# Stochastic Dual Coordinate Descent

Guillaume Desforges & Michaël Karpe & Matthieu Roux

June 14th 2018

## Introduction

In machine learning, the process of fitting a model to the data requires to solve an optimization problem. The difficulty resides in the fact that this optimization quickly becomes very complex when dealing with real problems. The Stochastic Gradient Descent (SGD) is a very popular algorithm to solve those problems because it has good convergence guaranties. Yet, the SGD does not have a good stopping criteria, and its solutions are often not accurate enough.

The Stochastic Dual Coordinate Ascent tries to solves the optimization problem by solving its dual problem. Instead of optimizing the weights, we optimize a dual variable from which we can compute the weights and thus solve the former. This method can give good results for specific problems : for instance, solving the dual problem of the SVM has proven to be effective and to give interesting results, with a linear convergence in some cases.

In this report, we compile the key theoretical points necessary to have a global understanding of the SDCA.

First we introduce the general forms of the machine learning problems we want to use SDCA on and how we get to the dual form. We introduce the duality gap and how it is useful. Then we study computational performances of the method by trying to apply SDCA on concret problems. Finally we conclude on SDCA strengths and weaknesses.

**Note** *We especially added experimentations since the presentation of our poster.*

# 1 Methods

## 1.1 Dual problem

Since a machine learning problem is an optimization problem, one can try to write its dual.

As an example, we present a dual problem for the logistic regression. The multiclass logistic regression, with the cross entropy loss, is a generalization of the binary logistic regression that requires more work, so for now we stick to the binary problem which is interesting enough.

We use the following usual notations :

$X \in \mathbf{X} = \mathbb{R}^p$ the random variable for the description space;

$Y \in \mathbf{Y} = \{-1, 1\}$ the random variable for the label.

We recall that the model is the following :

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

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

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

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

$$\mathcal{L}(w, z, \alpha) = \sum_i (C \log(1 + z_i) + \alpha_i z_i) - \sum_i \alpha_i e^{-z_i} + \frac{1}{2} w^T w \tag{3}$$

We note $w^*$ and $z^*$ the variables solution of the optimization problem

$$\min_{w,z} \mathcal{L}(w, z, \alpha) = \mathcal{L}(w^*, z^*, \alpha) = \psi(\alpha) \tag{4}$$

Where, $w^* = \sum_i \alpha_i y_i x_i$.

It leads to the following dual problem :

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i \in I} (-\alpha_i \log(\alpha_i) - (C - \alpha_i) \log(C - \alpha_i)) - \frac{1}{2} \alpha^T Q \alpha \\ s.t. \quad & I = \{i,\ 0 < \alpha_i < C\} \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{5}$$

We note $Q = (Q_{ij})_{i,j}$ where $Q_{ij} = y_i x_i^T x_j y_j$. It appears that the optimization problem is fully parametrized by the scalar products of the data.

## 1.2 A more generic form

If we consider now a generic loss function $\phi$, either L-Lipschitz or $(1/\gamma)$-smooth, we can rewrite the optimization problem as follow, where P refers to primal form :

$$\min_{w \in \mathbb{R}^d} P(w) \quad \text{where } P(w) = \left[ \frac{1}{n} \sum_{i=1}^{n} \phi_i(w^\top x_i) + \frac{\lambda}{2} \|w\|^2 \right] \tag{6}$$

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

Let's now introduce the convex conjugate, a very useful tool in the dual space. It is defined as follow $\phi_i$ : $\phi_i^*(u) = \max_z(zu - \phi_i(z))$. In the case $\phi_i$ is $(1/\gamma)$-smooth, its convex conjugate $\phi_i^*$ is $\gamma$-strongly convex. We are now able to have a compact notation for the dual problem :

$$\max_{\alpha\in\mathbb{R}^n} D(\alpha) \quad \text{where } D(\alpha) = \left[\frac{1}{n}\sum_{i=1}^{n} -\phi_i^*(-\alpha_i) - \frac{\lambda}{2}\left\|\frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i x_i\right\|^2\right] \tag{7}$$

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

Some loss functions used in the report are described in the table in appendix A.

## 1.3 SDCA principle

When optimizing with the standard SGD, the learning rate $\epsilon$ is not adaptative, and it can be more or less complicated to find a good one, which could provide good properties of convergence.

With DCA, instead of ascending in the direction of the gradient with a fixed increment, it maximizes the objective value on one dual coordinate at once. The idea is then to repeat the operation on all dual coordinate so that it converges to the maximum of the objective function. We can then give it stochastic properties by chosing randomly the coordinate to optimize at each iteration : it is then the SDCA.

## 1.4 Duality Gap

SDCA optimizes the dual problem, which is not the primal problem. This difference is called the duality gap, which we explain here.

Let $x_1, \ldots, x_n \in \mathbb{R}^d$, $\phi_1, \ldots, \phi_n$ scalar convex functions, $\lambda > 0$ regularization parameter.

Let us focus on the following optimization problem:

$$\min_{w\in\mathbb{R}^d} P(w) \quad \text{where } P(w) = \left[\frac{1}{n}\sum_{i=1}^{n}\phi_i(w^\top x_i) + \frac{\lambda}{2}\|w\|^2\right] \tag{8}$$

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

Moreover, we say that a solution $w$ is $\epsilon_P$-sub-optimal if $P(w) - P(w^*) \leq \epsilon_P$.

We analyze here the required runtime to find an $\epsilon_P$-sub-optimal solution using SDCA.

Let $\phi_i^* : \mathbb{R} \to \mathbb{R}$ be the convex conjugate of $\phi_i$ : $\phi_i^*(u) = \max_z(zu - \phi_i(z))$.

The dual problem of (8) is defined as follows:

$$\max_{\alpha\in\mathbb{R}^n} D(\alpha) \quad \text{where } D(\alpha) = \left[\frac{1}{n}\sum_{i=1}^{n} -\phi_i^*(-\alpha_i) - \frac{\lambda}{2}\left\|\frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i x_i\right\|^2\right] \tag{9}$$

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

Let $w(\alpha) = \frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i x_i$. We have $w(\alpha^*) = w^*$, $P(w^*) = D(\alpha^*)$, $\forall(w,\alpha), P(w) = D(\alpha)$ due to classic optimization results.

We define the duality gap : $P(w(\alpha)) - P(w^*)$

## 1.5 The SDCA algorithm

The SDCA algorithm is as follow.

**Algorithm 1** Procedure SCDA with averaging option

**procedure** SCDA($\alpha^{(0)}, \phi, T_0$)
  $w^{(0)} \leftarrow w(\alpha^{(0)})$
  **for** $t = 1, \ldots, T$ **do**
    Randomly pick $i$
    $\Delta\alpha_i \leftarrow \arg\max -\phi_i^*(-(\alpha_i^{(t-1)} + \Delta\alpha_i)) - \frac{\lambda n}{2} \left\| w^{(t-1)} + (\lambda n)^{-1} \Delta\alpha_i x_i \right\|^2$ (*)
    $\alpha^{(t)} \leftarrow \alpha^{(t-1)} + \Delta\alpha_i e_i$
    $w^{(t)} \leftarrow w^{(t-1)} + (\lambda n)^{-1} \Delta\alpha_i x_i$
  **return** $\overline{w} = \frac{1}{T - T_0} \sum_{i=T_0+1}^{T} w^{(t-1)}$

$T_0$ 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.

## 1.6 Stopping time

We consider the following assumptions: $\forall i, \|x_i\| \leq 1$, $\forall (i,a), \phi_i(a) \geq 0$ and $\forall i, \phi_i(0) \leq 1$.
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 $\mathbb{E}[P(\overline{w}) - D(\overline{\alpha})] \leq \epsilon_P$, it suffices to have a total number of iterations of

$$T \geq n + \max\left(0, \left\lceil n \log\left(\frac{\lambda n}{2L^2}\right)\right\rceil\right) + \frac{20L^2}{\lambda \epsilon_P} \quad \left(\text{resp. } T > \left(n + \frac{1}{\lambda\gamma}\right) \log\left[\frac{1}{(T - T_0)\epsilon_P}\left(n + \frac{1}{\lambda\gamma}\right)\right]\right) \quad (10)$$

## 1.7 Why SDCA over SGD ?

SGD finds an $\epsilon_P$-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 $\lambda$ 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.

# 2 Experiments

## 2.1 Implementation

The experiments in this report were done with our own implementation, available on GitHub :
`https://github.com/GuillaumeDesforges/enpc-malap-project-sdca`
We implemented :

- Logistic regression estimators that can use an `Optimizer` object
- SGD optimizer
- SDCA optimizer
- projections (polynomial and gaussian)
- some data utilities

## 2.2 Datasets

We used our implementation on :

- Heart Deseases : `https://archive.ics.uci.edu/ml/datasets/Arrhythmia`

## 2.3 Protocol

We took the raw datasets as input and fit an estimator thanks to either SGD or SDCA optimizer, and compared the primal objective function during learning.
The hyperparameters were fixed to standart values.

## 2.4 Results

## Conclusion

## References

# A Losses used

| |
|---|
| **Squared loss:** |
| $$\phi_i(a) = (a - y_i)^2$$ |
| $$\phi_i^*(-a) = -ay_i + a^2/4$$ |
| $$\Delta\alpha_i = \frac{y_i - x_i^\top w^{(t-1)} - 0.5\alpha_i^{(t-1)}}{0.5 + \|x_i\|^2/(\lambda n))}$$ |
| **Absolute deviation loss:** |
| $$\phi_i(a) = \|a - y_i\|$$ |
| $$\phi_i^*(-a) = -ay_i,\ a \in [-1, 1]$$ |
| $$\Delta\alpha_i = \max\left(1, \min\left(1, \frac{y_i - x_i^\top w^{(t-1)}}{\|x_i\|^2/(\lambda n)} + \alpha_i^{(t-1)}\right)\right) - \alpha_i^{(t-1)}$$ |
| **Log loss:** |
| $$\phi_i(a) = \log(1 + \exp(-y_i a))$$ |
| $$\phi_i^*(-a) = -ay_i \log(ay_i) + (1 - ay_i)\log(1 - ay_i)$$ |
| $$\Delta\alpha_i = \frac{(1 + \exp(x_i^\top w^{(t-1)} y_i))^{-1} y_i - \alpha_i^{(t-1)}}{\max(1, 0.25 + \|x_i\|^2/(\lambda n))}$$ |
| **($\gamma$-smoothed) Hinge loss:** |
| $$\phi_i(a) = \max\{0, 1 - y_i a\}$$ $$\phi_i^*(-a) = -ay_i + \gamma a^2/2,\ ay_i \in [0, 1]$$ |
| $$\Delta\alpha_i = y_i \max\left(0, \min\left(1, \frac{1 - x_i^\top w^{(t-1)} y_i - \gamma\alpha_i^{(t-1)} y_i}{\|x_i\|^2/(\lambda n) + \gamma} + \alpha_i^{(t-1)} y_i\right)\right) - \alpha_i^{(t-1)}$$ |

Table 1: Used loss functions, convex conjugates and closed form of solutions of problem (*).

$\Delta\alpha_i$ is the notation we use to represent the increment to add to $\alpha_i$ (one coordinate, at a given iteration) to maximize the objective function with respect to that coordinate.