# Stochastic Approximations and Efficient Learning

Léon Bottou
AT&T Labs - Research,
200 S. Laurel Avenue,
Middletown NJ 07748, USA
email: leonb@research.att.com

Noboru Murata
Waseda University
3-4-1 Ohkubo, Shinjuku
Tokyo 169-8555, JAPAN
Email: murata@elec.waseda.ac.jp

Running Title: Stochastic Approximations and Efficient Learning

Contact Author: Léon Bottou.

## INTRODUCTION

Many learning algorithm work by repeatedly picking one example and updating the learning system parameters on the basis of this example only. The evolving state of the parameters is the only trace of the previously seen examples. These algorithms, called *stochastic algorithms* or *online algorithms* date back to early works on recursive identification and learning systems (Robbins and Monro, 1951; Widrow and Hoff, 1960; Amari, 1967; Tsypkin, 1971). This was the dawn of the computer age. Algorithmic simplicity was a practical requirement.

The analysis of online algorithms is much more difficult than that of ordinary optimization algorithms. Practical successes in signal processing (Widrow and Stearns, 1985) motivated the creation of sophisticated mathematical tools known as *stochastic approximations* (Ljung and Söderström, 1983; Benveniste, Metivier and Priouret, 1990).

Online algorithms are still extremely useful because they enjoy significant performance advantages for large scale learning problems (Le Cun et al., 1998). This paper describes their properties using the stochastic approximation theory as a very broad framework, and provides a brief overview of newer insights obtained information geometry (see NEUROMANIFOLDS AND INFORMATION GEOMETRY) and replica calculations (see STATISTICAL MECHANICS OF ON-LINE LEARNING AND GENERALIZATION).

The first section describes and illustrates a general framework for neural network learning algorithms based on stochastic gradient descent. The second section presents stochastic approximation results describing the *final phase*. The third section discusses the conceptual aspects of the *search phase* and comments some of the newest results.

## STOCHASTIC GRADIENT

This section presents a simplified version of the general framework for stochastic gradient learning algorithms introduced in (Tsypkin, 1971). Several examples of neural network learning algorithms are then discussed in this context.

### Stochastic Gradient and Back-propagation

Let us introduce the necessary concepts using the well-known example of multilayer networks (see PERCEPTRONS, ADALINES, AND BACKPROPAGATION). Each example $z$ is a pair composed of an input pattern $x$ and a desired output $y$. The output of a multilayer network is a differentiable function $f(x,w)$ of the input pattern $x$ and the network weight vector $w$. The performance of the network on a particular example $z=(x,y)$ is measured by a *loss function* $Q(z,w)$. In the multilayer network case, the loss function usually is the squared distance between the desired output $y$ and the network output $f(z,w)$:

$$Q_{\text{mse}}(z,w) \;\stackrel{\triangle}{=}\; \tfrac{1}{2}\left(y - f(x,w)\right)^2 \tag{1}$$

One seeks a weight vector $w$ that minimize the loss function averaged over the examples. There is however a subtlety in how we define this average. Do we average over the few examples available in the training set, or over all the examples that Nature cares to produce? The first average is named *empirical risk* and measures only the training set performance. The second average is called the *expected risk* and measures the much more interesting generalization performance (Vapnik, 1998).

Both empirical and expected risks can be written as:

$$C(w) \;\triangleq\; \mathbf{E}_z\, Q(z,w) \;\triangleq\; \int Q(z,w)\, dP(z) \tag{2}$$

where $dP(z)$ represents a probability distribution of examples and $\mathbf{E}_z$ represents the expectation with respect to this distribution. In the case of the expected risk, $dP(z)$ is an unknown *ground truth distribution* representing the laws of Nature. In the case of the empirical risk, $dP(z)$ is a discrete distribution defined by the training set examples $z_{(1)} \ldots z_{(L)}$. The expectation (2) then reduces to a simple average $1/L \sum_i Q(z_{(i)}, w)$.

It has often been suggested (Rumelhart et al., 1986) to minimize the risk (2) using a gradient descent algorithm. Each iteration of the algorithms moves the weights along the steepest descent direction,

$$w_{t+1} \;=\; w_t - \gamma_t \int \nabla_w\, Q(z, w_t)\, dP(z) \tag{3}$$

where $\gamma_t$ is an adequately chosen positive learning rate and $\nabla_w\, Q$ denotes the gradient of the loss function $Q$ with respect to the weight vector $w$. This optimization algorithm cannot be used to optimize the expected risk case because the ground truth distribution is unknown. One must optimize the empirical risk instead and hope that the training set performance is a good predictor of the generalization performance. Each iteration of this *batch gradient descent* involves visiting all the training examples in order to compute the average gradient.

The *stochastic gradient descent* algorithm is a drastic simplification of the above algorithm. Each iteration consists of picking a single random example $z_t$, and updating the weight vector $w$ according to the following rule:

$$w_{t+1} \;=\; w_t - \gamma_t \nabla_w\, Q(z_t, w_t) \tag{4}$$

This algorithm illustrates the concept of *stochastic approximation.* It is hoped that iterating equation (4) behaves like iterating its mean (3) despite the noise introduced by this simplified procedure.

The stochastic gradient descent does not need to remember which examples were visited during the previous iterations. This makes this algorithm suitable for the online adaptation of deployed systems. In such a situation, the stochastic gradient descent directly optimizes the expected risk, since the examples are randomly drawn from the ground truth distribution.

The stochastic gradient descent can also pick examples from a finite training set. This procedure optimizes the empirical risk. The number of iterations is usually larger than the size of the training set. The examples are therefore presented multiple times to the network.

## General Stochastic Gradient Descent

The above example suggests a more general framework:

Let us consider a learning system whose goal consists in minimizing a cost function (2) that can be expressed as the expectation of a loss function $Q(z,w)$. Each iteration of the *generalized stochastic gradient descent algorithm* consists in picking a random example $z_t$ and updating the parameter vector $w$ using the update rule (4).

A number of well known learning algorithms can be viewed as particular cases of the generalized stochastic gradient descent algorithm. A few examples a presented below. Additional examples can be found in (Bottou, 1998). Most results discussed in this contribution are based on the general framework and therefore apply to all these learning systems alike.

**Adaline** — The early neural network learning algorithms adapt the parameters of a single threshold unit. Input patterns $x$ are recognized as members of class $y = \pm 1$ according to the sign of dot product $w'x$. Each iteration of the adaline learning algorithm (Widrow and Hoff, 1960) takes one training example $z_t = (x_t, y_t)$ and applies the *delta rule*:

$$w_{t+1} = w_t - \gamma_t (y_t - w'_t x_t) x_t \tag{5}$$

This is nothing more than an iteration of the stochastic gradient descent algorithm (4) with the following loss function:

$$Q_{\text{adaline}}(z,w) \;\stackrel{\triangle}{=}\; \tfrac{1}{2}\left(y - w'x\right)^2 \tag{6}$$

The adaline loss function does not take the discontinuity of the threshold unit into account. It is much more difficult to optimize the apparently more natural loss function $(y - \text{sign}(w'x))^2$ because its gradient is null almost everywhere

**Perceptron** — The perceptron algorithm provides a slightly different solution. The weights are updated according to the following rule if and only if the current example $z_t = (x_t, y_t)$ is mis-classified:

$$w_{t+1} = w_t + \gamma_t y_t \, x_t \tag{7}$$

This learning algorithm can be derived as an online gradient descent applied to the following loss function:

$$Q_{\text{perceptron}}(z,w) \;\stackrel{\triangle}{=}\; \tfrac{1}{2}(\text{sign}(w'x) - y)\, w'x \tag{8}$$

The loss function is not differentiable when $w'x$ is null. It has been shown that the average behavior of the stochastic gradient algorithm is not affected by such mild non-differentiabilities (Bottou, 1998).

**Unsupervised Clustering** — The $K$-Means algorithm is a popular clustering method which dispatches $K$ centroids $w_{(k)}$ in order to find clusters in a set of points $x_1, \ldots, x_L$. This algorithm can be derived by performing the online gradient descent with the following loss function.

$$Q_{\text{kmeans}}(x,w) \;\stackrel{\triangle}{=}\; \tfrac{1}{2} \min_k \, (x - w_{(k)})^2 \tag{9}$$

This loss function measures the average error achieved when each point $x$ is replaced by the nearest centroid $k^*(x)$. The mild non-differentiability of this loss function on the cluster region boundaries are handled as in the perceptron case. Equation (4) then yields the $K$-Means algorithm:

$$w_{t+1(k)} = w_{t(k)} + \gamma_t \begin{cases} x_t - w_{t(k)} & \text{if } {\scriptstyle k=k^*(x_t)} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

## Stochastic Gradient Dynamics

Given a suitable choice of the learning rates $\gamma_t$, the ordinary gradient descent algorithm (3) is known to converge to a local minimum of the cost function. This local minimum is a function of the initial parameters $w_0$. The parameter trajectory follows the meanders of the local attraction basin and eventually reaches the corresponding minimum.

The random noise introduced by stochastic gradient descent (4) disrupts this deterministic picture. The parameter trajectory can jump from basin to basin. One usually distinguish a *search phase* that explores the parameter space and a *final phase* that takes place in the vicinity of a minimum.

## FINAL PHASE

### Convergence

The final convergence phase takes place in the vicinity of a single minimum $w^*$. The following result rely on the *general convexity* hypothesis:

$$\forall \varepsilon > 0, \quad \inf_{(w-w^*)^2 > \varepsilon} (w - w^*)\, \nabla_w C(w) > 0 \tag{11}$$

This hypothesis simply states that the opposite of the gradient always points toward the minimum $w^*$. This hypothesis usually holds within the final convergence region because the cost function is locally convex.

In order to obtain convergence, the parameter updates $\gamma_t \nabla_w Q(z,w)$ must become smaller and smaller when the parameter vector approaches the optimum $w^*$. This implies that either the gradients or the learning rates must vanish in the vicinity of the optimum.

The size of the gradients can be described with a first order expansion of $\nabla_w Q$ in $w^*$. There usually are constants $A, B \geq 0$ such that the following bound holds within the final convergence region:

$$\begin{aligned} & \mathbf{E}_z \left( \nabla_w Q(z,w)^2 \right) \\ & \quad \approx \; \mathbf{E}_z \left( \nabla_w Q(z,w^*)^2 \right) + \mathbf{E}_z \left( (w - w^*) \ldots \right) \\ & \quad < \; A + B\,(w - w^*)^2 \end{aligned} \tag{12}$$

The constant $A$ must be greater than the residual variance of the gradients at the optimum $\mathbf{E}_z \left( \nabla_w Q(z,w^*)^2 \right)$. This residual variance can be zero for certain rare noiseless problems where $w^*$ simultaneously optimizes the loss for every examples. It is strictly positive in most practical cases. The average norm of the gradients then does not vanish when the parameter vector $w$ approaches the optimum. Therefore one must use *decreasing learning rates*, e.g.:

$$\sum \gamma_t^2 < \infty \tag{13}$$

The presence of constant $A$ in (12) marks a critical difference between stochastic and ordinary gradient descent. There is no such constant in the case of the ordinary gradient descent. A simple analysis then yields an expression for the maximal constant learning rate (Le Cun et al., 1998). In the stochastic gradient case, this analysis suggests that the parameter vector eventually hovers around the minimum $w^*$ at a distance roughly proportional to $\gamma_t$. Quickly decreasing the learning rate is therefore tempting. Suppose however that the learning rates decrease so fast that $\sum \gamma_t = R < \infty$. This would effectively maintain the parameters within a certain radius of their initial value. It is therefore necessary to enforce the following condition:

$$\sum \gamma_t = \infty \tag{14}$$

The general convexity hypothesis (11) and the three conditions (12), (13) and (14) are sufficient conditions for the almost sure convergence of the stochastic gradient descent (4) to the optimum $w^*$ (Bottou, 1998).

## Second Order Methods

Significant speed-ups of the final phase can be achieved using second order methods (Le Cun et al., 1998). The simplest second order method re-scales the gradient vector using the inverse of the Hessian matrix $H(w_t)$ in a manner analogous to Newton's algorithm:

$$w_{t+1} \;=\; w_t - \gamma_t \, H(w_t)^{-1} \, \nabla_w \, Q(z_t, w_t) \tag{15}$$

It is however preferable to use instead the Gauss-Newton matrix (Le Cun et al., 1998) or the Fisher Information matrix (section “Natural Gradient”). These matrices asymptotically behave like the Hessian matrix and also provide superior performance during the search phase.

State-of-the-art batch optimization algorithms (Dennis and Schnabel, 1983) reach so-called super-linear convergence speed *on the training set* using such second order methods. Such speeds are not possible with stochastic gradient because the convergence speed is determined by the learning rate schedule which is limited by the condition (14). This speed limit has often been presented as a severe drawback. However, training algorithms are about the generalization performance. Well-designed second order stochastic algorithm have been shown to saturate the Cràmer-Rao bound (Murata, 1998). This result means that batch algorithms that converge faster on the training set are merely over-training.

## SEARCH PHASE

Although our understanding of the search phase is still very incomplete, empirical and theoretical evidence indicate that stochastic gradient algorithms enjoy significant advantages over batch algorithms.

Stochastic gradient descent can take advantage of the redundancies of the training set. Consider the extreme case where a training set of size 1000 is inadvertently composed of 10 identical copies of a set with 100 samples. Averaging the gradient over all 1000 patterns gives the exact same result as computing the gradient based on just the first 100. Batch gradient descent is wasteful because it re-computes the same quantity 10 times before one parameter update. On the other hand, stochastic gradient will see a full epoch as 10 iterations through a 100-long training set.

The stochastic approximation tools only provide weak results on the search phase, e.g. (Bottou, 1998). Stronger results have been obtained using information geometry (see NEUROMANIFOLDS AND INFORMATION GEOMETRY) and statistical mechanics (see STATISTICAL MECHANICS OF ON-LINE LEARNING AND GENERALIZATION). This section reviews what new information has been brought by these new approaches.

### Information Geometry

Describing the geometry of the cost function is paramount to understanding the convergence of a training algorithm. The final phase results, for instance, are build upon two hypothesis (11, 12) that adequately summarize the geometry of the cost function in the final convergence region. This is much more difficult to achieve for the search phase because the parameter vector can explore every aspect of the cost function.

Information geometry provides an elegant description of the global geometry of the cost function. It is best introduced by casting the learning problem as a density estimation problem. A multilayer perceptron $f(x, w)$, for instance, can be regarded as a parametric regression model $y = f(x, w) + \varepsilon$ where $\varepsilon$ represents an additive Gaussian noise. The network function $f(x, w)$ then becomes part of the Gaussian location model:

$$p(z, w) \;=\; C \exp\left(-\frac{(y - f(x, w))^2}{2\sigma^2}\right) \tag{16}$$

The optimal parameters are found by minimizing the Kullback-Leibler divergence between $p(z, w)$ and the ground truth $P(z)$. This is equivalent to the familiar optimization of the mean square loss (1):

$$\mathbf{E}_z \log \frac{P(z)}{p(z, w)} = \frac{1}{\sigma^2} \mathbf{E}_z Q_{mse}(z, w) + \text{ Constant} \tag{17}$$

The essential idea consists of endowing the space of the parameters $w$ with a distance that reflects the proximity of the distributions $p(z, w)$ instead of the proximity of the parameters $w$. Multilayer networks, for instance, can implement the same function with very different weights vectors. The new distance distorts the geometry of the parameter space in order to represent the closeness of these weight vectors.

The infinitesimal distance between distributions $p(z,w)$ and $p(z,w+dw)$ can be written as follows:

$$D(w||w+dw) \approx dw'J(w)dw \tag{18}$$

where $J(w)$ is the Fisher Information matrix:

$$J(w) \triangleq \int \left(\nabla_w \log p(z,w) \nabla_w \log p(z,w)'\right) p(z,w)dz \tag{19}$$

The determinant $|J(w)|$ of the Fisher information matrix usually is a smooth function of the parameter $w$. The parameter space is therefore composed of *Riemannian domains* where $|J(w)| \neq 0$ separated by *critical sub-spaces* where $|J(w)| = 0$.

## Natural Gradient

Natural Gradient (see NEUROMANIFOLDS AND INFORMATION GEOMETRY) provides an almost ideal way to search a Riemannian domain. The gradient $\nabla_w C(w)$ defines the steepest descent direction in the Euclidian space. The steepest descent direction in a Riemannian domain differs from the Euclidian one. It is defined as the vector $dw$ which maximizes $C(w) - C(w+dw)$ in the $\delta$-neighborhood:

$$D(w||w+dw) \approx dw'J(w)dw \leq \delta. \tag{20}$$

A simple derivation then shows that multiplying the gradient by the inverse of the Fisher Information matrix yields the steepest Riemannian direction. The Natural Gradient algorithm applies the same correction to the stochastic gradient descent algorithm (4):

$$w_{t+1} = w_t - \gamma_t J^{-1}(w_t) \nabla_w Q(z, w_t), \tag{21}$$

The similarity between the update rules (15) and (21) is obvious. This link becomes clearer when the Fisher Information matrix is written in Hessian form:

$$J(w) \triangleq \int -\left(\nabla_w^2 \log p(z,w)\right) p(z,w)dz \tag{22}$$

where $\nabla_w^2$ denotes a second derivative. When the parameter approaches the optimum, distribution $p(z,w)$ becomes closer to the ground truth $dP(z)$, and the Fisher Information matrix aligns with the Hessian matrix $\nabla_w^2 \mathbf{E}_z Q(z,w)$. The natural gradient asymptotically behaves like the Newton algorithm.

## Critical Subspaces and Symmetries

Consider a neural network with two fully connected layers and $n$ hidden units. The weight space contains at least $n!$ symmetrical Riemannian domains corresponding to all permutations of the hidden units and delimited by critical sub-spaces. The cost landscape in the vicinity of a critical subspace contains multiple directions along which no first order change of the cost can be detected. Such critical sub-spaces create considerable problems for gradient descent algorithms. Batch gradient descent often remains trapped in these extremely flat regions of the cost landscape, because the gradient $\nabla_w C(w)$ is zero along those directions that would allow the algorithm to escape the critical subspace.

These problems arise because the gradient vector no longer carries enough information to describe the local cost geometry. Unlike their batch counterpart, stochastic gradient algorithms have a random component and therefore can resort to luck in picking each training example $z_t$. A lucky pick sometimes transports the parameter vector into a more interesting area of the cost landscape.

Critical subspaces have both a hierarchical and a symmetric structure (Fukumizu and Amari, 2000). Replica calculations (see STATISTICAL MECHANICS OF ON-LINE LEARNING AND GENERALIZATION) have brought new insights into the symmetry breaking properties of online algorithms. The natural gradient algorithm (21) seems to perform remarkably well (Rattray et al., 1999) in this context too.

## DISCUSSION

Online learning algorithms have been introduced in the early days of the computer age by virtue of their low computational requirements. Online learning algorithms today enjoy significant performance advantages for large scale learning problems (large dimension, millions of examples). Online algorithms are also mandatory when a system must continuously adapt while being used.

A large number of common learning algorithms can be understood as instances of the general stochastic gradient descent algorithm. The stochastic approximation theory provides a general analysis of the convergence of these algorithms. This analysis describes very accurately the behavior of online algorithms in the vicinity of a local solution. Second order online algorithms provide asymptotically optimal convergence speed.

The behavior of online algorithms during the search phase is much less understood. Information Geometry pictures the search space as an ensemble of Riemannian manifolds separated by critical sub-spaces. Critical subspaces cause considerable problems to all gradient algorithms. Online gradient algorithms can overcome these problems thanks to noise introduced by the random example selection process. Statistical mechanics methods have recently brought new insights on this topic. These results might be the precursors of vastly more efficient online learning algorithms.

## REFERENCES

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, EC-16:299–307.

* Benveniste, A., Metivier, M., and Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations.* Springer Verlag, Berlin, New York.

Bottou, L. (1998). Online Algorithms and Stochastic Approximations, 9-42. In Saad, D., editor, *Online Learning and Neural Networks.* Cambridge University Press, Cambridge, UK.

* Dennis, J. and Schnabel, R. B. (1983). *Numerical Methods For Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Fukumizu, K. and Amari, S. (2000). Local Minima and Plateaus in Hierarchical Structures of Multilayer Perceptrons *Neural Networks*, 13(3):317–320

* Le Cun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). Efficient Backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science 1524. Springer Verlag.

* Ljung, L. and Söderström, T. (1983). *Theory and Practice of recursive identification.* MIT Press, Cambridge, MA.

Murata, N. (1998). A Statistical Study of On-line Learning. In Saad, D., editor, *Online Learning and Neural Networks*, 63-92. Cambridge University Press, Cambridge, UK.

Rattray, M. and Saad, D. (1999). Analysis of Natural Gradient Descent for Multilayer Neural Networks. *Physical Review E*, 59(4):4523-4532.

Robbins, H. and Monro, S. (1951). A Stochastic Approximation Model. *Ann. Math. Stat.*, 22:400–407.

Rumelhart, D. E. and Hinton, G. E. and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition*, I, 318–362, Bradford Books, Cambridge, MA.

* Tsypkin, Y. (1971). *Adaptation and Learning in automatic systems.* Academic Press, New York.

* Vapnik, V. N. (1998). *Statistical Learning Theory.* Wiley.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. In *IRE WESCON Conv. Record, Part 4.*, pages 96–104.

* Widrow, B. and Stearns, S. D. (1985). *Adaptive Signal Processing.* Prentice-Hall.