependent on such transformations.

Variational methods follow a completely different approach, in order to resolve some of the above limitations. The main idea is that instead of trying to directly approximate the posterior using an analytical tool, such as Taylor expansions, we choose a set of candidate distributions and use the one closest to the posterior, to approximate it [16]. In this view, we now have to solve an optimization problem, a distance minimization one. It is worth noting that if the domain of the possible solutions has no restrictions, then the problem is trivial, and the solution is the posterior itself, since the distance becomes equal to zero. At this point it should be clear that restricting the space of feasible solutions, naturally results in a function that serves as an approximation to the posterior.

As it has already been mentioned, some sort of distance is needed, in order to measure how close two distributions are. A popular choice is to use the Kullback-Leibler divergence [21]. This function, used in various fields, measures how one probability distribution diverges from a second probability distribution. In the extreme case, when it is equal to zero, we can expect the two distributions to have the same behaviour.
1.10 Formulation of the problem

We start with some properties of the Kullback-Leibler divergence:

\[ KL(p||q) = \int_{-\infty}^{\infty} p(\theta) \log \frac{p(\theta)}{q(\theta)} \, d\theta \]

A first remark is that it is not symmetric in \( p \) and \( q \), so \( KL(p||q) \neq KL(q||p) \). In our case, we have assumed that the posterior distribution is intractable, so it is reasonable to assume that expectations w.r.t. it, are intractable, as well. On the contrary, most of the times, the variational distribution belongs to a well known family of parametric distributions, so calculating expectations w.r.t. it, should be tractable.

Furthermore, it is not difficult to prove that KL is a non negative function, with zero corresponding to the case \( p = q \), using Jensen’s inequality [6]:

\[
-KL(p||q) = -\int_{-\infty}^{\infty} p(\theta) \log \frac{p(\theta)}{q(\theta)} \, d\theta = \int_{-\infty}^{\infty} p(\theta) \log \frac{q(\theta)}{p(\theta)} \, d\theta \\
\leq \log \int_{-\infty}^{\infty} p(\theta) \frac{q(\theta)}{p(\theta)} \, d\theta = \log \int_{-\infty}^{\infty} q(\theta) \, d\theta \\
= \log 1 = 0 \\
\Rightarrow KL(p||q) \geq 0
\]

In our problem, we are interested in the discrepancy between the variational distribution, \( q \), and the posterior, \( p \). However, we usually use the joint distribution of the parameters and the data, \( p^* \), instead. This is justified, because:

\[
KL(q||p) = \int_{-\infty}^{\infty} q(\theta) \log \frac{q(\theta)}{p(\theta)} \, d\theta = \int_{-\infty}^{\infty} q(\theta) \log q(\theta) \, d\theta - \int_{-\infty}^{\infty} q(\theta) \log p(\theta) \, d\theta \\
= \int_{-\infty}^{\infty} q(\theta) \log q(\theta) \, d\theta - \int_{-\infty}^{\infty} q(\theta) \log \frac{p^*(\theta)}{Z} \, d\theta \\
= \int_{-\infty}^{\infty} q(\theta) \log q(\theta) \, d\theta - \int_{-\infty}^{\infty} q(\theta) \log p^*(\theta) \, d\theta + \int_{-\infty}^{\infty} q(\theta) \log Z \, d\theta \\
= \int_{-\infty}^{\infty} q(\theta) \log \frac{q(\theta)}{p^*(\theta)} \, d\theta + \log Z \\
= KL(q||p^*) + \log Z
\]
The resulting expression is:

\[ KL(q||p) = KL(q||p^*) + \log Z \]

where \( Z \) is the normalizing constant corresponding to the model’s evidence, \( p(X) \), which does not depend on the variational distribution or the parameters associated with it. This observation makes the objective of minimizing \( KL(q||p) \) equivalent to minimizing \( KL(q||p^*) \).

We can also use the last equality to derive a useful bound on the logarithm of the marginal likelihood:

\[ KL(q||p) \geq 0 \Rightarrow KL(q||p^*) + \log Z \geq 0 \Rightarrow \log Z \geq -KL(q||p^*) \]

The term \(-KL(q||p^*)\) is known as the variational free energy and it serves as a lower bound on the logarithm of the model’s evidence. This expression is also useful for model selection, since it can be used to provide an estimate on how well the variational distribution fits the posterior. Following the above, in many problems we prefer to maximize this term, instead of minimizing \( KL(q||p^*) \).

Another way to rewrite the objective function, using the joint distribution, is:

\[
KL(q||p^*) = \int_{-\infty}^{\infty} q(\theta) \log \frac{q(\theta)}{p^*(\theta)} d\theta = \int_{-\infty}^{\infty} q(\theta) \log \frac{q(\theta)}{p(X|\theta)p(\theta)} d\theta \\
= \int_{-\infty}^{\infty} q(\theta) \log q(\theta) d\theta - \int_{-\infty}^{\infty} q(\theta) \log p(X|\theta) d\theta - \int_{-\infty}^{\infty} q(\theta) \log p(\theta) d\theta \\
= \int_{-\infty}^{\infty} q(\theta) \log \frac{q(\theta)}{p(\theta)} d\theta - \int_{-\infty}^{\infty} q(\theta) \log p(X|\theta) d\theta \\
= KL(q||p) + E_{\theta \sim g}(-\log p(X|\theta))
\]

where, this time, \( p \) corresponds to the prior distribution of \( \theta \).

This equation is composed of the expected value of the negative log likelihood w.r.t. the variational distribution, plus a term that measures how far is the variational distribution is from the prior distribution of \( \theta \). This term can be interpreted as a form of penalization, favouring distributions that are not very far away from the prior [16].
1.11 Applications of Variational inference

In this section we will examine some applications of the variational framework to get a better understanding on the concept.

As we have mentioned, it is necessary to restrict the domain of the possible solutions of the problem, or else we end up with a trivial solution. A way to do so, is to constrain the solutions to be members of some parametric family, like the exponential one. An other way, is to use factorized distributions. This is a popular variational approach to impose structure on the solutions, while on the same time, there is no parametric assumption. The next step is to decide how to factorize the variational distribution. A standard way to do this, is to assume that there is no covariance between the parameters, meaning that the variational distribution is of the form:

\[ q(\theta_1, \ldots, \theta_N) = \prod_{i=1}^{N} q_i(\theta_i) \]

If we make this assumption, the resulting technique is called (naive) Mean Field Approximation [16]. In order to proceed, we have to compute the KL divergence between the joint distribution, \( p^* \), and \( q \), as defined above. In what follows, we express the problem in terms of \( -KL(q||p^*) \), instead of \( KL(q||p^*) \), so it is a maximization one. We will consider the derivation of \( q_j \), for some \( j \), treating all the terms not involving this quantity, as constant.
We will use $\Theta$ to denote the set of the parameters.

$$-KL(q||p^*) = \int_{-\infty}^{\infty} \prod_{i=1}^{N} q_i(\theta_i) \log \frac{p^*(\Theta)}{\prod_{i=1}^{N} q_i(\theta_i)} d\Theta$$

$$= \int_{-\infty}^{\infty} q_j(\theta_j) \prod_{i \neq j} q_i(\theta_i) \log p^*(\Theta) - \log \prod_{i=1}^{N} q_i(\theta_i) d\Theta$$

$$= \int_{-\infty}^{\infty} q_j(\theta_j) \prod_{i \neq j} q_i(\theta_i) \log p^*(\Theta) d\Theta -$$

$$\int_{-\infty}^{\infty} q_j(\theta_j) \prod_{i \neq j} q_i(\theta_i) \left( \sum_{i \neq j} \log q_i(\theta_i) + \log q_j(\theta_j) \right) d\Theta$$

$$= \int_{-\infty}^{\infty} q_j(\theta_j) \prod_{i \neq j} q_i(\theta_i) \log p^*(\Theta) d\Theta -$$

$$\int_{-\infty}^{\infty} q_j(\theta_j) \log q_j(\theta_j) d\theta_j + \text{const}$$

$$= \int_{-\infty}^{\infty} q_j(\theta_j) \log f_j(\theta_j) d\theta_j - \int_{-\infty}^{\infty} q_j(\theta_j) \log q_j(\theta_j) d\theta_j + \text{const}$$

where $\Theta_{-j}$ is the set of all the parameters, excluding $\theta_j$.

We have also defined:

$$\log f_j(\theta_j) = \int_{-\infty}^{\infty} \prod_{i \neq j} q_i(\theta_i) \log p^*(\Theta) d\Theta_{-j} = E_{-q_j}(p^*(\Theta))$$

the expected value of the joint distribution w.r.t. the distributions of all the $\theta_i$'s, except $\theta_j$.

It is not difficult to see that the last equation is equal to the negative KL divergence between $q_j$ and $f_j$, plus a constant term. This observation means that in order to maximize the above, we have to minimize the KL divergence. This happens when $q_j = f_j$, since the function becomes equal to zero. Taking into account that $q_j$ is a valid distribution, so it must be normalized, we conclude that:

$$q_j(\theta_j) = \frac{E_{-q_j}(p^*(\Theta))}{\int_{-\infty}^{\infty} E_{-q_j}(p^*(\Theta)) d\theta_j}$$
If we take some time to examine this equation, we will see it simply means that in order to get $q_j$ we simply have to compute the expected value of the logarithm of the joint distribution of the variables and the data w.r.t. all the other factors, except $q_j$. In practical applications it is usually preferred to use this form of the result:

$$\log q_j(\theta_j) = \mathbb{E}_{-q_i}(p^*(\Theta)) + \text{const}$$

It is also worth noting that this is not an explicit solution, because the expression for $q_j$ depends on all the other factors, $q_i$, for $i \neq j$. Therefore, in order to obtain a solution, we initialize all the distributions, $q_i$, accordingly, usually we just set them to some random distributions, and then cycle through all the factors, updating each one using the above equations [16]. It is not difficult to prove that the KL divergence decreases each time we update one of the $q_i$’s, so to acquire a solution, we repeat this pattern until the algorithm converges [22].

The main assumption we have made in the above model, is that there is no dependency among the parameters, which in turn, allowed us to factorize the variational distribution, accordingly. In practice, this assumption is rarely satisfied, so the resulting approximation might exhibit poor behaviour in capturing the posterior. An alternative would be to introduce some dependencies between the parameters, but in a way that the new model is still tractable, so, for example, we can use spanning tree or decomposable approximations. This approach is called structured Mean Field Approximation [16].

As our last application, we are going to examine a generalization of the Expectation Maximization (EM) algorithm. The idea is similar to the previous example, but it has a slight modification to account for the latent variables. In standard EM we have a set of latent variables, $z$, and a set of parameters, $\theta$. The objective is to maximize the likelihood of the data, under the model [23]. EM alternates between the following two steps:

i) Infer the posterior over the latent variables (E step)

ii) Provide a point estimate for the parameters (M step)

In a purely Bayesian setting we are not interested in maximizing the likelihood, but rather in obtaining the posterior distribution of the parameters. This can be done using our previous application, the Mean Field Approximation. We assume that the approximate posterior has the following
The form of this approximation implies that the parameters and the latent variables are independent, as well as the latent variables are iid, conditioned on $\theta$. Doing calculations similar to those in the derivation of the Mean Field updates, we find that the solutions to this problem are:

$$\log q(z) = \mathbb{E}_{q(\theta)}(\log p(X, \theta, z)) + \text{const}$$

and

$$\log q(\theta) = \mathbb{E}_{q(z)}(\log p(X, \theta, z)) + \text{const}$$

These solutions are implicit, but they provide us with an algorithm, called Variational Bayes EM (VBEM), which is the Bayesian analogue to EM [16]. This time the two steps are:

i) Update $q(z)$ (variational E step)

ii) Update $q(\theta)$ (variational M step)

In standard EM, during the E step, we use a point estimate of the parameters and compute the posterior over the latent variables. In VBEM we do the same, but instead of using a point estimate, we average over $\theta$.

The M steps are also quite similar: In EM we compute a point estimate for the parameters. In VBEM we update the hyperparameters of their distribution, instead.

Finally, it is not very difficult to prove that we can retrieve the regular EM, if we restrict the distribution of $\theta$ to be a point mass, i.e. a Dirac delta function, $q(\theta) = \delta(\theta - \theta^*)$, so at each iteration, we simply re-estimate $\theta^*$ [24].
2 Learning rate analysis

2.1 Introduction

In stochastic optimization problems, some kind of approximation, usually when computing the gradient, is needed. A typical setting is as follows: There is a loss function, which is the sum of partial losses, each one corresponding to a training data point, and the objective is to minimize the loss function. An approach that utilizes stochastic optimization, is to use the Stochastic Gradient Descent (SGD) algorithm. The difference between SGD and its deterministic counterpart (GD) is that instead of calculating the gradient for each one of the partial loss functions and then sum them, to form the full gradient, it only calculates the gradient for a mini-batch of the data points (often, just one point is used), and use it as an approximation for the full gradient. This is the standard case, but there are other ways to introduce stochasticity in such algorithms.

For large problems, where exact inference is infeasible, a well used practice in Bayesian Statistics is to use the Variational framework [16]. Here, the objective is to approximate the, unknown, posterior distribution of the parameters, with an other distribution, referred to as variational, usually a member of a well known, parametric, family. Using this approach, the objective function to be minimized takes the form of an expected value, with respect to the variational distribution, and so its gradient is also an expected value [25]. The goal here, is to tune the corresponding hyperparameters of the variational distribution so its distance from the true posterior is minimized. Most of the times, the calculation of this expected value is not straightforward, so it can be time consuming and computationally intense. A way to work around this problem, is to approximate this quantity in the following manner: Sample from the variational distribution, compute the desired expression on the sampled values and use their mean value as an approximation of the expected value [25].

The procedure described in the past lines, on how to calculate the expected value, follows the same reasoning, as the mini-batch case, in the sense that instead of using the true gradient, we sample from the distribution of
the data, so it can be combined with SGD, to optimize the variational parameters.

The objective of the remaining, is to examine several ways to set the value of the learning rate during training. To achieve this, we consider the updates of both SGD and GD, assuming that the optimal learning rate for the GD is known, even if, in practice, it is not. This is not a major problem, because there are options to set the GD learning rate to a near optimal value, e.g. using the Newton-Raphson method. Following the above, the goal is to minimize the distance between the deterministic and the stochastic updates, with respect to the learning rate of SGD.

For the rest of the analysis, we have the following set up: we use $x$ to denote the parameter we want to optimize, i.e. the hyperparameters of some distribution, for the Variational case, or the model’s parameters, for the mini-batch one. We also denote with:

$$L(x) = \text{E}(f(x)), \text{ the exact gradient of the function}$$

$$\overline{L}(x) = \frac{1}{S} \sum_{s=1}^{S} f(x^{(s)}), \text{ the approximation of the gradient on the sampled values}$$

We notice that the expected value of $\overline{L}$ is $L$.

### 2.2 On the selection of metric

The quantities under consideration are random variables, so every metric defined on probability spaces can be used. In particular, considering them as members of the $L^p$ spaces, assuming they are $p$-th power integrable, any of the $p$-norms:

$$\|f\|_p \equiv \left( \int_{S} |f|^p \, d\mu \right)^{1/p} = \text{E}(|f|^p)^{1/p}$$

where $f$ is a random variable, $S$ is a probability space and $\mu$ is a probability measure on $S$, is a valid candidate. Of all these metrics, the only one that forms a Hilbert space is for $p = 2$ [26]. This means that $L^2$ has a lot of useful properties. It is a vector space, it comes with an inner product, which allows for both length and angle to be measured, and it is complete. Overall, it is the generalization of an Euclidean space. These are some of the reasons, why this particular norm is selected to measure the distance between random variables for the following analysis.
2.3 One step variance reduction

The first case we consider, is to find the optimal learning rate for just one iteration, going from $t-1$ to $t$. In general, SGD and GD result in different values in each iteration. Here, we suppose that the value in the $t-1$ iteration is the same for both of them, and equal to $x_{t-1}$. So, after this assumption, we are ready to proceed with the formal part of the analysis.

The GD update is:

$$x_t^* = x_{t-1} - \eta^* L(x_{t-1})$$

while the SGD one, is:

$$x_t = x_{t-1} - \eta L(x_{t-1})$$

where $\eta^*$ is the GD learning rate, $L(x_{t-1})$ is the true gradient, in this case, an expected value, and $\bar{L}(x_{t-1})$ is the approximation of the expected value.

We are interested in minimizing the 2-norm of the difference between $x_t$ and $x_t^*$. Since the function $f(x) = \sqrt{x}$ is increasing, it suffices to minimize the expectation of the square of their difference. We start by simplifying this quantity.

We see that,

$$E((x_t - x_t^*)^2) = E((x_{t-1} - \eta\bar{L}(x_{t-1}) - x_{t-1} + \eta^* L(x_{t-1}))^2)$$

$$= E((\eta \bar{L}(x_{t-1}) - \eta^* L(x_{t-1}))^2)$$ (3)

where the expectation is taken over the Variational distribution of $\theta$. It is worth noting that, since the expected value of $\eta$ does not necessarily equals to $\eta^*$, the expectation of $\eta \bar{L}(x_{t-1}) - \eta^* L(x_{t-1})$ is not equal to zero.

Now, we can proceed with differentiating the above quantity, with respect to $\eta$, to obtain the optimal value.
2.4 Derivation of optimal $\eta$

The derivative of (3) is

$$\frac{d}{d\eta} E((x_t - x_t^*)^2) = \frac{d}{d\eta} E((\eta L(x_{t-1}) - \eta^* L(x_{t-1}))^2)$$

$$= E\left( \frac{d}{d\eta} (\eta L(x_{t-1}) - \eta^* L(x_{t-1}))^2 \right)$$

$$= 2E(L(x_{t-1})(\eta L(x_{t-1}) - \eta^* L(x_{t-1})))$$

$$= 2E(L(x_{t-1})\eta L(x_{t-1})) - 2E(L(x_{t-1})\eta^* L(x_{t-1}))$$

$$= 2\eta E(L^2(x_{t-1})) - 2\eta^* L(x_{t-1}) E(L(x_{t-1}))$$

$$= 2\eta E(L^2(x_{t-1})) - 2\eta^* L(x_{t-1}) L(x_{t-1})$$

$$= 2\eta E(L^2(x_{t-1})) - 2\eta^* L^2(x_{t-1})$$

We are allowed to swap expectation with differentiation, since the expected value is an integral, and the integration limits are not functions of $\eta$.

Setting it equal to zero, we obtain the solution

$$\eta = \frac{\eta^* L^2(x_{t-1})}{E(L^2(x_{t-1}))} \quad (4)$$

We also have to find the second derivative, to confirm that this is indeed the minimum. It is trivial to see, that this derivative is equal to $2E(L^2(x_{t-1}))$.

This expression is always non-negative, since it is the expected value of a non-negative random variable. So, indeed, this value of $\eta$ minimizes the distance between the updates.

We devote the next section on remarks on the obtained result.
2.5 Remarks

There are several ways to rewrite the expression, regarding the optimal value of $\eta$, and gain different insights into the result.

- The first way we consider, is to replace the expected value on the denominator, using the well known equality, $\text{Var}(x) = \text{E}(x^2) - \text{E}(x)^2$. By doing so, we get the following:

$$
\eta = \frac{\eta^* L^2(x_{t-1})}{\text{Var}(\bar{L}(x_{t-1})) + L^2(x_{t-1})}
$$

In this expression, the only stochastic component is the variance of $\bar{L}$. As the variance decreases, towards zero, we get that:

$$
\eta \rightarrow \frac{\eta^* L^2(x_{t-1})}{\text{Var}(\bar{L}(x_{t-1})) + L^2(x_{t-1})} \rightarrow 0 \\
\eta^* L^2(x_{t-1}) = \eta^*
$$

So, in the limit, the two step sizes coincide. Also, in the extreme case, when $\text{Var}(\bar{L}(x_{t-1})) = 0$, the same result holds.

- Another way to rewrite (4) is based on the fact that $\text{E}(\bar{L}(x_{t-1})) = L(x_{t-1})$. We substitute this, on the numerator to get that:

$$
\eta = \frac{\eta^* L^2(x_{t-1})}{\text{Var}(\bar{L}(x_{t-1})) + L^2(x_{t-1})} \xrightarrow{\text{var}(\bar{L}) \rightarrow 0} \frac{\eta^* L^2(x_{t-1})}{L^2(x_{t-1})} = \eta^*
$$

This expression resembles the ADAM one. The numerator is the square of the expected value of the gradient’s estimator. Accordingly, ADAM’s numerator, is the square root of the same quantity, but calculated in a certain way, by using moving, exponentially decaying, averages. As for the denominators, we notice that the expression in (4), uses the second moment of the same estimator. Similarly, ADAM’s denominator, is the square root of that quantity, but, again, calculated in a certain, empirical, way. So, we see that both updates, make use of the same statistical functions, but ADAM modifies them, slightly, by using their square roots.
2.6 Multi step variance reduction

So far, we have only considered the problem of setting the optimal value for $\eta$ for just one iteration, without taking into account what has already happened in the previous iterations. Most, if not all, of the best performing algorithms in stochastic optimization, use information from past iterations, to set the value of the update. For example, as we have already mentioned, ADAM algorithm calculates both the first and the second moment in an empirical way, using past information. Specifically, denoting the first moment approximation at iteration $t$ by $m_t$, and the second moment approximation by $v_t$, the expressions for updating them are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where $\beta_1, \beta_2 \in [0, 1)$, and $g_t$ is the approximation of the gradient in the $t$ iteration. Taking a closer look at these updates, we see that to calculate the value of both $m_t$ and $v_t$, the values of all the previous gradients are used. Other popular optimization algorithms, like Adagrad and Adadelta [27], also keep a track of past gradients.

Following the above, it is reasonable to take the analysis, of the previous section, a step further and include information of past iterations. A simple way to extend this reasoning is to consider a window of $n$ iterations, keep the value of $\eta$ constant, throughout these iterations and find its optimal value, subject to minimizing the sum of the distance between the GD and the SGD updates, produced in each iteration.

So, the objective is to minimize

$$\sum_{t=T-n}^{T} \mathbb{E}((x_t - x_t^*)^2)$$

with respect to $\eta$, where the window of $n$ iterations is from $T - n$ to $T$.

As we have already mentioned, the value of the SGD learning rate, $\eta$, is considered to be constant throughout these iterations, but it is not necessary to assume the same for the GD learning rate, $\eta^*$, so they are allowed to take different values in each iteration.
2.7 Derivation of optimal $\eta$

To proceed with the calculations, we have to make some simplifications. As we have already discussed, the updated values in each iteration are, in general, different for GD and SGD. The major simplification is that we are going to assume that, in each iteration, $t$, the value of the parameter in iteration $t - 1$ is the same for both algorithms. Now, we can rewrite the objective function, as follows:

\begin{equation}
(5) = \sum_{t=T-n}^{T} E((\eta \overline{L}(x_{t-1}) - \eta_t^* L(x_{t-1}))^2) 
\end{equation}

By making use of the derivative formula obtained in previous sections, we have that:

\begin{align*}
\frac{d}{d\eta} \sum_{t=T-n}^{T} E((x_t - x_t^*)^2) &= \frac{d}{d\eta} \sum_{t=T-n}^{T} E((\eta \overline{L}(x_{t-1}) - \eta_t^* L(x_{t-1}))^2) \\
&= \sum_{t=T-n}^{T} \frac{d}{d\eta} E((\eta \overline{L}(x_{t-1}) - \eta_t^* L(x_{t-1}))^2) \\
&= \sum_{t=T-n}^{T} 2\eta E(\overline{L}^2(x_{t-1})) - 2\eta_t^* L^2(x_{t-1}) \\
&= 2\eta \sum_{t=T-n}^{T} E(\overline{L}^2(x_{t-1})) - 2 \sum_{t=T-n}^{T} \eta_t^* L^2(x_{t-1})
\end{align*}

Setting this equal to zero, we get the solution:

\begin{equation}
\eta = \frac{\sum_{t=T-n}^{T} \eta_t^* L^2(x_{t-1})}{\sum_{t=T-n}^{T} E(\overline{L}^2(x_{t-1}))} 
\end{equation}

Again, we have to check the second derivative to confirm that it is the minimum. The desired derivative is equal to $2 \sum_{t=T-n}^{T} E(\overline{L}^2(x_{t-1})) \geq 0$, so this is indeed the minimum.
2.8 Remarks

- We can write equation (7) the following way: if we set

\[ a_t = \frac{L^2(x_{t-1})}{\sum_{t=T-n}^{T} E(L^2(x_{t-1}))} \]

then we get that:

\[ \eta = \sum_{t=T-n}^{T} a_t \eta^*_t \]

so, we see that the optimal solution, turns out to be a linear combination of the GD learning rates.

- If we keep the GD learning rates constant throughout the \( n \) iterations, so \( \eta^*_t = \eta^* \) for all \( t \) and rewrite (5) as:

\[ \eta = \frac{\sum_{t=T-n}^{T} \eta^* L^2(x_{t-1})}{\sum_{t=T-n}^{T} \text{Var}(L(x_{t-1})) + L^2(x_{t-1})} \]

then supposing that the the term \( \sum_{t=T-n}^{T} \text{Var}(L(x_{t-1})) \) tends to zero, then \( \eta = \eta^* \), as in the single step case.
2.9 Online interpretation

Another approach to gain insight on the obtained result is by rewriting as an online update. To do that, we have to slightly adjust the notation. So far we have considered a window of \( n \) iterations, ranging from \( T - n \) to \( T \). The equation (7) is the optimal value of \( \eta \) for the span of these iterations. So, we will use a transcript on \( \eta \), to reflect that this is the optimal value, up to \( T \) iteration. Following this modification, we can rewrite (7) as

\[
\eta_T = \frac{\sum_{t=T-n}^{T} \eta_t^* L^2(x_{t-1})}{\sum_{t=T-n}^{T} E(\mathcal{L}^2(x_{t-1}))}
\]  

(8)

We can now proceed with some calculations

\[
\eta_T = \frac{\sum_{t=T-n}^{T} \eta_t^* L^2(x_{t-1})}{\sum_{t=T-n}^{T} E(\mathcal{L}^2(x_{t-1}))} = \frac{\sum_{t=T-n}^{T-1} \eta_t^* L^2(x_{t-1})}{\sum_{t=T-n}^{T-1} E(\mathcal{L}^2(x_{t-1}))} + \frac{\eta_T^* L^2(x_{T-1})}{E(\mathcal{L}^2(x_{T-1}))} \\
\sum_{t=T-n}^{T-1} E(\mathcal{L}^2(x_{t-1})) \sum_{t=T-n}^{T} E(\mathcal{L}^2(x_{t-1})) + \frac{\eta_T^* L^2(x_{T-1})}{E(\mathcal{L}^2(x_{T-1}))} \sum_{t=T-n}^{T} E(\mathcal{L}^2(x_{t-1})) \\
\sum_{t=T-n}^{T-1} E(\mathcal{L}^2(x_{t-1})) \sum_{t=T-n}^{T} E(\mathcal{L}^2(x_{t-1})) + \eta_{single} \frac{E(\mathcal{L}^2(x_{T-1}))}{\sum_{t=T-n}^{T} E(\mathcal{L}^2(x_{t-1}))}
\]

where \( \eta_{single} \) is the optimal value of \( \eta \) in the single step variance reduction case, moving from iteration \( T - 1 \) to \( T \).
We can write the above expression in a more compact way, denoting the sum of the expectations up to iteration $T$ with $s_T = \sum_{t=T-n}^{T} E(\mathcal{T}^2(x_{t-1}))$.

So, we get that
\[
\eta_T = \eta_{T-1} \frac{s_{T-1}}{s_T} + \eta_{\text{single}} \frac{E(\mathcal{T}^2(x_{t-1}))}{s_T} \tag{9}
\]

This equation provides us with a way to express the optimal learning rate, up to iteration $T$, as a function of the corresponding learning rate, up to iteration $T-1$. This is also provides with new insight on how to interpret the result, as a linear combination of $\eta_{T-1}$ and $\eta_{\text{single}}$.

Furthermore, this is not just a linear combination of these two quantities, but a convex one. It is easy to see that both $\frac{s_{T-1}}{s_T}$ and $\frac{E(\mathcal{T}^2(x_{t-1}))}{s_T}$ are non negative. Also their sum is equal to
\[
\frac{s_{T-1}}{s_T} + \frac{E(\mathcal{T}^2(x_{t-1}))}{s_T} = \frac{s_{T-1} + E(\mathcal{T}^2(x_{t-1}))}{s_T} = \frac{s_T}{s_T} = 1
\]
so $\eta_T$ is indeed a convex combination of $\eta_{T-1}$ and $\eta_{\text{single}}$.

We can also rewrite (8) in a slightly different way, to study what happens in the limit, as the variance tends to zero. This, leads to
\[
\eta_T = \eta_{T-1} \frac{s_{T-1}}{s_T} + \eta_T \frac{L^2(x_{t-1})}{s_T} \tag{10}
\]

This is not a convex combination, since for a random variable $X$ it holds that $E(X^2) \geq E(X)^2$, but providing that the variance reduces, as the number of iteration increases, we have that
\[
\frac{s_{T-1}}{s_T} + \frac{L^2(x_{t-1})}{s_T} \xrightarrow{\text{Var}(\mathcal{L}) \to 0} \frac{s_{T-1} + E(\mathcal{T}^2(x_{t-1}))}{s_T} = \frac{s_{T-1} + E(\mathcal{T}^2(x_{t-1}))}{s_T} = \frac{s_T}{s_T} = 1
\]

In addition, it is easy to see that both $\frac{s_{T-1}}{s_T}$ and $\frac{L^2(x_{t-1})}{s_T}$ are non negative, so in the limit, when the variance is close to zero, the optimal $\eta_T$ turns out to be a convex combination of $\eta_{T-1}$ and $\eta_T^*$.
This result resembles the first remark from the one step variance reduction case. There, in the limit, as variance tends to zero, we conclude that the optimal learning rates, for both GD and SGD, coincide, so they are interchangeable. Here, in the limit, the optimal $\eta_T$ turns out to be a convex combination of $\eta_{T-1}$ and $\eta^*_T$, instead of $\eta_{single}$, as in the non limit case.
3 Gaussian approximations

3.1 Introduction

In modern large scale machine learning problems, it is often the case that methods stemming from Bayesian Statistics are employed. Their robust treatment on the subject, combined with the intuitive interpretation of standard techniques, such as Maximum a posteriori (MAP) inference, their built in resistance to over-fitting, among other features, like theoretical convergence results [28], make them an appealing approach for problems with massive amount of data.

Unfortunately, as the models get more complex and interesting, the posterior distribution gets intractable. This is due to the fact that the normalizing constant, in non conjugate, straightforward cases, is an integral over multiple dimensions, so calculating it requires computationally intensive calculations. This limitation has led to various developments in Approximate Bayesian methods.

Variational inference methods fall in the above category. The main idea is that it is not necessary to calculate the normalizing constant. Instead, one can assume that the posterior distribution has a specific form, usually a member of a well known parametric family of distributions. This is called the Variational distribution, and a standard choice is to pick it from the exponential family, which includes the Normal distribution. Following this assumption, the set of candidate distributions is infinite, for example all the Normal distributions, for all the values of $\mu$ and $\sigma^2$, so the next step is to tune these hyperparameters so the Variational distribution is as close to the true posterior as it can be.

To measure the distance between these distributions we make use of the Kullback-Leibler divergence between the variational distribution and the true posterior. This turns out to be an expected value, under the variational distribution. So, the objective is to minimize this function w.r.t. the hyperparameters of the variational distribution. Of course the quality of the resulted approximation depends on how well the variational distribution approximates the true posterior. This is usually incorporated in the trade off
between the complexity of the variational model and the efficiency of the minimization of the Kullback-Leibler divergence. An important thing to notice, is that the transformed problem is an optimization one, while the original was about finding a normalizing constant, so it is an integration problem.

3.2 The model

First of all we remind that the definition of the Kullback-Leibler divergence between two distributions, $p(\theta)$ and $q(\theta)$, is

$$KL(p||q) = \int_{-\infty}^{\infty} p(\theta) \log \frac{p(\theta)}{q(\theta)} d\theta$$

The first step to specify our model is to choose a variational distribution, to approximate the posterior. Here, we assume that this distribution is Gaussian. Also, to keep the presentation easier to follow, we will only consider the case of an one dimensional parameter, so the variational distribution is a one dimensional Normal, but the results of this section generalize to the multidimensional case.

Formally, we assume that $\theta \sim N(\mu, \sigma^2)$ and we will use $N(\theta|\mu, \sigma^2)$ to denote the pdf of this distribution. In addition, we will denote the joint probability of the parameter and the observations by $g(\theta) = p(\theta, X) = p(X|\theta)p(\theta)$. We need notation for the joint distribution, instead of the posterior, because minimizing the Kullback-Leibler divergence between the variational distribution and the true posterior, is equivalent to maximizing the following function w.r.t. the variational parameters, in this case $\mu, \sigma$:

$$\mathcal{F}(\mu, \sigma) = \int_{-\infty}^{\infty} N(\theta|\mu, \sigma^2) \log \frac{g(\theta)}{N(\theta|\mu, \sigma^2)} d\theta$$

(11)
3.3 Reparameterization

One way to proceed with eq (11) is to rewrite $\theta$ as $\theta = \mu + z\sigma$, where $z \sim N(0, 1)$. Now we change the variable in the integral, from $\theta$ to $z$, to get that:

$$F(\mu, \sigma) = \int_{-\infty}^{\infty} \frac{1}{\sigma} N(z|0, 1) \log \frac{g(\mu + z\sigma)\sigma}{N(z|0, 1)} dz = \int_{-\infty}^{\infty} N(z|0, 1) \log \frac{g(\mu + z\sigma)\sigma}{N(z|0, 1)} dz$$

$$= E_{z\sim N(0,1)}(\log g(\mu + z\sigma)) + \log \sigma + H_{z\sim N(0,1)} \quad (12)$$

where $H_{z\sim N(0,1)}$ is the entropy of the standard Normal distribution, which is constant w.r.t. $\mu, \sigma$.

The above procedure is called reparameterization and it is used to express the variable as an invertible function of another variable, such that the distribution of the new random variable does not depend on the variational parameters [25]. In this case, the distribution of $\theta$ is $N(\theta|\mu, \sigma^2)$, while the distribution of $z$ is $N(0, 1)$, so we see that it no longer depends on $\mu, \sigma$. It is a widely used technique, because it typically results in Monte Carlo estimators with lower variance than the ones from the original equation, in this case those in eq (11).

Now we are ready to proceed with maximizing $F$ w.r.t. $\mu, \sigma$, so we need the corresponding partial derivatives. After a few trivial calculations we have that:

$$\frac{\partial}{\partial \mu} F(\mu, \sigma) = \frac{\partial}{\partial \mu} E_{z\sim N(0,1)}(\log g(\mu + z\sigma)) \quad (13)$$

$$\frac{\partial}{\partial \sigma} F(\mu, \sigma) = \frac{\partial}{\partial \sigma} E_{z\sim N(0,1)}(\log g(\mu + z\sigma)) + \frac{1}{\sigma} \quad (14)$$

so we see that both partial derivatives include a term that is the derivative of an expected value. There are different approaches on how to rewrite this term in order to end up with different estimators, since most of the times the desired expectations cannot be analytically calculated, so by rewriting it in certain ways we get estimators with different variances. For the rest of the section we will mainly present the computations regarding the partial derivative w.r.t. $\mu$, since those about $\sigma$ are very similar.
A general method for estimating derivatives like the above, is to use the log derivative trick [29], also known as likelihood ratio or reinforce, which makes use of a property of the derivative of the logarithm of a function. More specifically it employs the formula \( \frac{d}{dx} f(x) = f(x) \frac{d}{dx} \log f(x) \) and its analogous in the multidimensional case.

To proceed, we first change the variable back to \( \theta \) and then use the log derivative trick. So eq (13) becomes:

\[
\frac{\partial}{\partial \mu} \mathcal{F}(\mu, \sigma) = \frac{\partial}{\partial \mu} \mathbb{E}_{z \sim N(0,1)}(\log g(\mu + z\sigma)) = \frac{\partial}{\partial \mu} \mathbb{E}_{\theta \sim N(\mu, \sigma^2)}(\log g(\theta)) =
\]

\[
= \frac{\partial}{\partial \mu} \int_{-\infty}^{\infty} \log g(\theta) N(\theta | \mu, \sigma^2) d\theta = \int_{-\infty}^{\infty} \log g(\theta) \frac{\partial}{\partial \mu} N(\theta | \mu, \sigma^2) d\theta =
\]

\[
= \int_{-\infty}^{\infty} \log g(\theta) \frac{(\theta - \mu)}{\sigma^2} N(\theta | \mu, \sigma^2) d\theta =
\]

\[
= \mathbb{E}_{\theta \sim N(\mu, \sigma^2)}(\log g(\theta) \frac{(\theta - \mu)}{\sigma^2})
\]

(15)

There are also other ways to handle eq (13), that result in different expressions. Previously, our first move was to change the variable back to \( \theta \). We could, instead, use the chain rule and change the variable back to \( \theta \) afterwards. Applying the chain rule on the function inside the expectation yields:

\[
\frac{\partial}{\partial \mu} \log g(\mu + z\sigma) = \frac{\partial}{\partial (\mu + z\sigma)} \log g(\mu + z\sigma) \cdot \frac{\partial}{\partial \mu} (\mu + z\sigma) = \frac{d}{d(\mu + z\sigma)} \log g(\mu + z\sigma)
\]

Putting them all together, we rewrite the eq (13) using the above and then changing the variable back to \( \theta \):

\[
\frac{\partial}{\partial \mu} \mathcal{F}(\mu, \sigma) = \frac{\partial}{\partial \mu} \mathbb{E}_{z \sim N(0,1)}(\log g(\mu + z\sigma)) = \mathbb{E}_{z \sim N(0,1)}(\frac{\partial}{\partial \mu} \log g(\mu + z\sigma)) =
\]

\[
= \mathbb{E}_{z \sim N(0,1)}(\frac{d}{d(\mu + z\sigma)} \log g(\mu + z\sigma)) = \mathbb{E}_{\theta \sim N(\mu, \sigma^2)}(\frac{d}{d\theta} \log g(\theta))
\]

(16)
The same procedure can be applied on eq (14), so after doing similar calculations we end up with two different expressions for the partial derivative w.r.t. $\sigma$. The log derivative trick yields:

$$\frac{\partial F(\mu, \sigma)}{\partial \sigma} = E_{\theta \sim N(\mu, \sigma^2)}(\log g(\theta)((\frac{\theta - \mu}{\sigma})^2 - 1)) \frac{1}{\sigma} + \frac{1}{\sigma}. \quad (17)$$

while using the chain rule results in:

$$\frac{\partial F(\mu, \sigma)}{\partial \sigma} = E_{\theta \sim N(\mu, \sigma^2)}(\frac{\theta - \mu}{\sigma} \frac{d}{d\theta} \log g(\theta)) + \frac{1}{\sigma}, \quad (18)$$

### 3.4 The problem

In the above section, we derived different expressions for the same quantities. Specifically, we saw that

$$\frac{\partial F(\mu, \sigma)}{\partial \mu} = E_{\theta \sim N(\mu, \sigma^2)}(\log g(\theta)(\frac{\theta - \mu}{\sigma^2})) = E_{\theta \sim N(\mu, \sigma^2)}(\frac{d}{d\theta} \log g(\theta)) \quad (19)$$

and that

$$\frac{\partial F(\mu, \sigma)}{\partial \sigma} - 1 = E_{\theta \sim N(\mu, \sigma^2)}((\theta - \mu) \frac{d}{d\theta} \log g(\theta)) = E_{\theta \sim N(\mu, \sigma^2)}(\log g(\theta)((\frac{\theta - \mu}{\sigma})^2 - 1)) \quad (20)$$

Each of these equalities can be used in order to obtain an estimator for the corresponding quantity. For the rest, we will, again, be concerned only with the computations about the estimators regarding the partial derivative w.r.t. $\mu$, although we will also state the results about the partial derivative w.r.t. $\sigma$. This is, because the reasoning is quite similar for these cases, and having seen one of the two, makes the derivation of the other pretty straightforward.
The last equality in eq (19) gives us two different ways of estimating
the desired derivative. We can draw $S$ samples from the $N(\theta|\mu,\sigma^2)$
distribution, and then estimate the derivative using either

$$\frac{1}{S} \sum_{s=1}^{S} \frac{d}{d\theta} \log g(\theta)|_{\theta = \theta^{(s)}}$$ \hspace{1cm} (21)$$

or

$$\frac{1}{S} \sum_{s=1}^{S} \log g(\theta^{(s)}) \frac{\theta^{(s)} - \mu}{\sigma^2}$$ \hspace{1cm} (22)$$

The expected value of both estimators is equal to the true partial
derivative, so they are both unbiased. In addition, they are consistent, meaning
that

$$\frac{1}{S} \sum_{s=1}^{S} \frac{d}{d\theta} \log g(\theta)|_{\theta = \theta^{(s)}} \xrightarrow{S \to \infty} E_{\theta \sim N(\mu,\sigma^2)}\left(\frac{d}{d\theta} \log g(\theta)\right)$$

and

$$\frac{1}{S} \sum_{s=1}^{S} \log g(\theta^{(s)}) \frac{\theta^{(s)} - \mu}{\sigma^2} \xrightarrow{S \to \infty} E_{\theta \sim N(\mu,\sigma^2)}(\log g(\theta) \frac{\theta - \mu}{\sigma})$$

as a consequence of the law of large numbers. Since they both converge
to true expectation, it is reasonable to ask if there is any reason to prefer one,
over the other. This question can be answered by examining the variance
of each estimator. We know that both will converge to the desired value,
so the lower the variance, the faster the convergence. For completeness, the
resulting estimators from eq (20), are

$$\frac{1}{S} \sum_{s=1}^{S} (\theta^{(s)} - \mu) \frac{d}{d\theta} \log g(\theta)|_{\theta = \theta^{(s)}}$$ \hspace{1cm} (23)$$

and

$$\frac{1}{S} \sum_{s=1}^{S} \log g(\theta^{(s)}) \left(\frac{\theta^{(s)} - \mu}{\sigma}\right)^2 - 1 \right).$$ \hspace{1cm} (24)$$
3.5 Intuition and a simple example

Before going on with the formal derivation of the results, it is worth taking some time to take a closer look at these estimators. The eq (21) includes information about the derivative of the function, while eq (22) doesn’t. On the other hand, we can rewrite (22), as

$$\frac{1}{S} \sum_{s=1}^{S} \frac{1}{\sigma^2} (\log g(\theta^{(s)}) \theta^{(s)} - \log g(\theta^{(s)}) \mu),$$

and this looks like a way to capture how the function changes around the mean, $\mu$, where $\sigma^2$ acts as a normalizer. So, it seems that (22) tries to stochastically approximate both the derivative of $\log g(\theta)$ and the desired expectation.

We use an example to demonstrate the convergence rate of each estimator. Let’s suppose that $g(\theta) = \exp(-\theta^2)$. We also assume that the variational distribution of $\theta$ is $N(1, 1)$. Plugging in the specifics of this example, the estimator (21) becomes

$$\frac{1}{S} \sum_{s=1}^{S} -2\theta^{(s)}$$

while (22) turns out to be

$$\frac{1}{S} \sum_{s=1}^{S} \theta^{(s)}^2 (1 - \theta^{(s)})$$

It is also easy to see, using either (15) or (16), that the true expectation is equal to -2. Next we draw 500 samples from the variational distribution. In Figure 2, we plot the estimated values as a function of the number of samples used.
3.6 Taylor approximation

In this section we present the main results of this chapter. In order to get approximate results on the variance of the estimators, we are going to use the second order Taylor expansion of \(\log g(\theta)\) around \(\mu\). Also, to simplify the notation, we will denote \(\log g(\theta)\) by \(f(\theta)\), for the rest of the section.

Assuming that \(f\) satisfies the required conditions [4], we have that

\[
f(\theta) = f(\mu) + f'(\mu)(\theta - \mu) + \frac{1}{2} f''(\mu)(\theta - \mu)^2 + O(\theta^3)
\]

\[
\approx f(\mu) + f'(\mu)(\theta - \mu) + \frac{1}{2} f''(\mu)(\theta - \mu)^2 \tag{25}
\]

Now we are ready to study the variance of estimator (22). So

\[
\text{Var}(\frac{\theta - \mu}{\sigma^2} f(\theta)) \approx \text{Var}(\frac{\theta - \mu}{\sigma^2} f(\mu)) + \text{Var}(\frac{(\theta - \mu)^2}{\sigma^2} f'(\mu)) + \text{Var}(\frac{(\theta - \mu)^3}{2\sigma^2} f''(\mu))
\]

\[
+ 2\text{Cov}(\frac{\theta - \mu}{\sigma^2} f(\mu), \frac{(\theta - \mu)^2}{\sigma^2} f'(\mu)) + 2\text{Cov}(\frac{\theta - \mu}{\sigma^2} f(\mu), \frac{(\theta - \mu)^3}{2\sigma^2} f''(\mu))
\]

\[
+ \text{Cov}(\frac{(\theta - \mu)^2}{\sigma^2} f'(\mu), \frac{(\theta - \mu)^3}{2\sigma^2} f''(\mu))
\]

\[
= \frac{f(\mu)^2}{\sigma^4} \text{Var}(\theta - \mu) + \frac{f'(\mu)^2}{\sigma^4} \text{Var}((\theta - \mu)^2) + \frac{f''(\mu)^2}{4\sigma^4} \text{Var}((\theta - \mu)^3)
\]

\[
+ \frac{f(\mu)f''(\mu)}{\sigma^4} \text{Cov}(\theta - \mu, (\theta - \mu)^3)
\]

\[
= \frac{f(\mu)^2}{\sigma^4} \sigma^2 + \frac{f'(\mu)^2}{\sigma^4} 2\sigma^4 + \frac{f''(\mu)^2}{4\sigma^4} 15\sigma^6 + \frac{f(\mu)f''(\mu)}{\sigma^4} 3\sigma^4
\]

\[
= \frac{f(\mu)^2}{\sigma^2} + 2f'(\mu)^2 + \frac{15f''(\mu)^2}{4} + 3f(\mu)f''(\mu) \tag{26}
\]

since \(\text{Cov}((\theta - \mu), (\theta - \mu)^2) = \text{Cov}((\theta - \mu)^3, (\theta - \mu)^2) = 0\),

\(\text{Cov}(\theta - \mu, (\theta - \mu)^3) = 3\sigma^4\), \(\text{Var}(\theta - \mu) = \sigma^2\), \(\text{Var}((\theta - \mu)^2) = 2\sigma^4\)

and \(\text{Var}((\theta - \mu)^3) = 15\sigma^6\).
We differentiate eq (25) to get an expression for the derivative of $f$, so the variance of estimator (21) is approximately

$$\text{Var}(f'(\theta)) \simeq \text{Var}(f'(\mu) + f''(\mu)(\theta - \mu)) = f''(\mu)^2 \sigma^2$$  \hspace{1cm} (27)$$

We proceed with subtracting the variances:

$$\text{Var}\left(\frac{\theta - \mu}{\sigma^2} f(\theta)\right) - \text{Var}(f'()) \simeq \frac{f(\mu)^2}{\sigma^2} + \frac{2}{2} f'(\mu)^2 + \frac{11 f''(\mu)^2 \sigma^2}{4} + 3 f(\mu) f''(\mu)$$

$$\geq \left(\frac{f(\mu)}{\sigma} + \frac{3}{2} \sigma f''(\mu)\right)^2 + 2 f'(\mu)^2 \geq 0$$

The above means that $\text{Var}(f'(\theta)) \leq \text{Var}\left(\frac{\theta - \mu}{\sigma^2} f(\theta)\right)$, so estimator (21) has approximately lower variance than estimator (22), as it was expected.

Following the same reasoning, the variance of (24) is approximately:

$$\text{Var}(f(\theta)(\frac{(\theta - \mu)^2}{\sigma^2} - 1)) \simeq 2 f(\mu)^2 + 10 f'(\mu)^2 \sigma^2 + \frac{74}{4} f''(\mu)^2 \sigma^4 + 10 f(\mu) f''(\mu) \sigma^2$$  \hspace{1cm} (28)$$

while for (23):

$$\text{Var}((\theta - \mu)f'(\theta)) \simeq f'(\mu)^2 \sigma^2 + 2 f''(\mu)^2 \sigma^4$$  \hspace{1cm} (29)$$

As before, we subtract the variances:

$$\text{Var}(f(\theta)(\frac{(\theta - \mu)^2}{\sigma^2} - 1)) - \text{Var}((\theta - \mu)f'(\theta)) \simeq \frac{33}{2} f''(\mu)^2 \sigma^4 + 10 f(\mu) f''(\mu) \sigma^2 + 2 f(\mu)^2$$

$$+ 9 f'(\mu)^2 \sigma^2 \geq 9 f'(\mu)^2 \sigma^2 \geq 0$$

because the polynomial $\frac{33}{2} f''(\mu)^2 \sigma^4 + 10 f(\mu) f''(\mu) \sigma^2 + 2 f(\mu)^2$ has no real root and $2 f(\mu)^2$, so it is positive for any value of $\sigma$. It is no surprise that employing the log derivative trick resulted in an estimator with higher variance, again, since it did not take into account any information about the derivative of $f$. 

---

42
3.7 Remarks

- Looking at eq (26), we see that the variance of the estimator is inversely proportional to the variance of the variational distribution, while, on the other hand, the variance in eq (27) is proportional to the variational one. The behaviour of the former can be considered odd, since it means that the more the variational distribution gets centred around the mean, as it incorporates more information, the worse, estimator (22), performs. In the limit, as the variational variance approaches zero, the variance of (22) becomes infinite. Estimator (21) does not suffer from this behaviour, since (27) means that the smaller the variational variance, the better the resulting estimates.

- In case $f$ is a linear function we have that $f''(\theta) = 0$ for every $\theta$. In turn, a consequence is that the variance in eq (27) becomes equal to zero, so it estimates perfectly the true expectation. In contrast, eq (26) is not equal to zero and estimator (22) continues to exhibit the peculiar behaviour of the above remark.
4 Experiments

4.1 The proposed algorithm

In the previous sections we derived a schedule for updating the learning rate, when performing optimization using SGD. In this section, we use this result to develop a new stochastic optimization algorithm. To do so, we need to calculate the learning rate in each iteration, namely \( \eta_T \). After inspecting equation (9), we see that, in order to proceed, we need the value of the following quantities: \( \text{E}(L(x_{t-1})) \), for \( \eta_{\text{single}} \), and \( \text{E}(L^2(x_{t-1})) \), where \( L(x_{t-1}) \) is the estimator of the gradient of the loss function, at \( x_{t-1} \). When the loss function is made by adding partial costs, corresponding to each training data point, these computations can become computationally intractable, as the number of the training instances scales up. A widely used alternative, is to approximate these expected values, using a mini-batch of the training dataset. In our case, we can use the following estimators:

\[
\text{E}(L(x_{t-1})) \approx \frac{1}{N} \sum_{i=1}^{N} f_i(x_{t-1})
\]

and

\[
\text{E}(L^2(x_{t-1})) \approx \frac{1}{N} \sum_{i=1}^{N} f_i^2(x_{t-1})
\]

where \( N \) is the number of data points in the mini-batch, and \( f_i \) is the gradient of the partial loss function, corresponding to data point \( i \).

To proceed with the actual implementation, we will not use equation (9) as it is, because there are some redundant calculations, but we will use equation (10). The above leads to a new algorithm for stochastic optimization in machine learning that allows dimension-wise adaptive learning rates similar to Adam and Adagrad.
To apply this algorithm we iterate the following steps, until convergence:

- Pick up the next mini-batch, of size $N$.
- Calculate the quantities $\frac{1}{N} \sum_{i=1}^{N} f_i(x_{t-1})$ and $\frac{1}{N} \sum_{i=1}^{N} f_i^2(x_{t-1})$.
- Update $s_T = \sum_{t=0}^{T} \mathbb{E}(L^2(x_{t-1}))$, by adding $\frac{1}{N} \sum_{i=1}^{N} f_i^2(x_{t-1})$ to $s_{T-1}$
- Compute $\eta_T$, using
  \[
  \eta_T = \eta_{T-1} \frac{s_{T-1}}{s_T} + \eta^* \frac{\left(\frac{1}{N} \sum_{i=1}^{N} f_i(x_{t-1})\right)^2}{s_T}
  \]
- Update the parameter, assuming it is a minimization problem, using
  \[
  x_T = x_{T-1} - \eta_T \frac{1}{N} \sum_{i=1}^{N} f_i(x_{t-1})
  \]

We note that in practical application we could avoid calculating the fraction $\frac{s_{T-1}}{s_T}$, and replace it by a constant value, $\beta$, close to one, since, as the iterations go by, this quantity will be approximately 1.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Optimal value</th>
<th>Train accuracy</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>-0.34087202802841576</td>
<td>92.22909090909090%</td>
<td>92.14%</td>
</tr>
<tr>
<td>Adam</td>
<td>-0.35499209954061617</td>
<td>91.75818181818182%</td>
<td>91.51%</td>
</tr>
<tr>
<td>Adagrad</td>
<td>-0.3455285756124631</td>
<td>92.17272727272727%</td>
<td>92.05%</td>
</tr>
<tr>
<td>Nesterov’s SGD</td>
<td>-0.34461700150626201</td>
<td>92.02545454545454%</td>
<td>92.15%</td>
</tr>
</tbody>
</table>

Table 1: Summary of the obtained results

4.2 Logistic Regression

The first model we are going to use, in order to evaluate the performance of our proposed algorithm, is Logistic Regression. In addition, we are going to make use of L2 regularization, to regulate the complexity of the model. We will train the model on the MNIST dataset, which consists of handwritten digits. The objective function we want to maximize is:

$$
\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log y_{nk} - \frac{\lambda}{2} \sum_{k=1}^{K} \|w_k\|^2
$$

where $t_{nk} = 1$ if and only if the n-th data point belongs to class $k$. As usual, $w_k$ denotes the parameters associated with the $k$-th class, and $y_{nk}$ is equal to the probability that the n-th data point belongs to class $k$, so

$$
y_{nk} = \frac{e^{w_k^T x_n}}{\sum_{k=1}^{K} e^{w_k^T x_n}}
$$

We trained the model using our algorithm, the Adam algorithm, the Adagrad algorithm and the Nesterov’s SGD one. We used a mini-batch of size 100, for all the algorithms.

For our algorithm we have to specify a value for the hyper-parameter $\eta^*$. Experiments hinted that an empirical rule to set this value, is that the fraction $\frac{\eta^*}{N}$, is $O(10^{-1})$. Consequently, since $N = 100$, we set $\eta^* = 10$. For Adam, we did not use a constant learning rate, but one that, at iteration $t$, was equal to $\frac{1}{\sqrt{t}}$. For the Adagrad algorithm, we set the learning rate to 0.1, throughout all of the iterations. Finally, the Nesterov’s SGD algorithm,
was employed to train our model. For the learning rate, we used the same schedule as with Adam, so, at iteration \( t \), the learning value was equal to \( \frac{1}{\sqrt{t}} \).

The results are summarized in Table 1. Our proposed algorithm achieved the best optimal value, as well as higher accuracy values, excluding Nesterov’s SGD test accuracy, which was slightly higher. In Figure 3 we present the plots of the values of the cost function, using the above algorithms. In addition, in Figure 3, we see how close the solution resulted from our algorithm, is to those from the other algorithms. The x-axis represents the value of the 2-norm of their difference, while the y-axis is about the infinity norm. A look at this plot, shows that the resulted solution is closest to the Adam one, considering the infinite norm.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Optimal value</th>
<th>Train accuracy</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>-0.19046138189099659</td>
<td>97.54363636363637%</td>
<td>96.87%</td>
</tr>
<tr>
<td>Adam</td>
<td>-0.19724928354960583</td>
<td>97.57272727272728%</td>
<td>96.98%</td>
</tr>
<tr>
<td>Adagrad</td>
<td>-0.18632669080874764</td>
<td>97.76181818181818%</td>
<td>97.09%</td>
</tr>
<tr>
<td>Nesterov’s SGD</td>
<td>-0.22390824425170944</td>
<td>96.08727272727273%</td>
<td>95.92%</td>
</tr>
</tbody>
</table>

Table 2: Summary of the obtained results

4.3 Neural Network

In addition to the previous model, we also trained a Neural Network with one hidden layer, consisting of 200 hidden units, using the same algorithms, as well as the MNIST dataset. The cost function is the same as before:

\[
\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log y_{nk} - \frac{\lambda}{2} \left( \sum_{k=1}^{200} \|w_{1k}\|^2 + \sum_{k=1}^{K} \|w_{2k}\|^2 \right)
\]

where \( t_{nk} = 1 \) if and only if the n-th data point belongs to class k, and the subscript 1 denotes the weights associated with the hidden layer, while the subscript 2 is used for the output layer ones. Also, \( y_{nk} \) is the output of the Neural Network. The activation function of the hidden layer, is:

\[
h(x) = \log(1 + e^x)
\]

while the activation of the output layer is the same as the one we used for the Logistic Regression experiment, the softmax function.

For this experiment we used didn’t use a constant value for the parameter \( \eta^* \), but we set it equal to \( \frac{80}{\sqrt{t}} \), where \( t \) is the number of iterations. For the Adam algorithm, we set the learning rate equal to 0.001, throughout all the iterations. A constant learning rate schedule was used for Adagrad as well, as it was set to 0.1, for all iterations. For Nesterov’s SGD we tested several different learning rates, but none of them resulted in a good performance. A constant value of 0.01 yielded the best results, although they we are still not as good as those obtained using the other algorithms.
Table 2 holds a summary of the results. We see that Adagrad achieved both the best optimal value, and higher accuracies, while our algorithm did better than Adam at finding the optimal value. A look at figure 4 shows that both Adagrad and our algorithm needed about the same number of iterations in order to converge, where Adam needed about half the amount.

Overall, our algorithm performed similar to some of the best stochastic optimization algorithms. Note that choosing a different norm in the derivation of the algorithm, results in a different schedule for the learning rate, so it can be considered as a family of algorithms. In that view, the proposed algorithm is just a member of the aforementioned family. Further research into ways to set the parameter $\eta^*$, as well as to approximate the first and second moment of the gradient estimator may result in better performance.
References


