Conjugate Descent for the Minimum Norm Problem

Alberto Torres-Barrán
Department of Computer Science
Universidad Autónoma de Madrid
28049 Madrid, Spain
alberto.torres@uam.es

José R. Dorronsoro
Department of Computer Science
Universidad Autónoma de Madrid
28049 Madrid, Spain
jose.dorronso@uam.es

Abstract

We propose a conjugate descent procedure for the Frank–Wolfe algorithm with swap steps for the Minimum Norm Problem that leads to a substantial decrease in the number of iterations required for the method to converge to a given precision while basically only doubling the iteration cost.

1 Introduction

Given a sample $S = \{x^1, \dots, x^N\}$ the Minimum Norm Problem seeks the pattern x in the convex hull $C(S)$ closest to the origin. More formally, MNP can be stated as a minizing the convex function $f(\alpha) = (\sum \alpha_p x^p) \cdot (\sum \alpha_q x^q) = \alpha^t Q \alpha$ over the simplex $\Delta_N = \{(\alpha_1, \dots, \alpha_N) : 0 \leq \alpha_p \leq N, \sum \alpha_p = 1\}$; Q denotes the kernel matrix $Q_{p,q} = x^p \cdot x^q$. While one of the simplest convex constrained minimization problems, it has received a more or less constant attention over the past 50 years or so. In fact, it was one of the problems considered by Frank and Wolfe in his seminal paper [1], it is intimately linked with the SVM problem [2], has been recently revisited, accompanying the renewed interest in Frank–Wolfe (FW) optimization [3] and has been shown to be equivalent to the constrained version of the Lasso problem [4, 5].

MNP is usually solved by iterations of the form $\alpha^{k+1} = \alpha^k + \rho_k d^k$, where d^k is some descent direction (i.e., $d^k \cdot \nabla f(\alpha^k) < 0$). In its FW solution d^k is taken as $d^k = d^k_{FW} = e^{L_k} - \alpha^k$, with $L_k = \arg\min_j \nabla f(\alpha^k)_j$; then we automatically have $\alpha^{k+1} = (1-\rho_k)\alpha^k + \rho_k e^{L_k} \in \Delta_N$ provided $0 \leq \rho_k \leq 1$, i.e., we achieve projection free descent iterations. The drawback of this is that the d^k_{FW} directions may result in a slow sublinear convergence [6] that cannot be improved. Several variants of FW, such as the away steps in [7, 8] have been suggested; the better suited for MNP is the Mitchell-Demnyanov-Malozemov (MDM) algorithm (referred as the swap FW method in [9]).

The MDM descent direction is $d^k = e^{L_k} - e^{U_k}$ with $U_k = \arg\max_j \{\nabla f(\alpha^k)_j : \alpha^k_j > 0\}$. Note that the update $\alpha^k + \rho_k(e^{L_k} - e^{U_k})$ may not lie in Δ_N, i.e., MDM is not projection–free, but, if needed, the projection turns out to be a simple clip of ρ_k. MDM is guaranteed to have linear convergence [10], that is, to have $\|\alpha^{k+1} - \alpha^*\| \leq \lambda \|\alpha^k - \alpha^*\|$, with $\lambda < 1$. In addition, its iterations only need N products just to update $\nabla f(\alpha^{k+1}) = g^{k+1}$ as $g^k + \rho_k\,(Q_{L_k,j} - Q_{U_k,j})$, because MDM's descent direction has only two non zero components (standard FW may need close to $2N$ products to update α^{k+1} and $\nabla f(\alpha^{k+1})$). In fact, MDM is very close to SMO, the standard algorithm in SVM for moderate–to–large size samples. However, MDM's linear convergence is only guaranteed once the face of $C(S)$ facing the origin is reached. Moreover, it is easy to find examples where λ is very close to 1 and convergence is thus slow. All this is due to the d^k directions possibly being far from optimal (very different from, say, $-\nabla f(\alpha^k)$).

Momentum is often used to improve descent directions, for instance, a successful momentum based approach to standard FW has been proposed in [11]. Besides, momentum can be seen from a conjugate descent point of view. Writing the momentum term as $m^k = \alpha^k - \alpha^{k+1}$ we can consider momentum updates of the form $\alpha^{k+1} = \alpha^k + \rho_k\ \left(d^k + \beta_k m^k\right) = \alpha^k + \rho_k\ p^k$ for appropriate ρ_k, β_k, with $p^k = d^k + \beta_k m^k$. Then $m^{k+1} = \alpha^{k+1} - \alpha^k = \rho_k p^k$, and it follows that $p^k = d^k + \beta_k \ m^k = d^k + \beta_k \ \rho_{k-1} \ p^{k-1} = d^k + \sigma_{k-1} \ p^{k-1}$. Therefore, we can combine a descent term with a properly scaled momentum term and arrive at the α^k update writing first $\alpha' = \alpha^k + \rho_k \ \left(d^k + \sigma_{k-1} p^{k-1}\right)$ for appropriate ρ_k and σ_{k-1} and projecting it in $\Delta_N$if needed to obtain the final α^{k+1}.

It is well known in unconstrained quadratic minimization that taking $d^k = -\nabla f(\alpha^k)$, is equivalent to conjugate gradient descent. There are however two computational drawbacks: the projection step and a a potentially much greater $O(N^2)$ iteration complexity. We show next how both can be avoided using a conjugate version of MDM that 1. only needs $2 \times N$ extra float products per iteration and, 2. reduces the projection step to a simple clipping of the ρ coefficient. We will give the details in the next section and numerically compare standard and conjugate MDM in Section 3; a short discussion and conclusions section ends the paper.

2 Conjugate MDM Algorithm

Assume that the conjugate direction $p_k = d^k + \sigma_{k-1} p^{k+1}$ at step k has been chosen and that the previous line minimization along p^{k-1} has been unclipped, i.e., the orthogonality condition $g^k \cdot p^{k-1} = 0$ holds. We find the unconstrained ρ'_k factor by line minimization along p_k, that is, by solving $0 = \nabla f(\alpha^k + \rho \ p^k) \cdot p_k$, which yields $\rho'_k = \frac{-g^k \cdot p^k}{p^k \cdot Qp^k} = \frac{-g^k \cdot d^k - \beta_{k-1} g_k \cdot p^{k-1}}{p^k \cdot Qp^k} = \frac{-g^k \cdot d^k}{p^k \cdot Qp^k}$. Since $g^k \cdot p^k = g^k \cdot d^k < 0$, p^k is always a descent direction. The unconstrained gain in f is now $\frac{1}{2} \frac{(g^k \cdot d^k)^2}{p^k \cdot Qp^k}$ and we can maximize that gain by choosing σ_{k-1} as to minimize $p^k \cdot Qp^k$. This results in $\sigma_{k-1} = -\frac{d^k \cdot Qp^{k-1}}{p^{k-1} \cdot Qp^{k-1}}$ and implies a second orthogonality condition, $p^k \cdot Qp^{k-1} = 0$.

We can summarize now our conjugate MDM updates. If the iteration ending in α^k along p^{k-1} has not been clipped, we 1. compute σ_{k-1} and p^k, 2. compute ρ'_k and $(\alpha')^{k+1}$ and, finally, 3. check whether $(\alpha')^{k+1} \in \Delta_N$ and, if not, clip it accordingly. On the other hand, if clipping has happened, i.e., we have hit the boundary of Δ_N, we simply reset p^{k-1} to 0 as it may lead to further boundary hits; then $p^k = d^k$ and we just perform a standard MDM update in the next iteration.

Working with MDM's descent directions $d = e_L - e_U$ greatly simplifies the previous computations. For this, we use an auxilar vector $\Gamma^j = Qp^j$ and constant $\delta_j = p^j \cdot Qp^j$ that we update at each iteration. We then have $\sigma_{k-1} = -\frac{d^k \cdot Qp^{k-1}}{p^{k-1} \cdot Qp^{k-1}} = -\frac{d^k \cdot \Gamma^{k-1}}{\delta^{k-1}} = \frac{\Gamma_U^{k-1} - \Gamma_L^{k-1}}{\delta^{k-1}}$, $\Gamma^k = Q(d^k + \sigma_{k-1} p^{k-1}) = Q^L - Q^U + \sigma_{k-1} \Gamma^{k-1}$ (Q^j denotes Q'sj–th column) and $\delta_k = p^k \cdot Qp^k = d^k \cdot Qp^k = d^k \cdot \Gamma^k = \Gamma_L^k - \Gamma_U^k$. This result in a general conjugate MDM iteration where starting from the previous α, its gradient $g = Q\alpha$, and p, Γ and δ, we iteratively

1. Select $L = \text{argmin}_i \ g_i, U = \text{argmax}_{\alpha_i > 0} \ g_i$.
2. Get the kernel matrix columns Q^L, Q^U if not previously cached.
3. Compute $\sigma = \frac{\Gamma_L - \Gamma_U}{\delta}$ and update $p = d + \sigma p$, $\Gamma = Q^L - Q^U + \sigma \Gamma$ and $\delta = \Gamma_L - \Gamma_U$.
4. Compute an unconstrained step $\rho' = \frac{g_U - g_L}{\delta}$ and clip it to ρ according to ρ', α and p.
5. Update α and g as $\alpha = \alpha + \rho \ p$; $g = g + \rho \ \Gamma$. If clipping has taken place, reset $\Gamma = 0$, $\delta = 1$.

We recover standard MDM from the preceding by setting $\sigma = 0$, $p = d$ and $\Gamma = Q^L - Q^U$. If N_p and N_Γ denote the number of non-zero components of p and Γ, the cost in produtcs of each iteration is N_p to update p in Item 3 and α in Item 5, N_Γ to update Γ in Item 3 and N to update the gradient g in Item 5. We expect $N_\Gamma \simeq N$ but N_p should be $\ll N$ and similarly the number of non-zero α_i should also be $\ll N$. Thus a conjugate iteration should have theoretical cost about twice as large as that of a standard MDM iteration and should lead to faster training if the number of MDM iterations are more than twice the number of CD MDM ones. In any case, note that the cost of the iterations in which a non–cached kernel column matrix has to be computed will require a much larger $N \ d$ number of products when working with patterns in a d dimensional space or even more in a kernel setting. We finish this section by briefly discussing clipping. First, notice that if $\sum_j p_j^{k-1} = 0$, we also have $\sum_j p_j^k = 0$ and, hence, $\sum_j \alpha_j^{k+1} = 1$. Thus, to ensure $\alpha \in \Delta_N$, we only have to use a ρ_k

value so that $0 \leq \alpha_j^k + \rho p_j^k \leq 1$. Set $\mathcal{P}_k^+ = \{i : p_i^k > 0\}$ and $\mathcal{P}_k^- = \{i : p_i^k < 0\}$; we want $0 < \rho_k \leq \frac{1-\alpha_i^k}{p_i^k}$ if $i \in \mathcal{P}_k^+$ and $0 < \rho_k \leq -\frac{\alpha_i^k}{p_i^k}$ if $i \in \mathcal{P}_k^-$. Thus, setting $\rho^+ = \min\left\{\frac{1-\alpha_i^k}{p_i^k} : i \in \mathcal{P}_k^+\right\}$ and $\rho^- = \min\left\{-\frac{\alpha_i^k}{p_i^k} : i \in \mathcal{P}_k^-\right\}$, we just take $\rho_k = \min\{\rho'_k, \rho^+, \rho^-\}$.

3 Numerical Experiments

We shall illustrate the smaller number of iterations needed by the conjugate MDM algorithm on two different settings. The first one is a synthetic dataset where we generate for various d values d–dimensional Gaussian blobs with a standard deviation of 1 centered at the origin (this makes it relatively easy for standard MDM), to which we add an extra 1 component so that the MNP has $(0, \dots, 0, 1)$ as solution. In the second one we will work with a number of datasets from the UCI repository and apply first the reduction proposed by M. Jaggi [4] of the constrained Lasso problem with data matrix X, target y and constraint ρ to a Nearest Point Problem between y/ρ and the convex hull spanned by the columns of X and $-X$. This can be easily transformed to an MNP one by subtracting y/ρ from these columns [5].

In all cases we will apply the MDM algorithm and its conjugate variant considering all sample patterns as starting points and counting the number of iterations until a convergence threshold is achieved. For this we use the KKT conditions of MNP that imply at an optimal α that $\Delta = \Delta(\alpha) = \max_{\alpha_i > 0} \nabla f(\alpha)_i - \min_j \nabla f(\alpha)_j \leq 0$. Accordingly, we will stop the iterations when $\Delta \leq \epsilon$ for some prefixed ϵ.

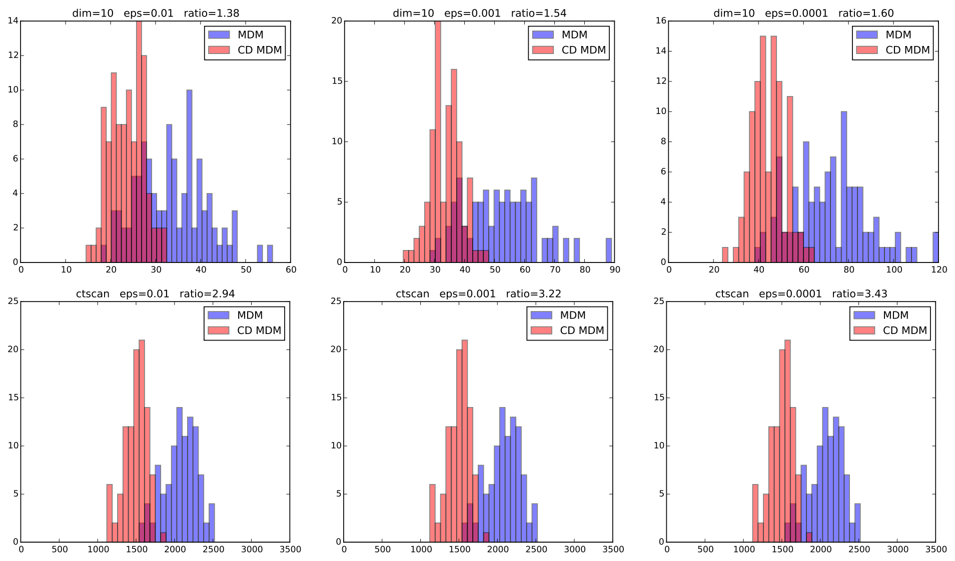

Figure 3 (top) shows histograms of the number of iterations needed by standard and conjugate MDM for random blob synthetic data with sample size 100 and dimension 50. The bottom row shows a similar figure for the $ctscan$ dataset, both with three different ϵ values (from left to right 0.01, 0.001 and 0.0001). The MDM histograms are clearly to the right of the conjugate ones and in almost every case the ratio of the medians grows with the precision ϵ. This ratio is smaller for the synthetic problem (relatively easy for MDM) but much more marked for `ctscan`. Similar ratios for the other problems are given in Table 1. As it can be seen, conjugate MDM always improves on the standard one, with high ratios in several cases.

4 Discussion and Conclusions

We have shown how a conjugate variant of the MDM algorithm (i.e., the Frank–Wolfe algorithm with swap steps) for the MNP achieves a substantial reduction of the number of iterations needed to

Table 1: Median ratios of the number of iterations of standard and conjugate MDM.

ϵ	dim 10	dim 50	dim 100	prostate	housing	year	ctscan	cpusmall
0.01	1.38	1.43	1.31	2.51	2.39	1.95	2.94	1.21
0.001	1.54	1.58	1.35	2.44	2.52	2.02	3.22	1.13
0.0001	1.60	1.73	1.36	2.44	2.51	2.07	3.44	1.14

arrive at a given convergence precision. While the MNP problem has an interest, both on its own and also as it provides an alternative way of solving the constrained Lasso problem, a potentially more interesting application of conjugate directions lies with the well known SMO algorithm to build SVMs either for classification or regression. The basic structure of the SMO iterations is very similar to the MDM ones and so would be the construction of conjugate descent directions. Thus a large reduction in the number of iterations could also be expected as well as in training times. This and also the consideration of momentum terms not only on α but also on the descent direction d is the subject of ongoing research.

Acknowledgments

With partial support from Spain's grants TIN2013-42351-P and S2013/ICE-2845 CASI-CAM-CM, and also of the Cátedra UAM–ADIC in Data Science and Machine Learning. The first author is also supported by the FPU–MEC grant AP-2012-5163. The authors also gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM

References

[1] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[2] J. L. Lázaro and J. R. Dorronsoro, "Simple proof of convergence of the SMO algorithm for different SVM variants," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 23, no. 7, pp. 1142–1147, 2012.

[3] K. L. Clarkson, "Coresets, sparse greedy approximation, and the frank-wolfe algorithm," *ACM Trans. Algorithms*, vol. 6, no. 4, pp. 63:1–63:30, Sep. 2010.

[4] M. Jaggi, "An Equivalence between the Lasso and Support Vector Machines," in *Regularization, Optimization, Kernels, and Support Vector Machines*. CRC Press, 2014.

[5] C. Alaíz, A. Torres, and J. R. Dorronsoro, "Solving constrained lasso and elastic net using ν–svms," in *Proceedings of ESANN 2015, Bruges, Belgium, 22-24 April 2015*, 2015, pp. 1382–1390.

[6] E. Gilbert, "Minimizing the Quadratic Form on a Convex Set," *SIAM Journal on Control*, vol. 4, pp. 61–79, 1966.

[7] P. Wolfe, "Convergence theory in nonlinear programming," in *Integer and Nonlinear Programming*, J. Abadie, Ed. North–Holland, 1970, pp. 1–36.

[8] J. GuéLat and P. Marcotte, "Some comments on Wolfe's 'away step'," *Mathematical Programming*, vol. 35, no. 1, 1986.

[9] R. Ñanculef, E. Frandi, C. Sartori, and H. Allende, "A novel frank-wolfe algorithm. analysis and applications to large-scale SVM training," *Inf. Sci.*, vol. 285, pp. 66–99, 2014.

[10] J. L. Lázaro and J. R. Dorronsoro, "Linear convergence rate for the MDM algorithm for the nearest point problem," *Pattern Recognition*, vol. 48, no. 4, pp. 1510–1522, 2015.

[11] E. Frandi, R. Ñanculef, and J. A. K. Suykens, "A partan-accelerated frank-wolfe algorithm for large-scale SVM classification," in *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, 2015, pp. 1–8.