Parallel coordinate descent for the Adaboost problem

Olivier Fercoq, olivier.fercoq@ed.ac.uk

THE UNIVERSITY *of* EDINBURGH

Introduction

The Adaboost algorithm [2] is a widely used classification algorithm. Its goal is to combine many weak hypotheses with high error rate to generate a single strong hypothesis with very low error.
We propose a new parallel version of Adaboost based on recent work on parallel coordinate descent [3]. Our results are:

- The logarithm of the exponential loss is Nesterov separable which implies the existence of an efficient separable overapproximations (Theorem 1).
- The parallel coordinate descent method on the Adaboost problem converges as $O(1/t)$ (Theorem 2) and we give its theoretical parallelisation speedup.
- We provide numerical examples and compare our algorithm with other approaches on a large scale learning problem.

The Adaboost problem

Let $M \in \mathbb{R}^{m\times n}$ be a matrix of features, $y \in \mathbb{R}^m$ be a vector of labels and $A_{j,i} = y_j M_{j,i}$.
The Adaboost problem is the minimisation of the exponential loss:

$$\inf_{\lambda\in\mathbb{R}^n} \frac{1}{m}\sum_{j=1}^{m} \exp((A\lambda)_j) = \inf_{\lambda\in\mathbb{R}^m} f(A\lambda) = \bar{f}_A,$$

where $f(x) = \frac{1}{m}\sum_{j=1}^m \exp(x_j)$.
We will also consider the following equivalent objective function with Lipschitz gradient

$$F(\lambda) = \log(f(A\lambda)),$$

and its associated $C^{1,1}$ Adaboost problem

$$\inf_{\lambda\in\mathbb{R}^n} F(\lambda).$$

Classically [2], this problem is solved by greedy coordinate descent. At each iteration, one selects the classifier with the largest error and updates its weight in order to decrease at most this error.
We propose here a randomised parallel coordinate descent method to solve this optimisation problem.

Separable Overapproximation

Let ω be the maximum number of element in a row of matrix A, that is

$$\omega = \max_{1\le j\le m} |\{i \in \{1,\dots,n\} \,:\, A_{i,j} \neq 0\}|.$$

For $\tau \le n$, let us denote

$$p_l = \frac{\binom{\omega}{l}\binom{n-\omega}{\tau-l}}{\binom{n}{\tau}},\ c_l = \max\left(\frac{l}{\omega}, \frac{\tau-l}{n-\omega}\right)$$

$$\beta = \sum_{k=1}^{\min(\omega,\tau)} \min\left(1, \frac{mn}{\tau}\sum_{l=k}^{\min(\omega,\tau)} c_l p_l\right).$$

Definition ([4]). A τ-nice sampling $\hat{S}$ is a random choice of coordinates such that $\forall S \subseteq \{1,\dots,n\}$,

$$\mathbf{P}(\hat{S} = S) = \begin{cases} \frac{1}{\binom{n}{\tau}}, & \text{if } |S| = \tau \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 1. *The function F has a coordinate-wise Lipschitz gradient with constants*

$$L_i = \max_{1\le j\le m} A_{j,i}^2, \quad 1 \le i \le n.$$

Moreover, if $\hat{S}$ is a τ-nice sampling, then

$$\mathbf{E}[F(\lambda + \delta_{[\hat{S}]})] \le F(\lambda) + \frac{\tau}{n}\left(\langle \nabla F(\lambda), \delta\rangle + \beta \|\delta\|_L^2\right).$$

Parallel Adaboost algorithm

Compute β and $(L_i)_{1\le i\le n}$
for $t \ge 0$ **do**
  Randomly generate S^t following sampling $\hat{S}$
  for $i \in S^t$ **do in parallel**
    $\delta_i \leftarrow \frac{1}{\beta L_i}\nabla_i F(\lambda^t)$
    $\lambda_i^{t+1} \leftarrow \lambda_i^t + \delta_i$
  end for
  if $F(\lambda^{t+1}) > F(\lambda^t)$ **then**
    $\lambda^{t+1} \leftarrow \lambda^t$
  end if
end for

Convergence

Theorem 2. *For an initial point $\lambda^0 \in \mathbb{R}^n$, accuracy $0 < \varepsilon < 2\bar{f}_A$ and confidence level $\rho > 0$, if*

$$t \ge \frac{4\beta n}{\tau}\frac{(1+2\tilde{w}/\tilde{c})^2}{\tilde{\gamma}}\frac{f(A\lambda^0)^2}{\bar{f}_A}\frac{1}{\varepsilon}(1+\log\frac{1}{\rho}) + 2,$$

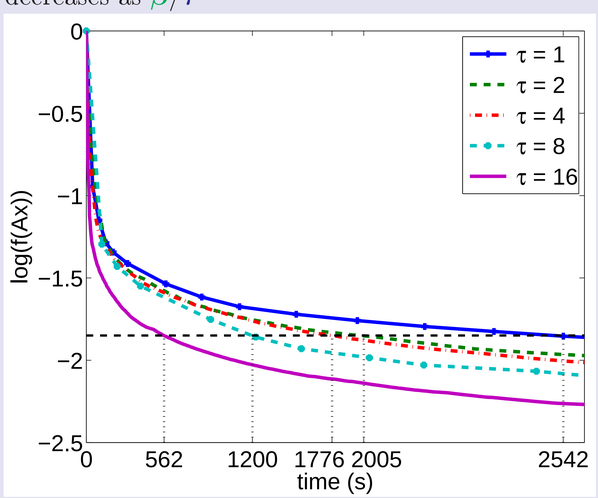
then $\mathbf{P}(f(A\lambda^t) - \bar{f}_A \le \varepsilon) \ge 1-\rho$.

The parameters $\tilde{w}$, $\tilde{c}$ and $\tilde{\gamma}$ depend on the geometry of the problem. The convergence speed is in $O(1/\varepsilon)$. The parallelisation speedup factor is β/τ.

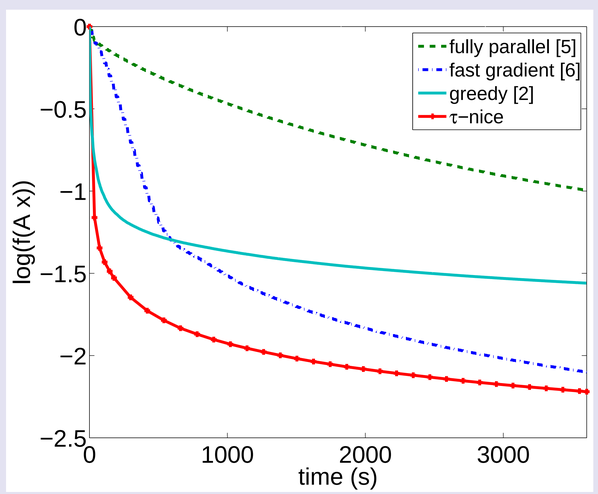
Numerical results

Malicious URL dataset: $m = 2,396,130$ examples, $n = 3,231,961$ features, $\omega = 414$.

Increasing the number of processors leads to acceleration: the time needed to reach -1.85 decreases as β/τ



Comparison with alternative approaches ($\tau = 16$ processors are used in each case)



Conclusion

- Iteration complexity of the order of $O(1/\varepsilon)$ even if the minimising sequences may diverge to infinity.
- Some parameters are difficult to compute and depend on the geometry of the problem but we give closed form formulas for all the parameters actually used in the algorithm.
- Random samplings are well suited to parallel coordinate descent: small cost per iteration, inter-core communication and β value.
- The numerical experiments demonstrate the efficiency of parallel coordinate descent with independent sampling, especially for large scale problems.
- The framework allows us to add bound constraints or a l_1 regulariser.

Acknowledgement

My work was supported by the British government through the EPSRC grant EP/I017127/1 (Mathematics for Vast Digital Resources)

References

[1] O. Fercoq, "Parallel coordinate descent for the Adaboost problem," in *Proc. of the International Conf. on Machine Learning and Applications*, 2013.

[2] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[3] O. Fercoq and P. Richtárik, "Smooth minimization of nonsmooth functions by parallel coordinate descent," 2013, arXiv:1309.5885.

[4] P. Richtárik and M. Takáč, "Parallel coordinate descent methods for big data optimization problems," *arXiv:1212.0873*, November 2012.

[5] M. Collins, R. E. Schapire, and Y. Singer, "Logistic regression, adaboost and bregman distances," *Machine Learning*, vol. 48, no. 1-3, pp. 253–285, 2002.

[6] I. Mukherjee, K. Canini, R. Frongillo, and Y. Singer, "Parallel boosting with momentum," 2013.