Stochastic Dual Coordinate Ascent

Guillaume DESFORGES & Michaël KARPE & Matthieu ROUX

Projet MALAP 2018 encadré par M. Guillaume OBOZINSKI (Laboratoire IMAGINE)

École des Ponts ParisTech

Introduction

The aim of this project is to highlight the *Stochastic Dual Coordinate Ascent* (SDCA), an optimization algorithm for Machine Learning.

It has been proven that solving the dual problem is very effective for *Support Vector Machines* (SVM). In this poster, we focused mainly on using this algorithm for *Logistic Regression* (LR).

This project summarizes a few necessary keys of knowledge in order to understand the dual problem, before discussing the algorithm and its possible implementation. Finally, we tested our own implementation against an existing implementation of LR that solves the primal problem on multiple datasets.

Illustration of Stochastic Gradient Descent

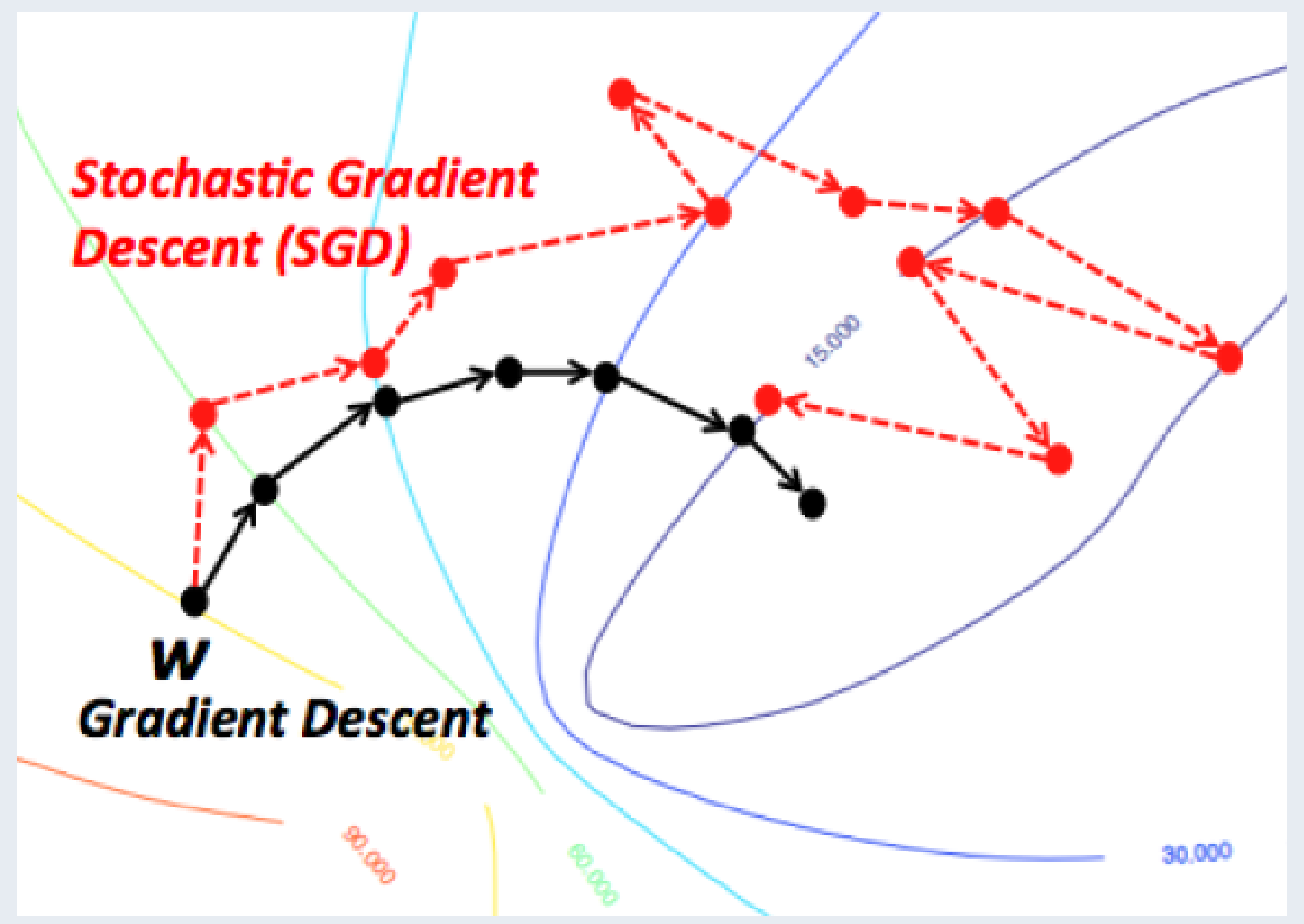

FIGURE – Illustration of Stochastic Gradient Descent.

Logistic regression example

In order to understand fully the method behind the first paper, let's take an example with the logistic regression. We will consider logistic regression only for binary classification. We use the following usual notations : $X \in \mathbf{X} = \mathbb{R}^p$ the random variable for the description space, and $Y \in \mathbf{Y} = \{-1, 1\}$ the random variable for the label. We recall that the model is the following :

$$\frac{\mathbb{P}(y = 1|X = x)}{\mathbb{P}(y = -1|X = x)} = w^T x, \quad w \in \mathbb{R}^p$$

Formulation of dual problem

We want to find w such that it maximizes the likelihood, or log-likelihood, with a term of regularization :

$$\min_w C \sum_i \log\left(1 + e^{-y_i w^T x_i}\right) + \frac{1}{2} w^T w$$

In order to get the dual problem, we rewrite it with an artificial constraint $z_i = e^{-y_i w^T x_i}$, and we have the following lagrangian :

$$\mathcal{L}(w, z, \alpha) = \sum_i (C \log(1 + z_i) + \alpha_i z_i) - \sum_i \alpha_i e^{-y_i w^T x_i} + \frac{1}{2} w^T w$$

We will note $w^* = \sum_i \alpha_i y_i x_i$ and z^* the variables solution of the optimization problem

$$\min_{w,z} \mathcal{L}(w, z, \alpha) = \mathcal{L}(w^*, z^*, \alpha) = \psi(\alpha)$$

In fact, it leads to the following dual problem :

$$\max_\alpha \sum_{i \in I} (-\alpha_i \log(\alpha_i) - (C - \alpha_i)\log(C - \alpha_i)) - \frac{1}{2}\alpha^T Q \alpha$$
$$s.t. \quad I = \{i \mid \; 0 < \alpha_i < C\}, \quad 0 \le \alpha_i \le C$$

We used the notation $Q = (Q_{ij})_{i,j}$ where $Q_{ij} = y_i x_i^T x_j y_j$

SDCA optimization problem

Let $x_1, \dots, x_n \in \mathbb{R}^d$, $\phi_1, \dots, \phi_n$ scalar convex functions, $\lambda > 0$ regularization parameter. Let focus on the following optimization problem :

$$\min_{w \in \mathbb{R}^d} P(w) = \left[\frac{1}{n}\sum_{i=1}^n \phi_i(w^\top x_i) + \frac{\lambda}{2}\|w\|^2\right] \tag{1}$$

with solution $w^* = \arg\min_{w \in \mathbb{R}^d} P(w)$.

Moreover, we say that a solution w is ϵ_P-sub-optimal if $P(w) - P(w^*) \le \epsilon_P$. We analyze here the required runtime to find an ϵ_P-sub-optimal solution using SDCA.

Let $\phi_i^* : \mathbb{R} \to \mathbb{R}$ be the convex conjugate of ϕ_i : $\phi_i^*(u) = \max_z(zu - \phi_i(z))$. The dual problem of (1) is defined as follows :

$$\max_{\alpha \in \mathbb{R}^n} D(\alpha) = \left[\frac{1}{n}\sum_{i=1}^n -\phi_i^*(-\alpha_i) - \frac{\lambda}{2}\left\|\frac{1}{\lambda n}\sum_{i=1}^n \alpha_i x_i\right\|^2\right] \tag{2}$$

with solution $\alpha^* = \arg\max_{a \in \mathbb{R}^n} D(\alpha)$.

If we define $w(\alpha) = \frac{1}{\lambda n}\sum_{i=1}^n \alpha_i x_i$, then we have $w(\alpha^*) = w^*$, $P(w^*) = D(\alpha^*)$, $\forall(w, \alpha), P(w) = D(\alpha)$ due to classic optimization results, and the duality gap $P(w(\alpha)) - P(w^*)$.

The SDCA algorithm is described further. T_0 can be chosen between 1 to T, and is generally chosen equal to $T/2$. However, in pratice, these parameters are not required as the duality gap is used to terminate the algorithm.

Losses used

Squared loss :

$$\phi_i(a) = (a - y_i)^2, \quad \phi_i^*(-a) = -a y_i + a^2/4$$
$$\Delta\alpha_i = \frac{y_i - x_i^\top w^{(t-1)} - 0.5\alpha_i^{(t-1)}}{0.5 + \|x_i\|^2/(\lambda n))}$$

Absolute deviation loss :

$$\phi_i(a) = |a - y_i|, \quad \phi_i^*(-a) = -a y_i,\ a \in [-1, 1]$$
$$\Delta\alpha_i = \max\left(1, \min\left(1, \frac{y_i - x_i^\top w^{(t-1)}}{\|x_i\|^2/(\lambda n)} + \alpha_i^{(t-1)}\right)\right) - \alpha_i^{(t-1)}$$

Log loss :

$$\phi_i(a) = \log(1 + \exp(-y_i a))$$
$$\phi_i^*(-a) = -a y_i \log(a y_i) + (1 - a y_i)\log(1 - a y_i)$$
$$\Delta\alpha_i = \frac{(1 + \exp(x_i^\top w^{(t-1)} y_i))^{-1} y_i - \alpha_i^{(t-1)}}{\max(1, 0.25 + \|x_i\|^2/(\lambda n))}$$

(γ-smoothed) Hinge loss :

$$\phi_i(a) = \max\{0, 1 - y_i a\}$$
$$\phi_i^*(-a) = -a y_i + \gamma a^2/2,\ a y_i \in [0, 1]$$
$$\Delta\alpha_i = y_i \max\left(0, \min\left(1, \frac{1 - x_i^\top w^{(t-1)} y_i - \gamma\alpha_i^{(t-1)} y_i}{\|x_i\|^2/(\lambda n) + \gamma} + \alpha_i^{(t-1)} y_i\right)\right) - \alpha_i^{(t-1)}$$

Difference between SGD and SDCA

A simple approach for solving SVM is stochastic gradient descent (SGD). SGD finds an ϵ_P-sub-optimal solution in time $O(1/(\lambda\epsilon_P))$. This runtime does not depend on n and therefore is favorable when n is very large. However, the SGD approach has several disadvantages :

1. It does not have a clear stopping criterion
2. It tends to be too aggressive at the beginning of the optimization process, especially when λ is very small
3. While SGD reaches a moderate accuracy quite fast, its convergence becomes rather slow when we are interested in more accurate solutions

SDCA Procedure algorithm

Algorithm 1 Procedure SCDA

```
procedure SCDA($\alpha^{(0)}, \phi, T_0, T$)
    $w^{(0)} \leftarrow w(\alpha^{(0)})$
    for $t = 1, \dots, T$ do
        Randomly pick $i$
        Find $\Delta\alpha_i$ to increase dual (*)
        $\alpha^{(t)} \leftarrow \alpha^{(t-1)} + \Delta\alpha_i e_i$
        $w^{(t)} \leftarrow w^{(t-1)} + (\lambda n)^{-1}\Delta\alpha_i x_i$
    if Averaging option then
        return $\overline{w} = \frac{1}{T-T_0}\sum_{i=T_0+1}^{T} w^{(t-1)}$
    if Random option then
        return $\overline{w} = w^{(t)}$ for a random $t \in [|T_0 + 1, T|]$
```

(*) corresponds to the following operation :

$$\Delta\alpha_i \leftarrow \arg\max_{\Delta\alpha_i} -\phi_i^*(-(\alpha_i^{(t-1)} + \Delta\alpha_i)) - \frac{\lambda n}{2}\left|\left|w^{(t-1)} + \frac{\Delta\alpha_i x_i}{\lambda n}\right|\right|^2$$

Choose of output option

Because of the stochastic behavior of the algorithm, the output is very sensitive to the iteration at which it stops. Indeed, coefficients vary suddenly, and the convergence is not really monotonous : at some point, it is uncertain whether the loss improves or not. There are essentially two ways of taking this into account. We can first stop at a random step. It actually has good results. The second method consists in averaging the last $\alpha^{(t)}$ obtained by the algorithm, making sure that the local variations of α are corrected.

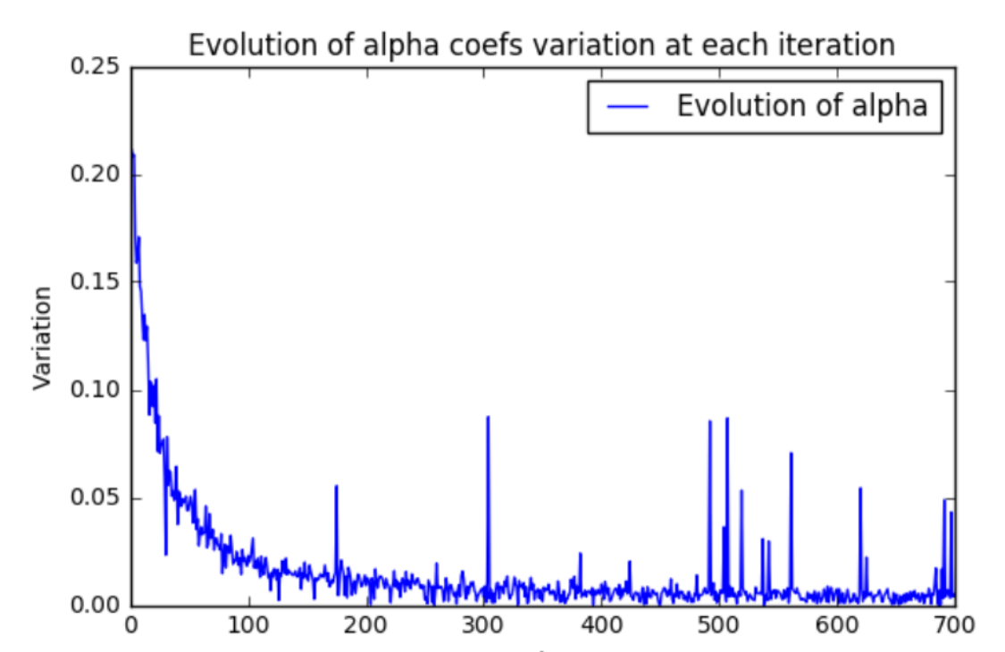

FIGURE – Illustration of the necessity to choose carefully the output.

Study framework

In this study, SDCA is computed either for L-Lipschitz loss functions or for $(1/\gamma)$-smooth loss functions. We recall that a function $\phi_i : \mathbb{R} \to \mathbb{R}$ is L-Lipschitz if $\forall a, b \in \mathbb{R}$, $|\phi_i(a) - \phi_i(b)| \le L|a - b|$, and that a function $\phi_i : \mathbb{R} \to \mathbb{R}$ is $(1/\gamma)$-smooth if it is differentiable and its derivative is $(1/\gamma)$-Lipschitz. Moreover, if ϕ_i is $(1/\gamma)$-smooth, then ϕ_i^* is γ-strongly convex. The different loss functions used are described in the corresponding table.

For the sake of simplicity, we consider the following assumptions : $\forall i, \|x_i\| \le 1$, $\forall(i, a), \phi_i(a) \ge 0$ and $\forall i, \phi_i(0) \le 1$. Under these assumptions, we have the following theorem :

Theorem Consider Procedure SDCA with $\alpha^{(0)} = 0$. Assume that $\forall i, \phi_i$ is L-Lipschitz (resp. $(1/\gamma)$-smooth). To obtain an expected duality gap of $\mathbb{E}[P(\overline{w}) - D(\overline{\alpha})] \le \epsilon_P$, it suffices to have a total number of iterations of

$$T \ge n + \max\left(0, \left\lceil n\log\left(\frac{\lambda n}{2L^2}\right)\right\rceil\right) + \frac{20L^2}{\lambda\epsilon_P}$$
$$\left(\text{resp. } T > \left(n + \frac{1}{\lambda\gamma}\right)\log\left[\frac{1}{(T - T_0)\epsilon_P}\left(n + \frac{1}{\lambda\gamma}\right)\right]\right)$$

Implementations

Our implementation is available here :

`https://github.com/GuillaumeDesforges/enpc-malap-project`

We first tested SDCA on logistic regression on a simple dataset.

Results on generated data

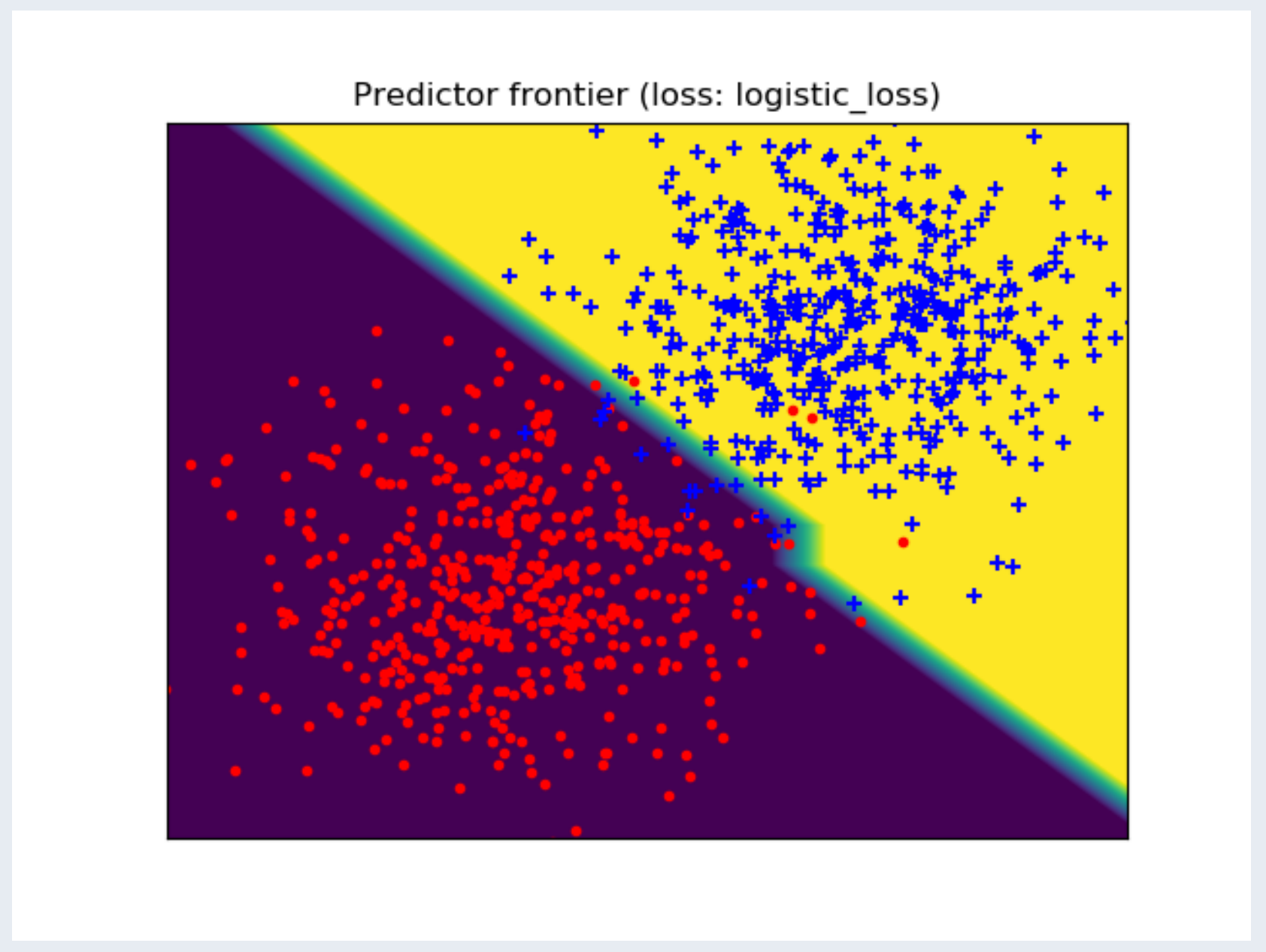

FIGURE – Logistic regression on a simple dataset.

Evolution of error

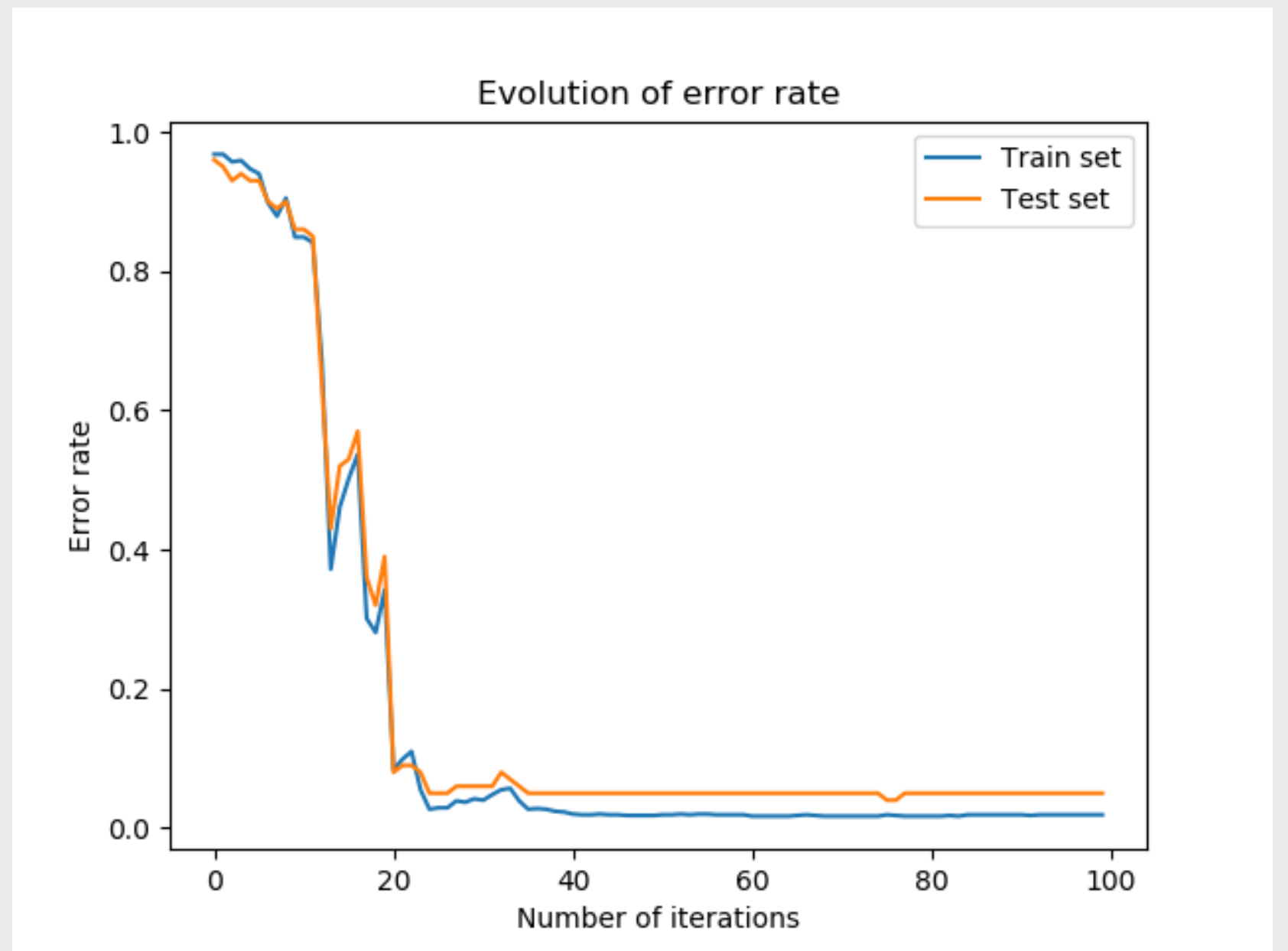

FIGURE – Convergence of SDCA for logistic regression on a simple dataset.

Conclusion

SDCA is an alternative way to compute a predictor for huge data sets with a high number of features. At the time of printing this poster, we did not have the time to present results for the huge data sets we tested, but our first experimentation seems to give fast and accurate results, as SDCA is able to counter issues faced with traditional SGD.

References

[1] Hsiang-Fu Yu & Fang-Lan Huang & Chih-Jen Lin.
Dual coordinate descent methods for logistic regression and maximum entropy models.
Machine Learning Journal.

[2] Shai Shalev-Shwartz & Tong Zhang.
Stochastic dual coordinate ascent methods for regularized loss minimization.
Journal of Machine Learning Research 14, 2013.