Stochastic Optimization and Automatic Differentiation for Machine Learning

SDCA and PEGASOS

Zakarya Ali

16 mai 2018

Introduction

I decided to implement multiple variants of the **SDCA (Stochastic Dual Coordinate Ascent)** algorithm (from the article Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization by Shai Shalev-Shwartz and Tong Zhang to estimate Support Vector Machines. First, I apply the algorithm on randomly generated data, then I use a credit fraud dataset to compare **SDCA** and **PEGASOS (Primal Estimated subGrAdient SOlver for SVM)**, a sub-gradient descent approach extracted from "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM" an article published by Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro and Andrew Cotter.

The code is available on `https://github.com/zakaryaxali/sdca-credit_fraud`

1 The algorithms

As a quick reminder, the Support Vector Machine is a model which follows this these notions :

- looking for the optimal hyperplane able to divide 2 classes by maximizing the distance between the closest points of these classes (see figure 1). The points on the border are called support vectors.
- If some point are in the wrong side, we adjust their weight to reduce their influence.
- When we don't find a linear separator, we project the points into higher space dimensions where they become linearly separable (kernel trick).

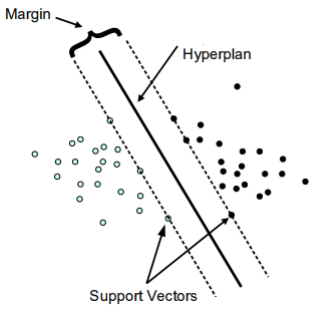


FIGURE 1: SVM principle

Optimizing the SVM model means minimizing the following penalized application :

$$P(\omega) = \frac{1}{n}\sum_{i=1}^{n}\phi_i(\omega^T X_i) + \frac{\lambda}{2}||\omega||^2$$

where :

- $X_1, .., X_n \in \mathbb{R}^d$ are the explanatory variables for the n observations
- λ is the SDCA regularization term
- $w \in \mathbb{R}$ is the parameter to optimize
- $\phi_i(a) = max(0, 1 - y_i a)$ the Hinge loss linked to the SVM
- $y_1, .., y_n \in \{-1, 1\}$ are the associated labels

1.1 SDCA

Figure 2 presents SDCA algorithm.

Procedure SDCA$(\alpha^{(0)})$

Let $w^{(0)} = w(\alpha^{(0)})$
Iterate: for $t = 1, 2, \ldots, T$:
  Randomly pick i
  Find $\Delta\alpha_i$ to maximize $-\phi_i^*(-(\alpha_i^{(t-1)} + \Delta\alpha_i)) - \frac{\lambda n}{2}\|w^{(t-1)} + (\lambda n)^{-1}\Delta\alpha_i x_i\|^2$
  $\alpha^{(t)} \leftarrow \alpha^{(t-1)} + \Delta\alpha_i e_i$
  $w^{(t)} \leftarrow w^{(t-1)} + (\lambda n)^{-1}\Delta\alpha_i x_i$
Output (Averaging option):
  Let $\bar{\alpha} = \frac{1}{T-T_0}\sum_{i=T_0+1}^{T}\alpha^{(t-1)}$
  Let $\bar{w} = w(\bar{\alpha}) = \frac{1}{T-T_0}\sum_{i=T_0+1}^{T} w^{(t-1)}$
  return $\bar{w}$
Output (Random option):
  Let $\bar{\alpha} = \alpha^{(t)}$ and $\bar{w} = w^{(t)}$ for some random $t \in T_0 + 1, \ldots, T$
  return $\bar{w}$

FIGURE 2: SDCA algorithm

For each step, we compute $\Delta\alpha_i = max(-1, min(\frac{y_i - x_i^T\omega^{t-1}}{||x_i||^2/(\lambda * n)} + \alpha_i^{t-1})) - \alpha_i^{t-1}$ (since we solve a Hinge loss SVM). Here we implement three variants of SDCA : SDCA random, permutation (see figure 3) and cyclic. From the reference article, it is a good choice to use $T_0 = \frac{T}{2}$, so we

Procedure SDCA-Perm$(\alpha^{(0)})$

Let $w^{(0)} = w(\alpha^{(0)})$
Let $t = 0$
Iterate: for epoch $k = 1, 2, \ldots$
  Let $\{i_1, \ldots, i_n\}$ be a random permutation of $\{1, \ldots, n\}$
  Iterate: for $j = 1, 2, \ldots, n$:
    $t \leftarrow t + 1$
    $i = i_j$
    Find $\Delta\alpha_i$ to increase dual (*)
    $\alpha^{(t)} \leftarrow \alpha^{(t-1)} + \Delta\alpha_i e_i$
    $w^{(t)} \leftarrow w^{(t-1)} + (\lambda n)^{-1}\Delta\alpha_i x_i$
Output (Averaging option):
  Let $\bar{\alpha} = \frac{1}{T-T_0}\sum_{i=T_0+1}^{T}\alpha^{(t-1)}$
  Let $\bar{w} = w(\bar{\alpha}) = \frac{1}{T-T_0}\sum_{i=T_0+1}^{T} w^{(t-1)}$
  return $\bar{w}$
Output (Random option):
  Let $\bar{\alpha} = \alpha^{(t)}$ and $\bar{w} = w^{(t)}$ for some random $t \in T_0 + 1, \ldots, T$
  return $\bar{w}$

FIGURE 3: SDCA Permutation algorithm

will use it during our implementation.

1.2 ASDCA

Figure 4 show Accelerated Mini-Batch Stochastic Dual Coordinate Ascent (ASDCA) algorithm from Accelerated Mini-Batch Stochastic Dual Coordinate Ascent article by Shai Shalev-Shwartz and Tong Zhang.

Procedure Accelerated Mini-Batch SDCA

Parameters scalars λ, γ and $\theta \in [0, 1]$; mini-batch size m
Initialize $\alpha_1^{(0)} = \cdots = \alpha_n^{(0)} = \bar{\alpha}^{(t)} = 0, x^{(0)} = 0$
Iterate: for $t = 1, 2, \ldots$
$u^{(t-1)} = (1 - \theta)x^{(t-1)} + \theta \nabla g^*(\bar{\alpha}^{(t-1)})$
Randomly pick subset $I \subset \{1, \ldots, n\}$ of size m and update the dual variables in I
$\alpha_i^{(t)} = (1 - \theta)\alpha_i^{(t-1)} - \theta \nabla \phi_i(u^{(t-1)})$ for $i \in I$
$\alpha_j^{(t)} = \alpha_j^{(t-1)}$ for $j \notin I$
$\bar{\alpha}^{(t)} = \bar{\alpha}^{(t-1)} + n^{-1} \sum_{i \in I} (\alpha_i^{(t)} - \alpha_i^{(t-1)})$
$x^{(t)} = (1 - \theta)x^{(t-1)} + \theta \nabla g^*(\bar{\alpha}^{(t)})$
end

FIGURE 4: ASDCA algorithm

For this algorithm, we need to implement the optimization procedure using the smooth version of the hinge loss (see notebook).

1.3 PEGASOS

PEGASOS (figure 5) is an algorithm with stochastic gradient descent on primal problem.

As cited in the pseudo-code, this algorithm has an optional gradient projection approach. Also, there is a mini-batch variant to PEGASOS (see figure 6).

2 Application

2.1 Randomly generated dataset

We generate randomly 1000 observations with 100 features for binary labels. Implementing previous algorithms on this dataset we get the results from figure 7 and table 1.

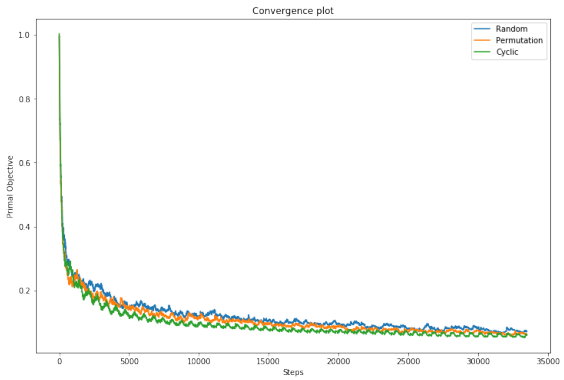
We observe similar results for the variants of a same algorithm. However, the overall comparison shows slower convergence speed for PEGASOS, lower accuracy for ASDCA and best speed and accuracy for mini-batch PEGASOS. Even though SDCA Permutation accuracy and average accuracy are the best, they are a bit slower to achieve.

INPUT: S, λ, T
INITIALIZE: Set $\mathbf{w}_1 = 0$
FOR $t = 1, 2, \ldots, T$
Choose $i_t \in \{1, \ldots, |S|\}$ uniformly at random.
Set $\eta_t = \frac{1}{\lambda t}$
If $y_{i_t} \langle \mathbf{w}_t, \mathbf{x}_{i_t} \rangle < 1$, then:
Set $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t + \eta_t y_{i_t} \mathbf{x}_{i_t}$
Else (if $y_{i_t} \langle \mathbf{w}_t, \mathbf{x}_{i_t} \rangle \geq 1$):
Set $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t$
[Optional: $\mathbf{w}_{t+1} \leftarrow \min \left\{ 1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+1}\|} \right\} \mathbf{w}_{t+1}$]
OUTPUT: $\mathbf{w}_{T+1}$

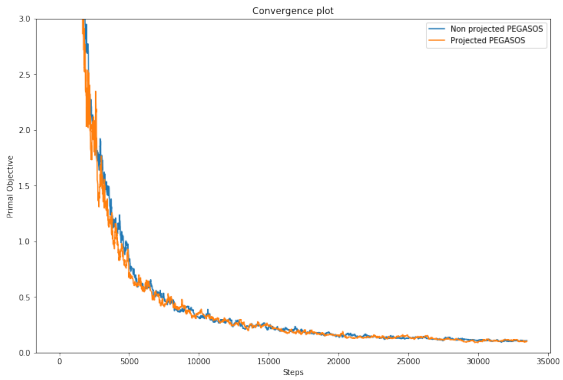
FIGURE 5: PEGASOS algorithm

INPUT: S, λ, T, k
INITIALIZE: Set $\mathbf{w}_1 = 0$
FOR $t = 1, 2, \ldots, T$
Choose $A_t \subseteq [m]$, where $|A_t| = k$, uniformly at random
Set $A_t^+ = \{i \in A_t : y_i \langle \mathbf{w}_t, \mathbf{x}_i \rangle < 1\}$
Set $\eta_t = \frac{1}{\lambda t}$
Set $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t + \frac{\eta_t}{k} \sum_{i \in A_t^+} y_i \mathbf{x}_i$
[Optional: $\mathbf{w}_{t+1} \leftarrow \min \left\{ 1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+1}\|} \right\} \mathbf{w}_{t+1}$]
OUTPUT: $\mathbf{w}_{T+1}$

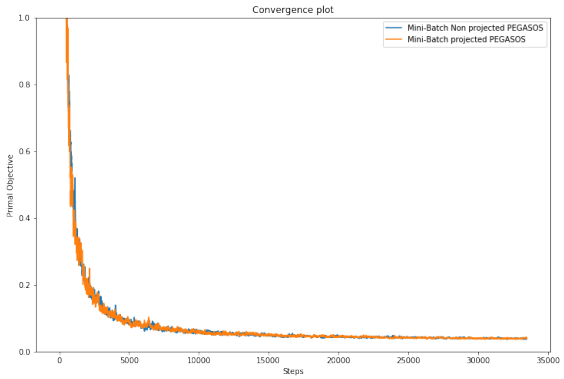
FIGURE 6: Mini-batch PEGASOS algorithm



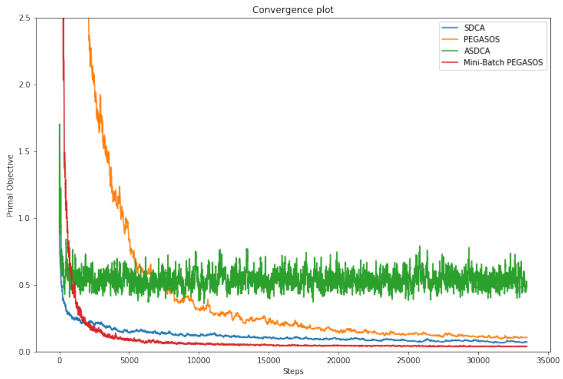
(a) SDCA



(b) PEGASOS



(c) Mini-batch PEGASOS



(d) Overall

FIGURE 7: Convergence of the different algorithms for simulated data

Algorithm	Accuracy	Averaged Accuracy
SDCA Random	0.912	0.918
SDCA Permutation	0.93	0.933
SDCA Cyclic	0.915	0.924
ASDCA	0.881	0.903
Non projected PEGASOS	0.915	0.918
Projected PEGASOS	0.903	0.912
Non projected mini-batch PEGASOS	0.921	0.918
Projected mini-batch PEGASOS	0.912	0.924
Always True	0.945	

TABLE 1: Algorithm Accuracies for simulated data

2.2 Credit fraud dataset

This is a Kaggle dataset on credit fraud transactions. It is higly unbalanced so we had to, prior to apply our algorithms, treat the train set accordingly. This dataset has 30 features.

Figure 8 and table 2 present the results.

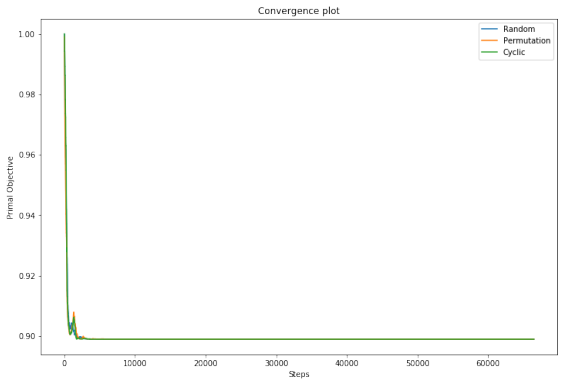
Algorithm	Accuracy
SDCA Random	0.648
ASDCA	0.646
Non projected PEGASOS	0.65
Non projected mini-batch PEGASOS	0.649
Always predict there is no fraud	0.998

TABLE 2: Algorithm Accuracies for credit card fraud data

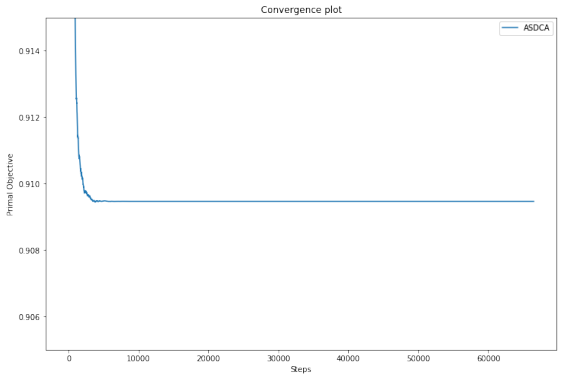
Due to the lack of balance of this dataset, accuracy of all the algorithms is quite low. However, similarly to the random dataset, we notice no real differences between variations of a same algorithm. Again, my implementation of the SDCA looks less performant than the other alorithms. Here again, Mini-batch PEGASOS is the best algorithm as it converges faster.

Conclusion

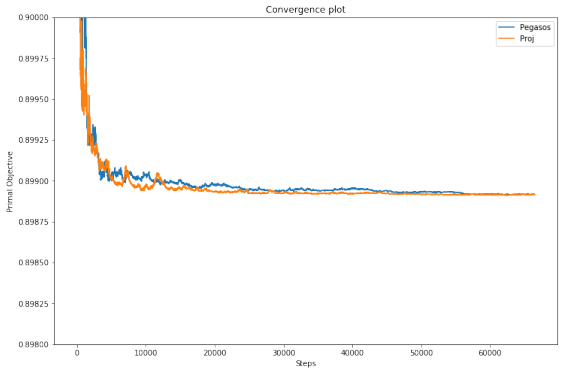
Both main algorithms have slightly the same performances in terms of accuracy for both datasets. Although PEGASOS seem to converge faster, we could have look for the influence of the regularization parameter λ. Mini-batch PEGASOS looks like the best overall implementation for linear SVM. A batch size analysis may be a good path to follow in order to finetune the algorithm and find better results.



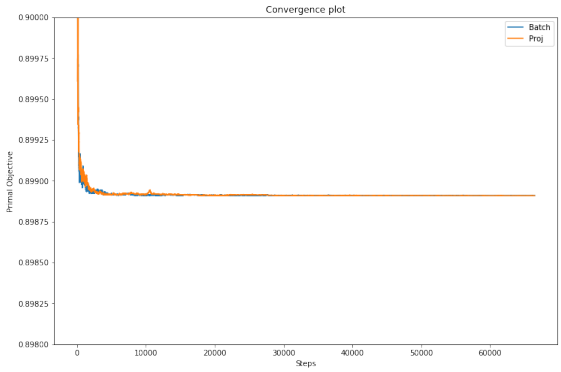
(a) SDCA



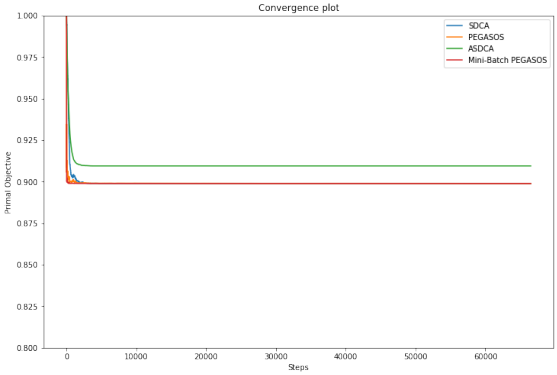
(b) ASDCA



(c) PEGASOS



(d) Mini-batch PEGASOS



(e) Overall

FIGURE 8: Convergence of the different algorithms for credit fraud data