Journal of Machine Learning Research () Submitted ; Published

# Erratum: SGDQN is Less Careful than Expected

**Antoine Bordes** ANTOINE.BORDES@LIP6.FR
*LIP6, Université Pierre et Marie Curie, Paris, France.*

**Léon Bottou** LEONB@NEC-LABS.COM
*NEC Laboratories America, Inc., Princeton, NJ, USA.*

**Patrick Gallinari** PATRICK.GALLINARI@LIP6.FR
*LIP6, Université Pierre et Marie Curie, Paris, France*

**Jonathan Chang** JONCHANG@FACEBOOK.COM
**S. Alex Smith** ASMITH@FACEBOOK.COM
*Facebook, Palo Alto, CA, USA*

**Editor:**

## Abstract

The SGD-QN algorithm described in (Bordes et al., 2009) contains a subtle flaw that prevents it from reaching its design goals. Yet the flawed SGD-QN algorithm has worked well enough to be a winner of the first Pascal Large Scale Learning Challenge (Sonnenburg et al., 2008). This document clarifies the situation, proposes a corrected algorithm, and evaluates its performance.

**Keywords:** Stochastic Gradient Descent, Support Vector Machine, Conditional Random Fields.

## 1. Introduction

Bordes et al. (2009) propose to improve the practical speed of stochastic gradient descent by efficiently estimating a diagonal matrix for rescaling the gradient estimates. The proposed algorithm, SGD-QN, works well enough to be a winner of the first Pascal Large Scale Learning Challenge (Sonnenburg et al., 2008). A couple months after the publication of the paper, Jonathan Chang and S. Alex Smith contacted Léon Bottou regarding some curious aspects of the algorithm mathematics (see section 4.1). This initial observation was then traced back to a more subtle flaw that prevents the proposed algorithm to truly reach is design objectives.

We first explain the flaw and present experimental results describing its consequences. Then we present a corrected algorithm and evaluate its performance for training both linear Support Vector Machines (SVMs) and Conditional Random Fields (CRFs). Finally we draw updated conclusions.

## 2. Setup

Consider a binary classification problem with examples $(\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, +1\}$. Given a set of examples $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$, we obtain a linear SVM classifier by minimizing the cost

$$\mathcal{P}_n(\mathbf{w}) \;=\; \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\ell(y_i\mathbf{w}^\top\mathbf{x}_i) \;=\; \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \ell(y_i\mathbf{w}^\top\mathbf{x}_i)\right). \qquad (1)$$

Each iteration of the SGD-QN algorithm consists of drawing an independent random example $(\mathbf{x}_t, y_t)$ from the training set and computing an updated parameter vector

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \frac{1}{t+t_0}\,\mathbf{B}\,\mathbf{g}_t(\mathbf{w}_t) \quad \text{with} \quad \mathbf{g}_t(\mathbf{w}_t) = \lambda \mathbf{w}_t + \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)\, y_t\, \mathbf{x}_t \tag{2}$$

where $\mathbf{B}$ is a diagonal scaling matrix estimated on-the-fly.

In the following, expectations and probabilities refer to the discrete distribution describing the training examples randomly picked from the finite training set at each iteration. Let $\mathcal{F}_t$ denote the examples $\{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_{t-1}, y_{t-1})\}$ picked before reaching the $t$-th iteration.

We would like to find $\mathbf{B}$ such that

$$\mathbf{w}_{t+1} - \mathbf{w}_t = \mathbf{B}\left(\mathcal{P}'_n(\mathbf{w}_{t+1}) - \mathcal{P}'_n(\mathbf{w}_t) + \xi_t\right), \tag{3}$$

with an error term $\xi_t$ verifying $\mathbb{E}\left[\xi_t \middle|\, \mathcal{F}_t\right] = 0$. Following Schraudolph et al. (2007), we replace the computationally expensive gradients $\mathcal{P}'_n$ by the cheap stochastic estimates $\mathbf{g}_\tau(\mathbf{w}_t)$ and $\mathbf{g}_\tau(\mathbf{w}_{t+1})$ computed on a same single example $(\mathbf{x}_\tau, y_\tau)$,

$$\mathbf{w}_{t+1} - \mathbf{w}_t = \mathbf{B}\left(\mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t) + \zeta_t + \xi_t\right), \tag{4}$$

where $\zeta_t$ represents the additional error term introduced by this substitution. Estimating $B$ with (3) or (4) leads to the same solution if we make sure that $\mathbb{E}\left[\zeta_t \middle|\, \mathcal{F}_t\right] = 0$ as well.

The SGD-QN algorithm updates the diagonal elements of matrix $\mathbf{B}$ on-the-fly on the basis of the term-by-term ratios of the observed differences $\mathbf{w}_{t+1} - \mathbf{w}_t$ and $\mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t)$. The obvious choices $\tau = t$ and $\tau = t+1$ only require one additional gradient evaluation because the parameter update formula (2) demands the computation of all the gradients $\mathbf{g}_t(\mathbf{w}_t)$ anyway.

## 3. The Flaw

Let us now evaluate

$$\mathbb{E}\left[\zeta_t \middle|\, \mathcal{F}_t\right] = \mathbb{E}\left[\mathcal{P}'_n(\mathbf{w}_{t+1}) - \mathcal{P}'_n(\mathbf{w}_t) - \mathbf{g}_\tau(\mathbf{w}_{t+1}) + \mathbf{g}_\tau(\mathbf{w}_t) \middle|\, \mathcal{F}_t\right].$$

Let us first consider the case $\tau = t+1$. Since $\mathbf{g}_{t+1}(\mathbf{w}_{t+1})$ is a function of $(\mathbf{x}_{t+1}, y_{t+1}, \mathbf{x}_t, y_t, \mathcal{F}_t)$,

$$\begin{aligned}
\mathbb{E}\left[\mathbf{g}_\tau(\mathbf{w}_{t+1}) \middle|\, \mathcal{F}_t\right] \;&=\; \int \mathbf{g}_{t+1}(\mathbf{w}_{t+1})\; dP(\mathbf{x}_{t+1}, y_{t+1}, \mathbf{x}_t, y_t \mid \mathcal{F}_t) \\
&=\; \int \left[\int \mathbf{g}_{t+1}(\mathbf{w}_{t+1})\; dP(\mathbf{x}_{t+1}, y_{t+1})\right] \; dP(\mathbf{x}_t, y_t \mid \mathcal{F}_t)\,.
\end{aligned}$$

Since the variables $\mathbf{w}_{t+1}$ and $(\mathbf{x}_{t+1}, y_{t+1})$ are independent, the inner integral above is simply the average of $\mathbf{g}_{t+1}(\mathbf{w}_{t+1})$ for all possible $(\mathbf{x}_{t+1}, y_{t+1})$ picked from the training set. Therefore

$$\mathbb{E}\left[\mathbf{g}_\tau(\mathbf{w}_{t+1}) \middle|\, \mathcal{F}_t\right] \;=\; \int \mathcal{P}'_n(\mathbf{w}_{t+1})\; dP(\mathbf{x}_t, y_t \mid \mathcal{F}_t) \;=\; \mathbb{E}\left[\mathcal{P}'_n(\mathbf{w}_{t+1}) \middle|\, \mathcal{F}_t\right].$$

Using a similar derivation for $\mathbb{E}\left[\mathbf{g}_\tau(\mathbf{w}_t) \middle|\, \mathcal{F}_t\right]$ with $\tau = t+1$, we can easily establish that $\mathbb{E}\left[\zeta_t \middle|\, \mathcal{F}_t\right] = 0$. Therefore estimating $\mathbf{B}$ on the basis of (4) leads to the same solution as estimating $\mathbf{B}$ on the basis of (3), albeit with a higher noise level.

Such a derivation is impossible when $\tau = t$ because $(\mathbf{x}_t, y_t)$ and $\mathbf{w}_{t+1}$ are not independent. Therefore we cannot ensure that estimating $\mathbf{B}$ with (3) or (4) leads to the same result. Unfortunately, the SGD-QN paper (see Section 5.3 of (Bordes et al., 2009)) describes the algorithm with $\tau = t$.

## 4. The Consequences

In order to take maximal advantage of sparse datasets, the Flawed SGD-QN algorithm (see Figure 2 in the original paper) splits the stochastic parameter update (2) in two halves in order to schedule them separately. The first half involves only the gradient of the loss,

$$\mathbf{w} \leftarrow \mathbf{w} - (t+t_0)^{-1}\mathbf{B}\,\ell'(y_t\,\mathbf{w}^\top\mathbf{x}_t)\,y_t\,\mathbf{x}_t\,.$$

The second half involves only the gradient of the regularization term,

$$\mathbf{w} \leftarrow \begin{cases} \mathbf{w} & \text{most of the time,} \\ \mathbf{w} - \texttt{skip}\,\lambda\,(t+t_0)^{-1}\mathbf{B}\,\mathbf{w} & \text{once every } \texttt{skip} \text{ iterations.} \end{cases}$$

The Flawed SGD-QN algorithm measures the differences $\mathbf{w}_{t+1} - \mathbf{w}_t$ and $\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)$ during iterations for which the second half does nothing. Therefore, using notations $[\mathbf{x}]_i$ for the $i$-th coefficient of vector $\mathbf{x}$, and $B_{ii}$ for the terms of the diagonal matrix $\mathbf{B}$, we always have

$$\frac{[\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)]_i}{[\mathbf{w}_{t+1} - \mathbf{w}_t]_i} \;=\; \lambda - \frac{\left(\ell'(y_t\mathbf{w}_{t+1}^\top\mathbf{x}_t) - \ell'(y_t\mathbf{w}_t^\top\mathbf{x}_t)\right)\,y_t\,[\mathbf{x}_t]_i}{B_{ii}\,(t+t_0)^{-1}\ell'(y_t\mathbf{w}^\top\mathbf{x}_t)\,y_t\,[\mathbf{x}_t]_i}\,.$$

- When $[\mathbf{x}_t]_i$ is nonzero, we can simplify this expression as

  $$\frac{[\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)]_i}{[\mathbf{w}_{t+1} - \mathbf{w}_t]_i} \;=\; \lambda - \frac{\ell'(y_t\mathbf{w}_{t+1}^\top\mathbf{x}_t) - \ell'(y_t\mathbf{w}_t^\top\mathbf{x}_t)}{B_{ii}\,(t+t_0)^{-1}\ell'(y_t\mathbf{w}_t^\top\mathbf{x}_t)}\,. \qquad (5)$$

  This ratio is always greater than $\lambda$ because of the loss function $\ell$ is convex. As explained in the original paper, the coefficients $B_{ii}$ then remain smaller than $\lambda^{-1}$.

- When $[\mathbf{x}_t]_i$ is zero, the original paper uses a continuity argument to justify the equality

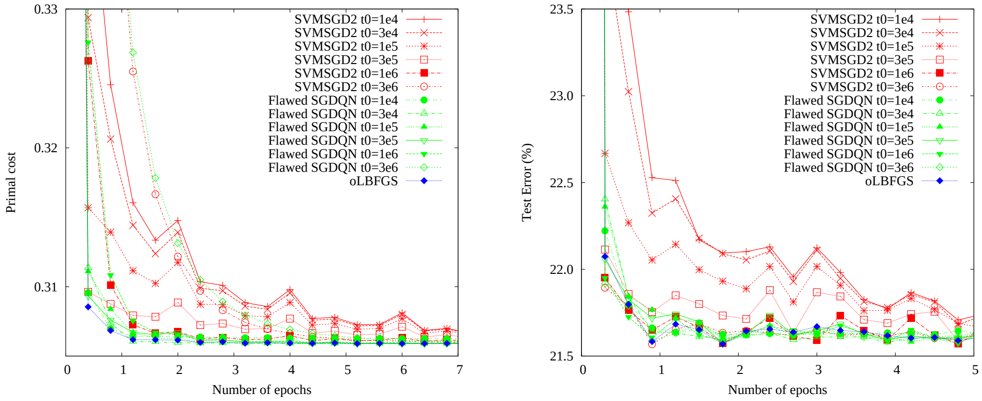
  $$\frac{[\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)]_i}{[\mathbf{w}_{t+1} - \mathbf{w}_t]_i} \;=\; \lambda\,. \qquad (6)$$

### 4.1 Impact on Dense Datasets

The coefficients $[\mathbf{x}_t]_i$ for dense datasets are rarely zero. Assume all the $B_{ii}$ are equal before being updated. All the ratios (5) will then be equal. Therefore all the $B_{ii}$ coefficients will be updated in exactly the same way and therefore remain equal. Since the $B_{ii}$ coefficients are initially equal, they *remain equal all the time*, except maybe when encountering an occasional zero in the patterns $\mathbf{x}_t$. This observation led to the discovery of the flaw.

Since the scaling matrix reduces to a scalar gain, similar results could in principle be obtained using the ordinary stochastic gradient descent with a better gain schedule. This clearly defeats the purpose of the SGD-QN algorithm design.

Figure 1 compares the evolutions of the training cost and the test misclassification error for the SVMSGD2 and the Flawed SGD-QN algorithms for selected values of the parameter $t_0$ instead of the usual heuristic defaults. We observe that there is almost always a choice of $t_0$ in SVMSGD2 that performs as well as the best choice of $t_0$ for the Flawed SGD-QN. Both algorithms perform identically poorly for excessive values of $t_0$. On the other hand, when $t_0$ is too small, the performance of Flawed SGD-QN degrades much more gracefully than the performance of SVMSGD2. In some cases, Flawed SGD-QN can even slightly outperforms SVMSGD2 because, despite the flaw, it can still update its learning rate on the course of learning. This explains partially why we have consistently obtained better results with the flawed algorithm.



DELTA DATASET

Figure 1: Plots of the training cost and test misclassification percentage versus the number of epochs for SVMSGD2 (red) and Flawed SGD-QN (green) for various values of $t_0$ on the dense Delta dataset. The Flawed SGD-QN algorithm never outperforms the best SVMSGD2.

### 4.2 Impact on Sparse Datasets

The situation is more complex in the case of sparse datasets because there is special case for updating the $B_{ii}$ coefficients when dealing with zero coefficients (6). As a result, the Flawed SGD-QN algorithm gives higher values to the scaling coefficients $B_{ii}$ when the $i$-th feature is more likely to be zero. Since this is a sensible scaling for such datasets, the Flawed SGD-QN algorithm works relatively well in the presence of sparse features.

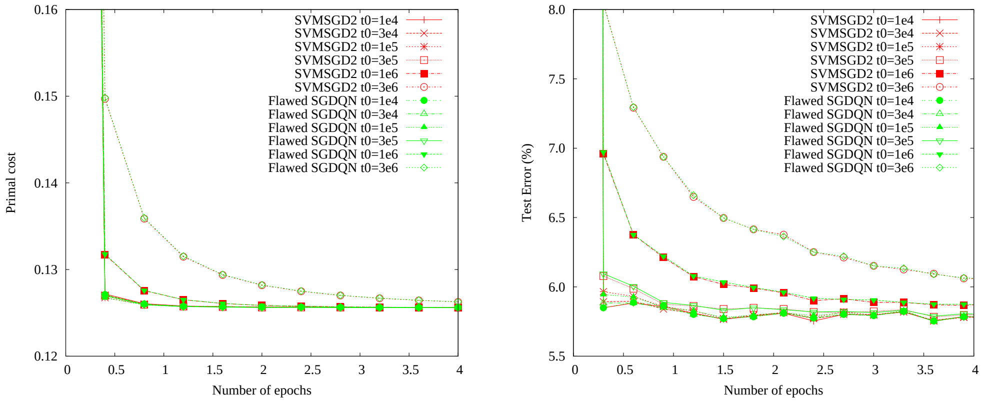
Figure 2 compares the SVMSGD2 and Flawed SGD-QN algorithms for many choices for the $t_0$ parameter on the Reuters RCV1 dataset. Unfortunately there is nothing to see there. Both algorithms reach optimal performance after processing only one half of the training set.

In order to find a more challenging sparse dataset, we have adapted both the SVMSGD2 and the Flawed SGD-QN algorithms for the optimization of Conditional Random Fields (Lafferty et al., 2001). This is an interesting case where preconditioning is difficult because the features are generated on the fly on the basis of position-independent templates.

Figure 3 compares the algorithms on the CoNLL 2000 “chunking” task (Sang and Buchholz, 2000) using the template setup provided as an example with the CRF++ code (Kudo, 2007). The Flawed SGD-QN algorithm reaches the best test performance after less epochs than the SVMSGD2 algorithm, but this does not translate into a large improvement in terms of training time.

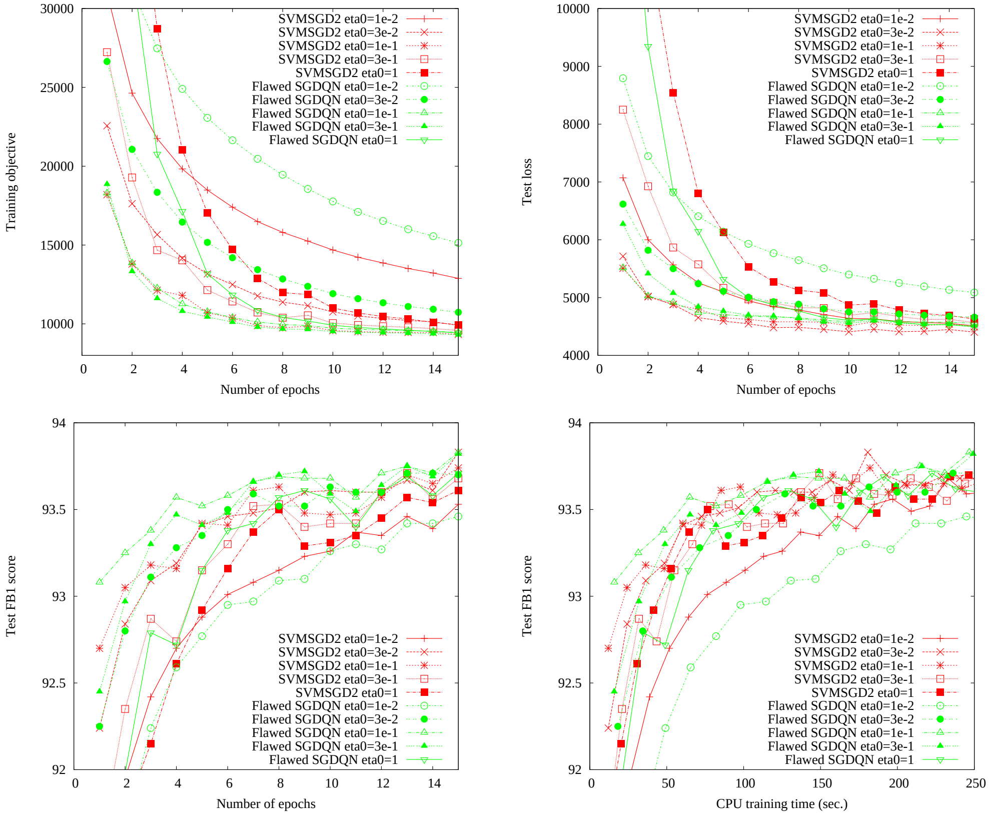
## 5. Correcting SGD-QN

At first glance, correcting SGD-QN simply involves computing the difference $\mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t)$ with $\tau = t+1$ instead of $\tau = t$. In fact, during the weeks preceding the Pascal challenge deadline, we tried both versions and found that picking $\tau = t+1$ performs significantly worse!



RCV1 DATASET

Figure 2: Plots of the training cost and test misclassification error versus the number of epochs for both SVMSGD2 (red) and Flawed SGD-QN (green) running on the RCV1 dataset. Both algorithms reach optimal performance after seeing half the training set.



CRF ON THE CONLL 2000 CHUNKING TASK

Figure 3: Plots of the training cost, test loss, and test F1 score for a CRF trained using both SVMSGD2 (red) and Flawed SGD-QN (green) for various initial rates $\eta_0 = \frac{1}{\lambda t_0}$.

| Flawed SGD-QN | Corrected SGD-QN |
|---|---|

Flawed SGD-QN

```
Require: $\lambda$, $t_0$, $T$, skip
$t \leftarrow 0$, $\mathbf{w} \leftarrow \mathbf{0}$, count $\leftarrow$ skip, $r \leftarrow 2$
$\mathbf{v} \leftarrow \mathbf{0}$, updateB $\leftarrow$ false, $\forall i \; B_{ii} \leftarrow (\lambda)^{-1}$
while $t \leq T$ do
    $\mathbf{w} \leftarrow \mathbf{w} - y_t \, \ell'(y_t \mathbf{w}^\top \mathbf{x}_t) \, (t + t_0)^{-1} \, \mathbf{B} \, \mathbf{x}_t$
    if updateB then
        $\forall i \;\; r_i \leftarrow [\mathbf{g}_t(\mathbf{w}) - \mathbf{g}_t(\mathbf{v})]_i \, / \, [\mathbf{w} - \mathbf{v}]_i$
        $\forall i \;\; r_i \leftarrow \min\{r_i, 100\lambda\}\}$
        $\forall i \;\; B_{ii} \leftarrow B_{ii} + \frac{2}{r}(r_i^{-1} - B_{ii})$
        updateB $\leftarrow$ false, $r \leftarrow r + 1$
    end if

    count $\leftarrow$ count $- 1$
    if count $\leq 0$ then
        count $\leftarrow$ skip, updateB $\leftarrow$ true
        $\mathbf{w} \leftarrow \mathbf{w}$ $-$ skip $\lambda \, (t + t_0)^{-1} \, \mathbf{B} \, \mathbf{w}$
        $\mathbf{v} \leftarrow \mathbf{w}$
    end if

    $t \leftarrow t + 1$
end while
return $\mathbf{w}$
```

Corrected SGD-QN

```
Require: $\lambda$, $t_0$, $T$, skip
$t \leftarrow 0$, $\mathbf{w} \leftarrow \mathbf{0}$, count $\leftarrow$ skip
$\mathbf{v} \leftarrow \mathbf{0}$, updateB $\leftarrow$ false, $\forall i \; B_{ii} \leftarrow (\lambda t_0)^{-1}$
while $t \leq T$ do

    if updateB then
        $\forall i \;\; r_i \leftarrow [\mathbf{g}_t(\mathbf{w}) - \mathbf{g}_t(\mathbf{v})]_i \, / \, [\mathbf{w} - \mathbf{v}]_i$
        $\forall i \;\; r_i \leftarrow \max\{\lambda, \min\{100\lambda, r_i\}\}$
        $\forall i \;\; B_{ii} \leftarrow B_{ii}(1 +$ skip $B_{ii} \, r_i)^{-1}$
        updateB $\leftarrow$ false
    end if
    $z \leftarrow y_t \mathbf{w}^\top \mathbf{x}_t$
    count $\leftarrow$ count $- 1$
    if count $\leq 0$ then
        count $\leftarrow$ skip, updateB $\leftarrow$ true
        $\mathbf{v} \leftarrow \mathbf{w}$
        $\mathbf{w} \leftarrow \mathbf{w}$ $-$ skip $\lambda \, \mathbf{B} \, \mathbf{w}$
    end if
    $\mathbf{w} \leftarrow \mathbf{w} - y_t \, \ell'(z) \, \mathbf{B} \, \mathbf{x}_t$
    $t \leftarrow t + 1$
end while
return $\mathbf{w}$
```

Figure 4: Pseudo-codes for the Flawed SGD-QN and Corrected SGD-QN algorithms. The main changes have been colored: each color stands for a particular change.

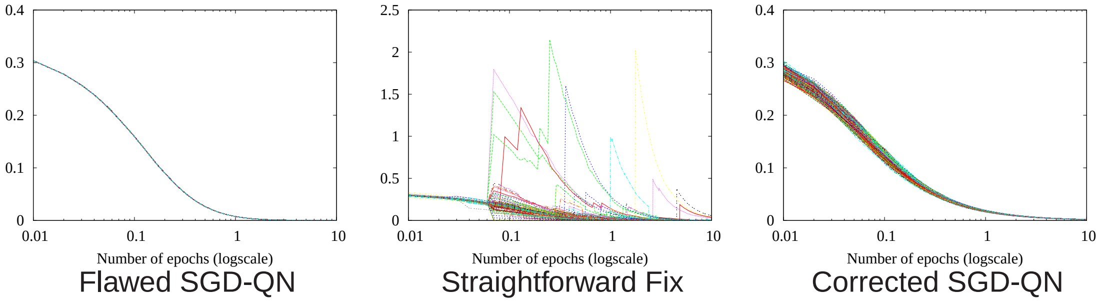
### 5.1 The Failure of the Straightforward Fix

When experimenting with the oLBFGS algorithm (Schraudolph et al., 2007), we observed and reported that setting the global learning gain was very difficult. We encounter the same difficulty when we modify the SGD-QN algorithm to use $\tau = t + 1$.

In order to form an intuition about the learning rate, we must pay attention to its influence on the stochastic noise. Stochastic gradient descent with a constant learning rate generates a cloud of parameter estimates $\mathbf{w}_t$ covering a zone whose extent is defined by the learning rate and by the curvature of the cost function. When the rates decrease with an appropriate speed, this zone shrinks around the solution. Rewriting (2) term-by-term gives

$$[\mathbf{w}_{t+1}]_i \;=\; [\mathbf{w}_t]_i - \eta_{i,t}^{\text{flawQN}} \, [\mathbf{g}_t(\mathbf{w}_t)]_i \quad \text{with } \eta_{i,t}^{\text{flawQN}} = \frac{B_{ii}}{t_0 + t} \,. \tag{7}$$

Since the algorithm periodically adapts $B_{ii}$ on the basis of the observed differences $\mathbf{w}_{t+1} - \mathbf{w}_t$ and $\mathbf{g}_{t+1}(\mathbf{w}_{t+1}) - \mathbf{g}_{t+1}(\mathbf{w}_t)$, the sequence of learning rates $\eta_{i,t}^{\text{flawQN}}$ can occasionally increase. This is confirmed by the middle plot of Figure 5 which displays the evolution of the learning rates $\frac{B_{ii}}{t_0+t}$ for SGD-QN implementing this straightforward fix. Such fluctuations are the source of the difficulty.



Flawed SGD-QN Straightforward Fix Corrected SGD-QN

Figure 5: Plots of the learning rates corresponding to each feature on the course of learning on the Delta dataset. All rates are equal for Flawed SGD-QN (left plot). As explained in Section 5.1, implementing the straightforward fix (middle plot) causes rates to alternatively increase or decrease very fast. The Corrected SGD-QN (right plot) proposes learning rates nicely decreasing at different speeds for each feature.

### 5.2 Managing the Speed of the Learning Rate Decrease

The schedule with which the learning rate decreases during training appears to be a key factor, so we propose to fix SGD-QN by using the second-order information to manage this diminution. Hence, we use learning rates of the form

$$\eta^{\mathsf{corQN}}_{i,t} = \left(\lambda t_0 + \sum_{k=1}^{t-1} r_{i,k}\right)^{-1} \qquad \text{where } \; r_{i,t} = \frac{\left[\mathbf{g}_{t+1}(\mathbf{w}_{t+1}) - \mathbf{g}_{t+1}(\mathbf{w}_t)\right]_i}{\left[\mathbf{w}_{t+1} - \mathbf{w}_t\right]_i}\,. \tag{8}$$

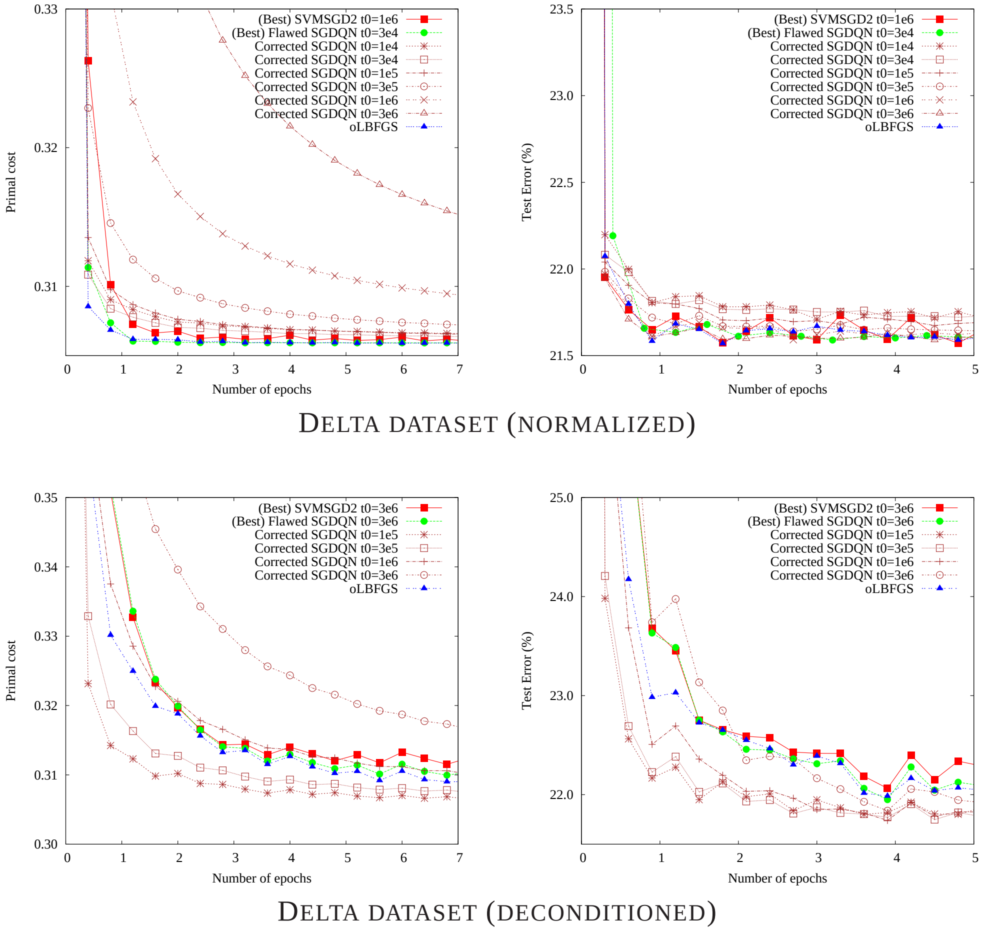
When $t$ becomes large, we recover an expression comparable to the original formulation (7), $\eta^{\mathsf{corQN}}_{i,t} = \frac{\bar{r}_i^{\,-1}}{\lambda t_0 \bar{r}_i^{\,-1} + t} + \mathrm{o}\left(\frac{1}{t}\right)$, where $\bar{r}_i$ denotes the average value of the ratios $r_{i,t}$ and can be viewed as the coefficient of a diagonal matrix $\mathbf{R}$ such that $\mathbf{R}\,\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right) = \mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t) + \zeta_t + \xi_t$.

It is also interesting to compare the formula $\eta^{\mathsf{corQN}}_{i,t}$ with the first order version $\eta^{\mathsf{SGD}}_{i,t} = \frac{1}{\lambda t_0 + \sum_{k=1}^{t} \lambda}$ which decreases the learning rate after each iteration by adding $\lambda$ to the denominator. Instead of adding a lower bound of the curvature, the proposed learning rate formula adds a stochastic estimate of the curvature.

Interestingly, Equation (8) leads to a convenient recursive formula

$$\eta^{\mathsf{corQN}}_{i,t} = \left(\frac{1}{\eta^{\mathsf{corQN}}_{i,t-1}} + r_{i,t-1}\right)^{-1} = \frac{\eta^{\mathsf{corQN}}_{i,t-1}}{1 \,+\, r_{i,t-1}\,\eta^{\mathsf{corQN}}_{i,t-1}}\,. \tag{9}$$

Figure 4 describes the Corrected SGD-QN algorithm and compares it with a slightly reorganized version of the Flawed SGD-QN algorithm. The diagonal matrix $\mathbf{B}$ is used to store the gains (8). The algorithm schedules a gain update (line 14) whenever it performs a regularization update (line 16). During the next iteration, the algorithm computes $r_{i,t-1}$ (line 6) and implements the learning rate update (9) with an additional multiplicative factor (line 8) because this only happens every `skip` iterations. The effect on the learning rates of using Corrected SGD-QN instead of Flawed SGD-QN is illustrated by Figure 5 if we compare the left and the right plots.



DELTA DATASET (NORMALIZED)

DELTA DATASET (DECONDITIONED)

Figure 6: Plots of the training cost and test misclassification error versus the number of epochs for SVMSGD2 (red) and Flawed SGD-QN (green) with their optimal $t_0$ parameter, and Corrected SGD-QN (brown) running on the Delta dataset. Both *normalized* (top) and *deconditioned* (bottom) cases are considered; see the text for details. All methods can perform roughly identically well on normalized examples but only the Corrected SGD-QN algorithm is able to handle ill-conditioned data.

### 5.3 Performances on Dense Datasets

Figure 6 (top row) compares the performances of Corrected SGD-QN with the best results of Flawed SGD-QN and SVMSGD2 on the Delta dataset. We must recognize that the improvement is minimal. Before running the SGD algorithms, we always precondition the dense datasets by centering all the features, normalizing their variances, and rescaling every example to ensure that $\|\mathbf{x}_k\| = 1$. This operation in fact steals all the improvements SGD-QN can bring. With its adaptive learning rates, the Corrected SGD-QN does not perform worse than the first order SVMSGD2 algorithm. Yet, implementing a strategy involving a single learning rate for all the features appears already very rewarding and, for such cases, the Flawed SGD-QN algorithm is a strong choice because of its capacity to adapt its learning rate.

Corrected SGD-QN should be more efficient for ill-conditioned data. To illustrate this assertion, we created a "deconditioned" version of Delta by applying the usual normalization procedures and then multiplying every tenth feature by twelve. Figure 6 (bottom row) compares the performances of SVMSGD2, Flawed SGD-QN and Corrected SGD-QN on this deconditioned data. The Flawed SGD-QN algorithm clearly suffers from the deconditioning operation because it can not assign a different learning rate per feature. The Corrected SGD-QN works much better. We also verified that the estimated learning rates replicate the deconditioning pattern.

In conclusion, on *dense datasets*, the Corrected SGD-QN bring little improvement over those associated with a *good preconditioning technique*. Preconditioning was probably the main reason of the good SGD-QN results on dense datasets in the Pascal Large Scale Challenge. This does not mean that SGD algorithms cannot be improved. Xu (2010) reports impressive results on Linear SVMs using a well sorted Averaged SGD algorithm (Polyak and Juditsky, 1992).

### 5.4 Performances on Sparse Datasets

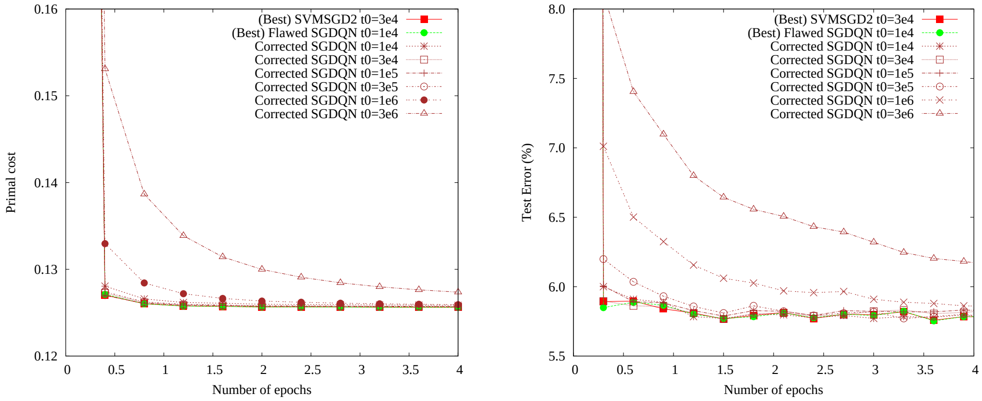
Preconditioning sparse datasets is much more difficult because it is impossible to center sparse features and keep them sparse. In addition, normalizing the variance of very rare features generates a small number of coefficients with high values. This fat tail distribution usually has very negative impact on the test performance. Figure 7 compares the SVMSGD2, Flawed SGD-QN and Corrected SGD-QN algorithms on the Reuters RCV1 dataset, but, as we explained for Figure 2, this task is too easy to draw any conclusions.

Figure 8 then compares the adaptations of SVMSGD2, Flawed SGD-QN (with their best parameters) and Corrected SGD-QN for Conditional Random Fields on the CoNLL 2000 "chunking" task with the setup described in Section 4.2. The Corrected SGD-QN algorithm achieves its optimal test performance after only 75 seconds while SVMSGD2 and Flawed SGD-QN need around twice this time. For comparison, the CRF++ LBFGS optimizer needs 4300 seconds on a slightly faster machine.

## 6. Conclusion

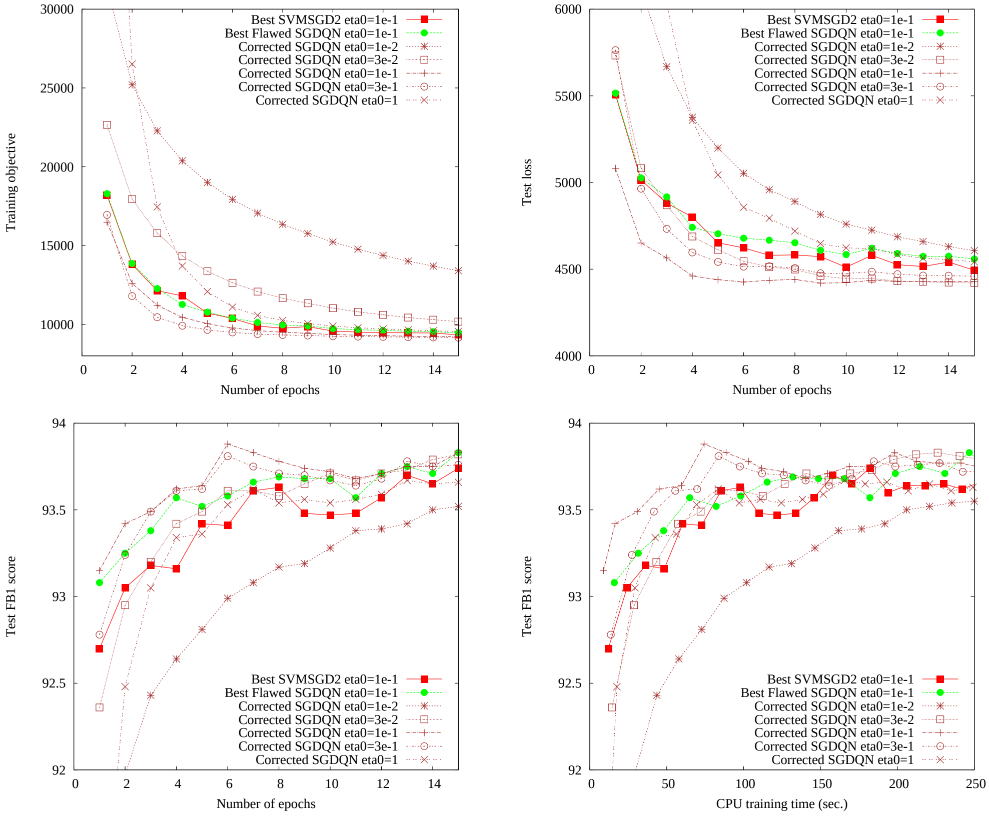
Despite its flaw, the original SGD-QN algorithm works well enough to be a winner of the first PASCAL Large Scale Learning Challenge (Sonnenburg et al., 2008) because it benefits from our careful preconditioning and handles sparse examples efficiently. However, as explained in this document, this original version often does not achieve the full benefits of a diagonal scaling approach.

This paper proposes a correction. Unlike the original SGD-QN algorithm, the Corrected SGD-QN algorithm discovers sensible diagonal scaling coefficients. However, experiments on dense data sets of intermediate dimensionality show that similar speed improvements can be achieved by simple preconditioning techniques such as normalizing the means and the variances of each feature and normalizing the length of each example. On the other hand, normalization is not always an attractive strategy. The Corrected SGD-QN algorithm then becomes interesting because it can adapt automatically to skewed feature distributions (see Section 5.3) or very sparse data (see Section 5.4.)



RCV1 DATASET

Figure 7: Plots of the training cost and test error versus the number of epochs for SVMSGD2 (red) and Flawed SGD-QN (green) with their optimal $t_0$ parameter, and Corrected SGD-QN (brown) running on the RCV1 dataset. All algorithms quickly reach optimal performance.



CRF ON THE CONLL 2000 CHUNKING TASK

Figure 8: Plots of the training cost, test loss, and test F1 score for a CRF trained using the best setups of SVMSGD2 (red) and Flawed SGD-QN (green), and Corrected SGD-QN for various initial rates $\eta_0 = \frac{1}{\lambda t_0}$ (brown). Corrected SGD-QN learns significantly faster.

## References

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, July 2009.

T. Kudo. CRF++: Yet another CRF toolkit, 2007. `http://crfpp.sourceforge.net`.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williams College, Williamstown, 2001. Morgan Kaufmann.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, 1992.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132. Lisbon, Portugal, 2000.

N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In *Proc. 11th Intl. Conf. on Artificial Intelligence and Statistics (AIstats)*, pages 433–440. Soc. for Artificial Intelligence and Statistics, 2007.

S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge. ICML 2008 Workshop, 2008. `http://largescale.first.fraunhofer.de`.

W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. Submitted to JMLR, 2010.