Stochastic Gradient Learning in Neural Networks

Léon Bottou
AT&T Bell Laboratories, Holmdel, NJ 07733 USA
phone: (908) 949 6364, email: leonb@neural.att.com

Abstract

Many connectionist learning algorithms consists of minimizing a cost of the form

$$C(w) = E(J(z,w)) = \int J(z,w)dP(z)$$

where dP is an unknown probability distribution that characterizes the problem to learn, and J, the loss function, defines the learning system itself. This popular statistical formulation has led to many theoretical results.

The minimization of such a cost may be achieved with a stochastic gradient descent algorithm, e.g.:

$$w_{t+1} = w_t - \epsilon_t \nabla_w J(z, w_t)$$

With some restrictions on J and C, this algorithm converges, even if J is non differentiable on a set of measure 0. Links with simulated annealing are depicted.

Résumé

De nombreux algorithmes connexionnistes consistent à minimiser un coût de la forme

$$C(w) = E(J(z,w)) = \int J(z,w)dP(z)$$

où dP est une distribution de probabilité inconnue qui caractérise le problème, et J, le critère local, décrit le système d'apprentissage lui même. Cette formulation statistique bien connue a donné lieu à de nombreux résultats théoriques.

La minimisation d'un tel coût peut être accomplie au moyen d'un algorithme de descente stochastique de gradient, par exemple:

$$w_{t+1} = w_t - \epsilon_t \nabla_w J(z, w_t)$$

Au prix de quelques restrictions sur C et J, cet algorithme converge, même si J n'est pas dérivable sur un ensemble de mesure nulle. Des liens avec les méthodes de recuit simulé sont également soulignés.

Keywords: Learning, Loss function, Stochastic gradient, Convergence

Topic: Theory

1 Introduction

Many neural network learning algorithms explicitly minimize a cost function. The back-propagation technique, for example, uses a gradient descent algorithm for minimizing the "Mean Squared Error" criterion.

Since the cost function is an average over all training examples, the computation of its gradient requires a loop over all the examples. In the *total gradient descent algorithm*, the weights then are updated once after each sweep over the training set. Proper learning rates ensure that this algorithm converges to a local minimum of the cost function.

The *stochastic gradient descent algorithm* however has been shown to be faster, more reliable, and less prone to reach bad local minima than standard gradient descent. In this algorithm, the weights are updated after the presentation of each example, according to the gradient of the loss function, i.e. the value of the cost for this example only (Le Cun, 1987).

The convergence of such stochastic algorithms actually has been studied for a long time in adaptive signal processing (Benveniste, Metivier and Priouret, 1987); most proofs however seldom extend to such non linear cases. A convergence theorem for the stochastic back-propagation algorithm for one hidden layered networks finally was formulated in (White, 1989) (theorems 3 and 4).

This paper extends these results to a wide family of connectionist algorithms.

First of all, we present a framework for the study of stochastic gradient descent. The case of certain cost functions, non differentiable at certain points, is discussed.

In the next section, a couple of learning algorithms are shown to comply with this framework. For example, Rosenblatt's Perceptron, MacQueen's K-Means and Kohonen's LVQ2 can all be seen as stochastic gradient descent on various cost functions.

In the fourth section, we formulate two convergence theorems. Proofs for these theorems are available in (Bottou, 1991). These theorems are completed by a discussion of the links between stochastic gradient and simulated annealing.

2 Stochastic Gradient

2.1 Framework

Consider, for instance, the problem of classifying patterns x into two classes $y = \pm 1$. We assume that there is a relationship between a pattern and its class, embodied by some probability distribution over the patterns and the classes $dp(x, y)$. If we know this distribution, we know the conditional probabilities $p(y|x)$ as well, and we can solve immediately the problem using the Bayes decision rule. Learning means "Acquiring enough knowledge about $dp(x, y)$ from the examples to solve the classification problem".

An Adaline (Widrow and Hoff, 1960) actually learns by (i) selecting a class of discriminant functions, the linear functions $f(x, w) = w^T x$, and (ii) defining a measure of the quality of the system, the mean squared error:

$$C(w) = \left\langle (y - f(x, w))^2 \right\rangle$$

The brackets denote the average value, over all possible examples, of the squared difference between the output and the actual class. The Adaline cost function thus is

$$C_{\text{Adaline}}(w) = \int (y - f(x, w))^2 dp(x, y)$$

Let us extend this approach to the general problem of learning: Consider a function, $J(z, w)$, the *loss* function, that measures the cost incurred by a system defined by some parameter w processing an observation z. In the case of an Adaline again, this function is equal to $(y - w^T x)^2$ and measures how well the output y for pattern x is approached. Learning consists of finding the parameter w^* that minimizes the following cost:

$$C(w) = E(J(z, w)) = \int J(z, w) dp(z) \tag{1}$$

Since the distribution $dp(z)$ is unknown, practical algorithms always consider instead an "empirical distribution" defined by the training set. Certain algorithms aim at controlling the effects of this approximation by introducing additional "regularization terms" in the loss function (Vapnik, 1982). This work does not address such problems. We assume that the loss function $J(z, w)$ is defined, and that we can draw independent random examples from a distribution $dp(z)$.

This framework is the basis of classical statistical inference theory. Hundreds of practical algorithms (Tsypkin, 1971) have been derived.

It is therefore natural that many connectionist learning algorithms minimize some instance of this general cost function. Several examples are given in the next section. It is surprising however that most of them share the same optimization algorithm.

2.2 Stochastic gradient

Minimizing the general cost (1) apparently is not a difficult problem. If the training set is *finite*, the distribution $dP(z)$ is discrete,

$$\begin{aligned} dP(z) &= \sum_{i=1}^{N} \frac{1}{N} \delta(z - z_i) \\ C(w) &= \frac{1}{N} \sum_{i=1}^{N} J(z_i, w) \end{aligned} \tag{2}$$

and we can explicitly apply the gradient descent algorithm. Each iteration involves a sum over the N examples z_i.

$$\begin{aligned} w_{t+1} &= w_t - \epsilon_t \nabla_w C(w_t) \\ &= w_t - \epsilon_t \int \nabla_w J(z, w_t) dP(z) \\ &= w_t - \epsilon_t \frac{1}{N} \sum_{i=1}^{N} \nabla_w J(z_i, w_t) \end{aligned} \tag{3}$$

The gain ϵ_t is either a positive scalar or a symmetric positive definite matrix. This algorithm is sometimes called the *total gradient* algorithm. It is known to converge to a local minimum of the cost.

Learning however often requires a training set large enough to contain precise informations about the real phenomenon. Each iteration of algorithm (3) involves a burdening computation of the average of $\nabla_w J(z, w)$ over the entire training set.

In the *stochastic gradient* descent algorithm the average behavior of the algorithm replaces this costly averaging operation. Each iteration of the stochastic gradient algorithm consists in drawing an example z at random and applying the parameter update rule:

$$w_{t+1} = w_t - \epsilon_t \nabla_w J(z, w_t) \tag{4}$$

Such *stochastic approximations* have been introduced in (Robbins and Monro, 1951). They are widely used for adaptive signal processing. The general convergence results of gradient descent however do not apply to stochastic gradient descent; the specific study of their convergence usually is fairly complex (Benveniste, Metivier and Priouret, 1987).

Nevertheless, algorithm (4) offers several theoretical and practical advantages over the total gradient algorithm (3):

- The convergence is much faster when the examples are redundant. Only a few examples are needed to perform, on average, the equivalent of one full sweep of the total gradient method over the training set.

  Consider, for example, a training set composed of several copies of the same examples. A total gradient algorithm wastefully computes the average of many identical gradients; a stochastic algorithm gets a good estimate of this average after considerably fewer operations.

- The total gradient (3) converges to a *local minimum* of the cost function. The algorithm then cannot escape this local minimum, which is sometimes a poor solution of the problem.

  In practical situations, the gradient algorithm may get stuck in an area where the cost is extremely ill conditionned, like a deep ravine of the cost function. This situation actually is a local minimum in a subspace defined by the largest eigenvalues of the Hessian matrix of the cost.

  The stochastic gradient algorithm (4) usually is able to escape from such bothersome situations, thanks to its random behavior (Bourrely, 1989).

- Most training sets are finite. Algorithm (4) thus converges to a minimum of the empirical cost (2). This cost hopefully is related to the actual problem. If the training set is not large enough, however, "overfitting" occurs (Vapnik, 1982).

  Sometimes, however, (e.g. in adaptive filtering problems), examples are provided on the fly. Conceptually, we can draw at each iteration a new independent example z from an unknown distribution $dP(z)$ that embodies the laws of Nature. Algorithm (4) then minimizes the actual cost, which is the expectation of the loss function over this unknown distribution. [1]

2.3 Variations

The stochastic gradient algorithm (4) may be further generalized to the following algorithm

$$w_{t+1} = w_t - \epsilon_t H(w_t) \tag{5}$$

[1] The algorithm thus is *consistent* if it converges to a global minimum. Unfortunately, it may converge to a local minima. The ability of the stochastic algorithm to escape some local minima is an important step towards consistency. (cf. 4.2).

where $H(w)$ is a random function of w whose expectation is the gradient of the cost:

$$\forall w,\ E(H(w)) = \nabla_w C(w) \tag{6}$$

In particular, if the loss function $J(z,w)$ is differentiable, choosing $H(w)$ equal to its gradient $\nabla_w J(z,w)$ for some random example z, gives algorithm (4).

We may also choose $H(w)$ equal to the average of the gradient of the loss over a couple of random examples, instead of using one example only for each iteration. Using such "small batches" is a common practice in some implementations of the back-propagation algorithm.

Finally, we can apply algorithm (4) even if the loss function $J(z,w)$ is not differentiable on a subset of measure 0 of the parameter space. The derivative of J on those points may be replaced by any value, provided that the equality $E(\nabla_w(J(z,w))) = \nabla_w C(w) = \nabla_w E(J(z,w))$ is still valid. This equality is guaranteed to be true if the increases of J are bounded around each point (z,w) by an integrable function $\Phi(.,.)$.

$$\begin{aligned} &\forall x, w,\ \exists \Phi \text{ integrable}, \eta > 0\ \ \forall h, |h| < \eta \\ &\frac{1}{|h|}(J(x, w+h) - J(x,w)) < \Phi(x,w) \end{aligned} \tag{7}$$

3 Examples

Many connectionist learning procedures actually use a stochastic gradient algorithm for minimizing an explicit or implicit cost function.

Several algorithms have been designed as the minimization of an explicit cost function. Most of the time, a stochastic gradient descent algorithm is used. This class includes Adalines, Multi-Layered Networks, feed-forward networks that uses special units, like Radial Basis Functions, and algorithms minimizing special cost functions (e.g. Mutual Entropy Criterions, Maximum of Likelihood training).

Some algorithms however are not based on the minimization of a cost function. In many cases however, we can prove that a cost function exists, and that the algorithm merely minimizes it with a stochastic gradient descent. The following three sections give examples of this:

3.1 Rosenblatt's Perceptron

Rosenblatt's Perceptron (Rosenblatt, 1957) is a linear discriminant device. A pattern x is recognized as a member of class $y = +1$ if $w^T x$ is positive, as a member of class and $y = -1$ otherwise. Weights w are updated only if a misclassification occur, using the following formula:

$$w_{t+1} = w_t + 2\epsilon\ y w_t^T x$$

Actually this formula is stochastic gradient descent applied to the following cost,

$$C_{\text{Perceptron}}(w) = \int -(y - \theta(w^T x)) w^T x\ dP(x)$$

where function $\theta(t)$ is equal to 1 if $t > 0$, to -1 otherwise. Although the loss function $-(y - \theta(w^T x))w^T x$ is not differentiable when $w^T x = 0$, it meets condition (7) if x is integrable for measure dP, and one may apply algorithm (4). We get the following algorithm:

$$w_{t+1} = w_t + \epsilon_t (y - \theta(w_t^T x)) x$$

This algorithm is exactly Rosenblatt's Perceptron algorithm. In particular the weights remain unchanged if the pattern is well classified.

3.2 K-Means

K-Means (MacQueen, 1967) is a popular clustering algorithm. It dispatches K centroids $w^{(k)}$, in order to find clusters in a set of points. At each iteration, a point x is considered, and the nearest centroid w^- is adapted according to the MacQueen formula:

$$w_{t+1}^- = \frac{t-1}{t} w_t^- + \frac{1}{t} x$$

This algorithm is the stochastic gradient minimization of the quantification error, i.e. the error on the position of the points if we replace them by the nearest centroid:

$$C_{\text{Kmeans}} = \int \operatorname*{Min}_{k=1}^{K} \, (x - w^{(k)})^2 \; dP(x)$$

Again, the loss function is not differentiable on points located on the Voronoï boundaries of the set of centroids. The loss however meets condition (7) if x and x^2 are integrable for measure dP. On other points, the derivative of the loss is the derivative of the distance to the nearest centroid w^-. We can apply the stochastic gradient descent algorithm:

$$w_{t+1}^- = w_t^- + \epsilon_t (x - w_t^-)$$

This algorithm is equivalent to Mac Queen formula if we choose $\epsilon_t = \frac{1}{t}$. This choice of ϵ_t meets the usual conditions $\sum_{i=0}^{\infty} \epsilon_t = \infty$ and $\sum_{i=0}^{\infty} \epsilon_t^2 < \infty$.

3.3 Kohonen's LVQ2 rule

Kohonen's LVQ2 rule (Kohonen, Barna and Chrisley, 1988) is a powerful pattern recognition algorithm. Like K-Means, it uses a fixed set of reference points $w^{(k)}$. A class $y^{(k)}$ is associated with each point. An unknown pattern x is recognized as a member of the class associated with the nearest reference point.

For an example pattern x, let us denote by w^- the nearest reference point and by w^+ the nearest reference point among those associated with the right class y. These two points are adapted as follows:

$$\begin{aligned}
&\text{if } w^+ \neq w^- \text{ and } (x - w^+)^2 < (1+\delta)(x - w^-)^2 \\
&\quad w_{t+1}^- = w_t^- - \epsilon_t (x - w_t^-) \\
&\quad w_{t+1}^+ = w_t^+ + \epsilon_t (x - w_t^+)
\end{aligned}$$

where δ is a small positive number: If the pattern is misclassified, and if a good reference point is in the vicinity, the algorithm pushes the nearest (wrong) reference w^- away from the pattern, and pulls the nearest good reference w^+ towards the pattern.

Consider the following loss function:

$$\begin{aligned}
&\text{if } w^+ = w^- \text{ i.e. } x \text{ is well classified} \\
&\quad J_{\text{Lvq2}}((x,y), w) = 0 \\
&\text{if } (x - w^+)^2 < (1+\delta)(x - w^-)^2
\end{aligned}$$

$$J_{\mathrm{Lvq2}}((x,y),w) = \frac{(x-w^+)^2 - (x-w^-)^2}{\delta(x-w^-)^2}$$

otherwise

$$J_{\mathrm{Lvq2}}((x,y),w) = 1$$

and the corresponding cost function:

$$C_{\mathrm{Lvq2}}(w) = \int J_{Lvq2}((x,y),w)\ dP(x,y)$$

This cost function appears to be a convenient approximation of the expectation of the number of misclassifications achieved by the system (Bottou, Denker and Guyon, 1991). As usual, J_{Lvq2} is not differentiable, but meets condition (7) if x and x^2 are integrable for measure dP. Applying algorithm (4) gives the following algorithm:

if $w^+ \neq w^-$ i.e. x is misclassified

if $(x-w^+)^2 < (1+\delta)(x-w^-)^2$

$$w^-_{t+1} = w^-_t - \epsilon_t \Gamma_1 (x - w^-_t)$$
$$w^+_{t+1} = w^+_t + \epsilon_t \Gamma_2 (x - w^+_t)$$

where

$$\Gamma_1 = \frac{1}{\delta(X-w^-)^2}, \quad \Gamma_2 = \Gamma_1 \frac{(X-w^+)^2}{(X-w^-)^2}$$

This algorithm is equivalent to LVQ2. The two scalar coefficients Γ_1 and Γ_2, merely modify the proper schedule for decreasing the learning rate ϵ_t.

4 Convergence

This section presents convergence results for algorithm (5) under condition $E(H(w)) = \nabla_w C(w)$, (6).

These results are valid for any probability measure $dP(z)$: The stochastic gradient algorithm actually minimizes the expectation of the loss function defined over the probability distribution from which the examples are drawn at each iteration.

- If the examples are randomly drawn from a training set, we minimize the expectation of the loss function over the discrete probability distribution defined by the training set. In other words, the empirical cost (2) is minimized. This is the usual notion of convergence.

- If we can obtain a new random example at each iteration, algorithm (5) minimize the expectation of the loss function over a probability distribution that embodies the laws of Nature for this problem. In other words, if we have a infinite $\aleph_0$ number of examples, the cost over *future examples* is minimized.

Sufficient conditions for convergence are explicited in two theorems. They address the convergence of w_t to extremal points of the cost $C(w)$. These points may be local minima, or even maxima.

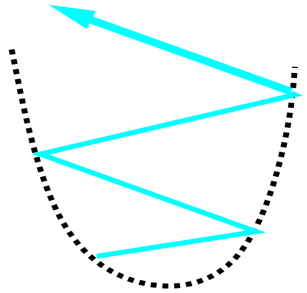

Figure 1: If $H(w)$ is to steep, the algorithm diverges.

In order to complete these results, we present arguments that establish a link between stochastic gradient descent and simulated annealing. Although not rigorous, this argument explains the ability of stochastic gradient descent to escape local minima for finally reaching a good minimum.

4.1 Sufficient Conditions

The proof of the following theorems uses a method introduced by (Gladyshev, 1965), and widely used in the mathematical study of adaptive signal processing. We first consider a discrete Lyapunov function and prove that this function is a quasi-martingale. It therefore converges with probability 1. We then look for the properties of the limit. Details are given in (Bottou, 1991).

4.1.1 Convex case

The first result applies to functions $C(w)$ that have a single minimum w^*. In addition, we require that

$$\forall \epsilon > 0, \quad \operatorname*{Inf}_{|w-w^*|>\epsilon} (w-w^*)\nabla_w C(w) > 0$$

All strictly convex functions meet these requirements. The above formula just means that the gradient of the cost is oriented towards the right direction. We thus avoid dealing with sinuous ravines, flat areas, and pathological situations around the optimum. This condition actually is a classical requirement for convex optimization.

Theorem 1 *For any measure $dP(z)$, if the cost $C(w) = E(J(z,w))$ is continuous, differentiable and has a unique minimum w^*, if this minimum fulfills the following condition,*

$$\forall \epsilon > 0, \quad \operatorname*{Inf}_{|w-w^*|>\epsilon} (w-w^*)\nabla_w C(w) > 0$$

and if the following assertions are true, then (w_t) defined by algorithm (5) converges to w^* with probability 1.

$$\begin{aligned}
&i) \quad && \forall w,\ E(H(w)) = \nabla_w C(w) \\
&ii) \quad && \sum_{t=1}^{\infty} \epsilon_t = \infty, \quad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty \\
&iii) \quad && \exists A, B,\ \forall w,\ E(H(w)^2) < A + B(w-w^*)^2
\end{aligned}$$

Assertion *i)* has been discussed in section 2.3. Assertion *ii)* is a very classical assumption on the schedule of the learning rates ϵ_t. Assertion *iii)* means that the standard deviation of the random function $H(w)$ does not grow faster than linearly [2] with the parameters w.

This latter assertion is easy to understand. If $H(w)$ takes too large values, and if ϵ_t does not decrease quickly enough, each iteration of algorithm (5) jumps on average too far over the solution w^*. The algorithm diverges (cf. Figure 1).

4.1.2 General case

Most cost functions used in neural networks however are not convex, and have both local and global minima. These minima sometimes are entire portions of the weight space. It is thus impossible to tell whether w_t converges to a particular point. The following result shows that $C(w_t)$ converges and that $\nabla_w C(w_t)$ converges to 0.

Theorem 2 *For any measure $dP(z)$, if the cost $C(w) = E(J(z,w))$ is differentiable up to the third derivatives, with bounded second and third derivatives,*
and if the following assertions are true,

$$\begin{array}{rl} i) & \forall w,\ E(H(w)) = \nabla_w C(w) \\ ii) & \sum_{t=1}^{\infty} \epsilon_t = \infty,\ \ \sum_{t=1}^{\infty} \epsilon_t^2 < \infty \\ iii) & \exists A, B,\ \forall w,\ E(H(w)^2) < A + BC(w) \\ iv) & \exists C_{\min},\ \forall w,\ C_{\min} < C(w) \end{array}$$

then $C(w_t)$ converges with probability 1 and $\nabla_w C(w_t)$ converges to 0 with probability 1.

Assertions *i)*, *ii)* and *iii)* are essentially similar to those of the previous theorem. Assertion *iv)* states that $C(w)$ must be larger than some value $C_{\min}$. This assertion hopefully is true if we want to minimize $C(w)$.

If we denote by $d(w, \mathrm{Sing}(C))$ the distance between w and the set of extremal points of C, and if we assume that

$$\forall \epsilon > 0,\quad \operatorname*{Inf}_{d(w,\mathrm{Sing}(C))} \nabla_w C(w) > 0$$

then (w_t), defined by algorithm (5), converges to the set of extremal points of C.

The assumption however means that the cost retains a non zero slope when w becomes large. This not exactly true in the case of back-propagation, because sigmoids have flat asymptotes. This is consistent with experiments: Weights tend to become large and diverge, unless we constrain them with a weight decay, twisted sigmoids, or more frequently, by using smaller target values than the sigmoid asymptotes.

4.2 Link with Simulated Annealing

This latter result only addresses the convergence of the algorithm to the set of extrema of the cost function. This set also includes maxima. Indeed, we may start the algorithm on a maximum w_0 of the loss function $J(z,w)$ for some example z_0. Consider then a trivial training

[2] This assertion may be relaxed to a higher degree polynomial, at the expense of more stringent constraints on the learning rates ϵ.

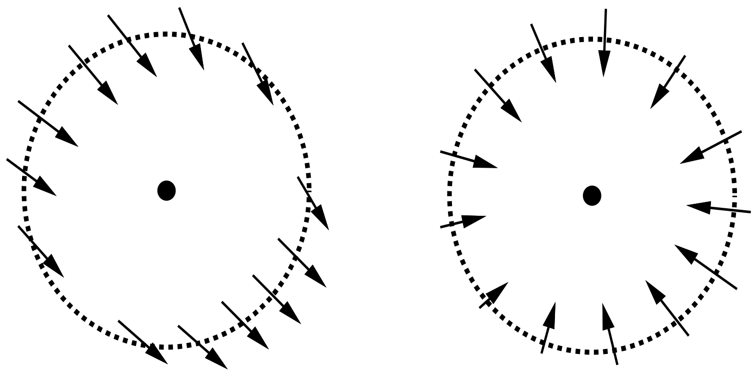

Figure 2: Two points in parameter space, surrounded by two balls. Arrows denote the gradients of the cost function. Left: An ordinary point. Right: A local maximum.

set composed of this example only. Since the gradient of the loss always is zero, algorithm (4) is trapped.

This is however unlikely to happen: When the training set becomes large the algorithm introduces a noise which cancels this unstable situation. Unless the loss function $J(z_i, w_0)$ is extremal for all examples in the training set, algorithm (4) can escape the trap.

Clearly, these theorems miss a mild hypothesis about the noise resulting from the distribution $dP(z)$. The algorithm actually behaves like a simulated annealing (Kirkpatrick, Gelatt and Vecchi, 1983), whose temperature is controlled by the learning rate ϵ_t. We give here a non rigorous argument explaining this analogy.

Let us denote $q_t(w)$ the density of probability of w_t defined by algorithm (5). By theorem 2, the support of $q_t(w)$ converges to the set of extrema of $C(w)$.

Consider a small ball $\mathcal{B}$ around some point in w space, and suppose ϵ_t is small enough to neglect quantity $\epsilon_t H(w)$ compared to the radius of the ball. Only points w_t located on the surface of the ball are likely to enter or leave the ball during the next iteration.

Each point w_S of the surface of the ball creates an infinitesimal increase $\epsilon_t q_t(w_S) H(w_S).N dw_S$, of the probability Q_t of the ball. Vector N is a normal vector to the surface of the ball.

The difference $Q_{t+1} - Q_t$ thus is proportional to the mean flow of quantity $q_t(w)H(w)$ through the surface $\partial\mathcal{B}$ of the ball:

$$\begin{aligned}
Q_{t+1} - Q_t &= \epsilon_t E\left(\int_{\partial\mathcal{B}} q_t(w_S) H(w_S).N dw_S\right) \\
&= \epsilon_t \int_{\partial\mathcal{B}} q_t(w_S) \nabla_w C(w_S).N dw_S \\
&= \epsilon_t \int_{\mathcal{B}} \mathrm{div}(q_t(w) \nabla_w C(w)) dw \\
&= \epsilon_t \int_{\mathcal{B}} \nabla_w q_t(w).\nabla_w C(w) + q_t(w)\, \mathrm{div}\, \nabla_w C(w)) dw
\end{aligned}$$

This equation gives informations about the stability of Q_t around the extrema of $C(w)$ (cf. fig. 2).

- Around a local minimum, $\mathrm{div}\, C(w)$ is positive, and $\nabla_w C(w)$ is almost zero. If Q_t is non zero, $Q_{t+1} - Q_t$ is positive and Q_t increases.

- Around a local maximum, $\mathrm{div}\, C(w)$ is negative, and $\nabla_w C(w)$ is much smaller. If we neglect the term $\nabla_w q_t(w) \nabla_w C(w)$, we have $Q_{t+1} - Q_t \leq -K Q_t$. The quantity Q_t thus decreases exponentially.

Suppose that ϵ_t decreases slowly enough to allow q_t to reach an equilibrium and to permanently stay at the equilibrium. This is a *quasi-static equilibrium* assumption similar to the simulated annealing assumptions. We thus have $Q_{t+1} - Q_t = 0$, i.e.

$$\nabla_w q_t(w).\nabla_w C(w) + q_t(w)\, \mathrm{div}\, \nabla_w C(w) = 0$$

We can thus describe the properties of q_t at equilibrium:

- When the curvature of $C(w)$ is positive, $\mathrm{div}\, \nabla_w C(w)$ is positive. If $q_t(w) \neq 0$, its gradient $\nabla_w q_t(w)$ is oriented in the opposite direction of the gradient $\nabla_w C(w)$. The deeper a minimum w_0, the larger its probability $q_t(w_0)$.
- When the curvature of $C(w)$ is negative, $\mathrm{div}\, \nabla_w C(w)$ is negative. If $q_t(w) \neq 0$, its gradient $\nabla_w q_t(w)$ is oriented in the direction of the gradient $\nabla_w C(w)$. The density $q_t(w_0)$ on the maximum w_0 should thus be large. This contradicts the stability study. Therefore, $q_t(w)$ is zero.

This equilibrium is unstable if the density q_t is zero in the positive curvature area of some attraction basins. The noise introduced by the stochastic algorithm circumvents such an instable equilibrium. When ϵ_t decreases, we reach a global minimum, or at least a very good local minimum.

Apparently, under some mild assumption on $dP(z)$, algorithm (5) converges to a very good local minimum of $C(w)$. The deeper the minimum, the larger the probability to converge to it. As far as I know, there is no rigorous proof of that fact.

5 Conclusion

Stochastic gradient descent is a powerful optimization algorithm. It converges even when the loss function is non differentiable everywhere. In addition it can escape local minima and reach a better minimum.

This result applies to a wide class of connectionist algorithms. Many neural network learning algorithms indeed can be expressed as stochastic gradient descent of an explicit or implicit cost function.

Acknowledgments

This work has been carried out at Laboratoire de Recherche en Informatique, Université de Paris XI, during my thesis. I was supported by DRET grant N° 87/808/19.

References

Benveniste, A., Metivier, M., and Priouret, P. (1987). *Algorithmes adaptatifs et approximations stochastiques*. Masson.

Bottou, L. (1991). *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole.* PhD thesis, Université de Paris XI, 91405 Orsay cedex, France.

Bottou, L., Denker, J., and Guyon (1991). Correct, robust and convenient classifier cost functions. Submitted to NIPS*91.

Bourrely, J. (1989). Parallelization of a neural network learning algorithm on a hypercube. In *Hypercube and distributed computers.* Elsiever Science Publishing.

Gladyshev, E. (1965). On stochastic approximations. *Theory of Probability and its Applications*, 10:275–278.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Kohonen, T., Barna, G., and Chrisley, R. (1988). Statistical pattern recognition with neural network: Benchmarking studies. In *Proceedings of the IEEE Second International Conference on Neural Networks*, volume 1, pages 61–68, San Diego.

Le Cun, Y. (1987). *Modèles Connexionnistes de l'Apprentissage.* PhD thesis, Université Pierre et Marie Curie, Paris, France.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In LeCam, L. M. and Neyman, J., editors, *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics, and Probabilities*, volume 1, pages 281–297, Berkeley and Los Angeles, (Calif). University of California Press.

Robbins, H. and Monro, S. (1951). A stochastic approximation model. *Ann. Math. Stat.*, 22:400–407.

Rosenblatt, F. (1957). The perceptron: A perceiving and recognizing automaton. Technical Report 85-460-1, Project PARA, Cornell Aeronautical Lab.

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

Vapnik, V. (1982). *Estimation of dependences based on empirical data.* Springer Verlag.

White, H. (1989). Learning in artificial neural networks: A statistical perspective. *Neural computation*, 1(4):425–464.

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