GENO – GENeric Optimization for Classical Machine Learning (Supplemental Material)

1 More Experiments

Here, we add more experiments that illustrate the versatility and efficiency of the GENO approach.

1.1 Support Vector Machines

Support Vector Machines (SVMs) [9] have been studied intensively and are widely used, especially in combination with kernels [28]. They remain populat, as is indicated by the still rising citation count of the popular and heavily-cited solver LIBSVM [5]. The dual formulation of an SVM is given as the following quadratic optimization problem

$$\begin{array}{ll} \min_a & \frac{1}{2}(a \odot y)^\top K(a \odot y) - \|a\|_1 \\ \text{s.t.} & y^\top a = 0 \\ & 0 \leq a \leq c, \end{array}$$

where $K \in \mathbb{R}^{m \times m}$ is a kernel matrix, $y \in \{-1, +1\}^m$ is a binary label vector, $c \in \mathbb{R}$ is the regularization parameter, and $\odot$ is the element-wise multiplication. While the SVM problem with a kernel can also be solved in the primal [6], it is traditionally solved in the dual. We use a Gaussian kernel, i.e., $K_{ij} = \exp\left(-\gamma \|X_i - X_j\|_2^2\right)$ and standard data sets. We set the bandwith parameter $\gamma = 1/2$ which corresponds to roughly the median of the pairwise data point distances and set $C = 1$. Table 1 shows that the solver generated by GENO is as efficient as LIBSVM which has been maintained and improved over the last 15 years. Both solvers outperform general purpose approaches like CVXPY with OSQP [31], SCS [26], Gurobi [18], or Mosek [24] by a few orders of magnitude.

Table 1: Running times in seconds for solving a dual Gaussian-kernelized SVM. The optimality gap is close to 10^{-4} for all solvers and data sets. Missing entries in the table indicate that the solver did not finish within one hour.

Solver	Datasets							
	ionosphere	australian	diabetes	a1a	a5a	a9a	w8a	cod-rna
GENO	0.009	0.024	0.039	0.078	1.6	30.0	25.7	102.1
LIBSVM	0.005	0.010	0.009	0.088	1.0	18.0	78.6	193.1
SCS	0.442	1.461	3.416	11.707	517.5	-	-	-
OSQP	0.115	0.425	0.644	3.384	168.2	-	-	-
Gurobi	0.234	0.768	0.992	4.307	184.4	-	-	-
Mosek	0.378	0.957	1.213	6.254	152.7	-	-	-

1.2 Elastic Net

Elastic net regression [34] has also been studied intensively and is used mainly for mircoarray data classification and gene selection. Given some data $X \in \mathbb{R}^{m \times n}$ and a response $y \in \mathbb{R}^m$, elastic net regression seeks to minimize

$$\frac{1}{2m} \|Xw - y\|_2^2 + \alpha \left(\lambda \|w\|_1 + \frac{1 - \lambda}{2} \|w\|_2^2 \right),$$

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

where α and λ are the corresponding elastic net regularization parameters. The most popular solver is glmnet, a dual coordinate descent approach that has been implementated in Fortran [17]. In our experiments, we follow the same setup as in [17]. We generated Gaussian data $X \in \mathbb{R}^{m \times n}$ with m data points and n features. The outcome values y were generated by

$$y = \sum_{j=1}^{n} X_j \beta_j + k \cdot z,$$

where $\beta_j = (-1)^j \exp(-j/10)$, $z \sim \mathcal{N}(0, 1)$, and k is chosen such that the signal-to-noise ratio is 3. We varied the number of data points m and the number of features n. The results are shown in Table 2. It can be seen that the solver generated by GENO is as efficient as glment and orders of magnitude faster than comparable state-of-the-art general purpose approaches like CVXPY coupled with ECOS, SCS, Gurobi, or Mosek. Note, that the OSQP solver could not be run on this problem since CVXPY raised the error that it cannot convert this problem into a QP.

Table 2: Running times for the elastic net regression problem in seconds. Missing entries in the table indicate that the solver did not finish within one hour. The optimality gap is about 10^{-8} for all solvers which is the standard setting for glmnet.

m	n	Solvers					
		GENO	glmnet	ECOS	SCS	Gurobi	Mosek
1000	1000	0.11	0.10	43.27	2.33	21.14	1.77
2000	1000	0.14	0.08	202.04	9.24	58.44	3.52
3000	1000	0.18	0.08	513.78	22.86	114.79	5.38
4000	1000	0.21	0.09	-	38.90	185.79	7.15
5000	1000	0.27	0.11	-	13.88	151.08	8.69
1000	5000	1.74	0.62	-	28.69	-	13.06
2000	5000	1.49	1.41	-	45.79	-	27.69
3000	5000	1.58	2.02	-	81.83	-	50.99
4000	5000	1.24	1.88	-	135.94	-	67.60
5000	5000	1.41	1.99	-	166.60	-	71.92
5000	10000	4.11	4.75	-	-	-	-
7000	10000	4.76	5.52	-	-	-	-
10000	10000	4.66	3.89	-	-	-	-
50000	10000	13.97	6.34	-	-	-	-
70000	10000	18.82	11.76	-	-	-	-
100000	10000	23.38	23.42	-	-	-	-

1.3 Non-negative Least Squares

Least squares is probably the most widely used regression method. Non-negative least squares is an extension that requires the output to be non-negative. It is given as the following optimization problem

$$\begin{array}{ll} \min_x & \|Ax - b\|_2^2 \\ \text{s. t.} & x \geq 0, \end{array}$$

where $A \in \mathbb{R}^{m \times n}$ is a given design matrix and $b \in \mathbb{R}^m$ is the response vector. Since non-negative least squares has been studied intensively, there is a plenitude of solvers available that implement different optimization methods. An overview and comparison of the different methods can be found in [30]. Here, we use the accompanying code described in [30] for our comparison. We ran two sets of experiments, similarly to the comparisons in [30], where it was shown that the different algorithms behave quite differently on these problems. For experiment (i), we generated random data $A \in \mathbb{R}^{2000 \times 6000}$, where the entries of A were sampled uniformly at random from the interval $[0, 1]$ and a sparse vector $x \in \mathbb{R}^{6000}$ with non-zero entries sampled also from the uniform distribution of $[0, 1]$ and a sparsity of 0.01. The outcome values were then generated by $y = \sqrt{0.003} \cdot Ax + 0.003 \cdot z$, where $z \sim \mathcal{N}(0, 1)$. For experiment (ii), $A \in \mathbb{R}^{6000 \times 3000}$ was drawn form a Gaussian distribution

and x had a sparsity of 0.1. The outcome variable was generated by $y = \sqrt{1/6000} \cdot Ax + 0.003 \cdot z$, where $z \sim \mathcal{N}(0, 1)$. The differences between the two experiments are the following: (1) The Gram matrix $A^\top A$ is singular in experiment (i) and regular in experiment (ii), (2) The design matrix A has isotropic rows in experiment (ii) which does not hold for experiment (i), and (3) x is significantly sparser in (i) than in (ii). We compared the solver generated by GENO with the following approaches: the classical Lawson-Hanson algorithm [20], which employs an active set strategy, a projected gradient descent algorithm combined with an Armijo-along-projection-arc line search [1, Ch 2.3], a primal-dual interior point algorithm that uses a conjugate gradient descent algorithm [2] with a diagonal preconditioner for solving the Newton system, a subspace Barzilai-Borwein approach [19], and Nesterov's accelerated projected gradient descent [25]. Figure 1 shows the results for both experiments. Note, that the Barzilai-Borwein approach with standard parameter settings diverged on experiment (i) and it would stop making progress on experiment (ii). While the other approaches vary in running time depending on the problem, the experiments show that the solver generated by GENO is always among the fastest compared to the other approaches.

We provide the final running times of the general purpose solvers in Table 3 since obtaining intermediate solutions is not possible for these solvers. Table 3 also provides the function values of the individual solvers. It can be seen, while the SCS solver is considerably faster than the ECOS solver, the solution computed by the SCS solver is not optimal in experiment (i). The ECOS solver provides a solution with the same accuracy as GENO but at a running time that is a few orders of magnitude larger.

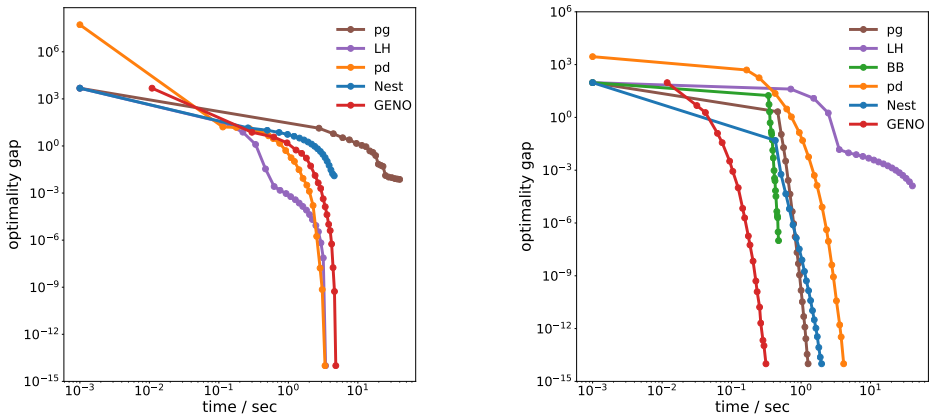

Figure 1: Running times for non-negative least squares regression. The figure on the left shows the running times for the experiment (i) and the figure on the right the running times for experiment (ii). The algorithms are projected gradient descent (pd), Lawson-Hanson (LH), subspace Barzilai-Borwein (BB), primal-dual interior point method (pd), Nesterov's accelerated projected gradient descent (Nest), and GENO.

Table 3: Running times and function values for the non-negative least squares problem.

m	n		GENO	ECOS	SCS	Gurobi	Mosek
2000	6000	time	4.8	689.7	70.4	187.3	24.9
		fval	0.01306327	0.01306327	0.07116707	0.01306330	0.01306343
6000	3000	time	0.3	3751.3	275.5	492.9	58.4
		fval	0.03999098	0.03999098	0.04000209	0.03999100	0.03999114

1.4 Non-linear Least Squares

GENO makes use of a quasi-Newton solver which approximates the Hessian by the weighted sum of the identity matrix and a positive semidefinite, low-rank matrix. One could assume that this does not work well in case that the true Hessian is indefinite, i.e., in the non-convex case. Hence, we also

conducted some experiments on non-convex problems. We followed the same setup and ran the same experiments as in [23] and compared to state-of-the-art solvers that were specifically designed to cope with non-convex problems. Especially, we considered the non-linear least squares problem, i.e., we seek to minimize the function $l(x) = \|\sigma(Ax) - b\|_2^2$, where $A \in \mathbb{R}^{m \times n}$ is a data matrix, $y \in \{0, 1\}^m$ is a binary label vector, and $\sigma(s) = 1/(1 + \exp(-s))$ is the sigmoid function. Figure 2 shows the convergence speed for the data set `w1a` and `a1a`. The state-of-the-art specialized solvers that were introduced in [23] are S-AdaNCG, which is a stochastic adaptive negative curvature and gradient algorithm, and AdaNCD-SCSG, an adaptive negative curvature descent algorithm that uses SCSG [21] as a subroutine. The experiments show that GENO outperforms both algorithms by a large margin. In fact, on the data set `a1a`, both algorithms would not converge to the optimal solution with standard parameter settings. Again, this problem cannot be modeled and solved by CVXPY.

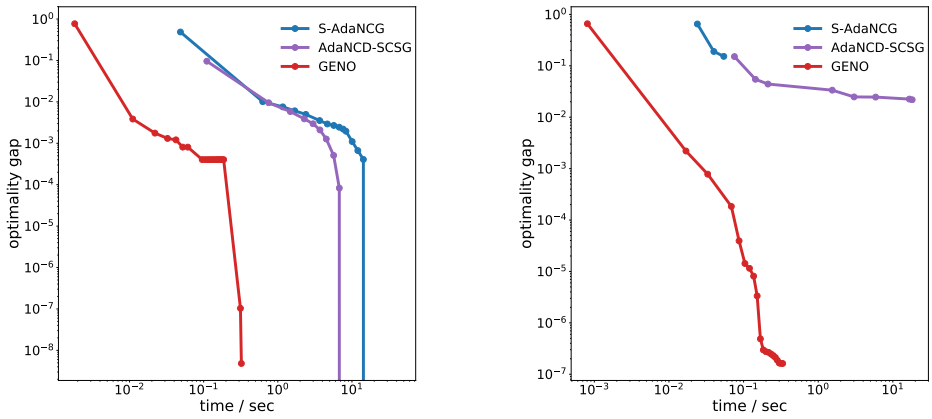

Figure 2: Running times for the non-linear least squares problem. The figure on the left shows the running times for the data set `w1a` and on the right for the data set `a1a`.

1.5 Compressed Sensing

In compressed sensing, one tries to recover a sparse signal from a number of measurements [4, 14]. See the recent survey [27] for an overview on this topic. The problem can be reduced to finding the solution to an underdetermined system of linear equations with minimal ℓ_1-norm. Hence, it can be written as the following optimization problem

$$\begin{aligned} \min_x \quad & \|x\|_1 \\ \text{s. t.} \quad & Ax = b, \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$ is a measurement matrix and $b \in \mathbb{R}^m$ is the vector of m measurements. Note, that this problem is a constrained problem with a non-differentiable objective function. It is known that when matrix A has the restricted isometry property and the true signal x^* is sparse, then Problem (1) recovers the true signal with high probability, if the dimensions m and n are chosen properly [3]. There has been made considerable progress in designing algorithms that come with convergence guarantees [7, 8]. Very recently, in [15] a new and efficient algorithm based on the iterative reweighted least squares (IRLS) technique has been proposed. Compared to previous approaches, their algorithm is simple and achieves the state-of-the-art convergence guarantees for this problem.

We used the same setup and random data set as in [15] and ran the same experiment. The measurement matrix $A \in \mathbb{R}^{150 \times 200}$ had been generated randomly, such that all rows are orthogonal. Then, a sparse signal x^* with only 15 non-zero entries had been chosen and the corresponding measurement vector b had been computed via $b = Ax^*$. We compared to their IRLS algorithm with the long-steps update scheme. Figure 3 shows the convergence speed speed towards the optimal function value as well as the convergence towards feasibility. It can be seen that the solver generated by GENO outperforms the specialized, state-of-the-art IRLS solver by a few orders of magnitude.

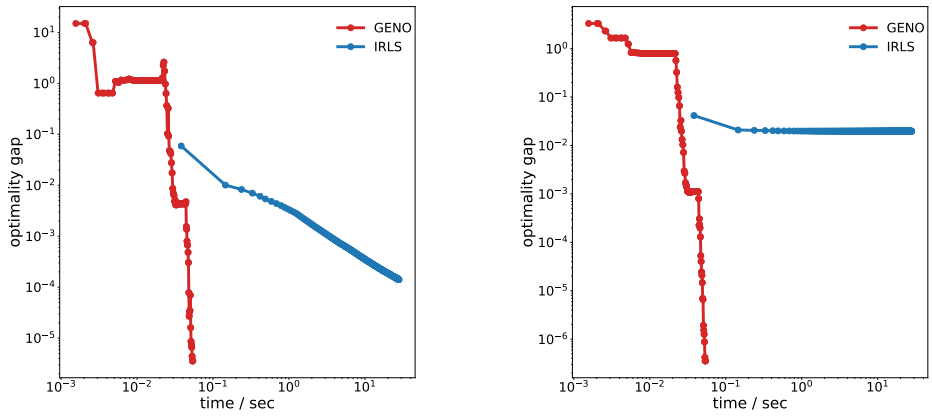

Figure 3: Running times for the compressed sensing problem. The figure on the left shows the convergence to the optimal objective function value and the figure on the right shows the norm of the constraint violation of the iterate $x^{(k)}$, i.e., $\left\|Ax^{(k)} - b\right\|_2$.

2 Summary of Solvers

Table 4: Summary of all solvers that were used in the experiments.

Name	Type	Reference
GENO	quasi-Newton w/ augmented Lagrangian	this paper
OSQP	ADMM	[31]
SCS	ADMM	[26]
ECOS	interior point	[13]
Gurobi	interior point	[18]
Mosek	interior point	[24]
LIBSVM	Frank-Wolfe (SMO)	[5]
LIBLINEAR	conjugate gradient + dual coordinate descent	[16, 33]
SAGA	SGD	[11]
SDCA	SGD	[29]
catalyst SDCA	SGD	[22]
Point-SAGA	SGD	[10]
glmnet	dual coordinate descent	[17]
Lawson-Hanson	direct linear equations solver w/ active set	[20]
projected gradient descent	proximal algorithm	[30]
primal-dual interior point w/ preconditioned conjugate gradient	interior point	[30]
subspace Barlizai-Borwein	quasi-Newton	[19]
Nesterov's method	accelerated gradient descent	[25]
SymANLS	block coordinate descent	[32]
SymHALS	block coordinate descent	[32]
S-AdaNCG	SGD	[23]
AdaNCD-SCSG	SGD	[23]
IRLS	IRLS w/ conjugate gradient method	[15]

3 GENO Models for all Experiments

```
parameters
   Matrix X
   Scalar c
   Vector y
variables
   Scalar b
   Vector w
min
   norm1(w) + c
   * sum(log(exp((-y) .* (X*w
   + vector(b))) + vector(1)))
```

Figure 4: ℓ_1-regularized Logistic Regression

```
parameters
   Matrix X
   Scalar c
   Vector y
variables
   Vector w
min
   0.5 * w' * w
   + c * sum(log(exp((-y) .* (X * w))
   + vector(1)))
```

Figure 5: ℓ_2-regularized Logistic Regression

```
parameters
   Matrix K symmetric
   Scalar c
   Vector y
variables
   Vector a
min
   0.5 * (a.*y)' * K * (a.*y) - sum(a)
st
   a >= 0
   y' * a == 0
```

Figure 6: Support Vector Machine

```
parameters
   Matrix X
   Scalar a1
   Scalar a2
   Scalar n
   Vector y
variables
   Vector w
min
   n * norm2(X*w - y).^2
   + a1 * norm1(w) + a2 * w' * w
```

Figure 7: Elastic Net

```
parameters
   Matrix A
   Vector b
variables
   Vector x
min
   norm2(A*x - b).^2
st
   x > 0
```

Figure 8: Non-negative Least Squares

```
parameters
   Matrix X symmetric
variables
   Matrix U
min
   norm2(X - U*U').^2
st
   U >= 0
```

Figure 9: Symmetric NMF

```
parameters
   Matrix X
   Scalar s
   Vector y
variables
   Vector w
min
   s * norm2(y - 0.5
   * tanh(0.5 * X * w)
   + vector(0.5)).^2
```

Figure 10: Non-linear Least Squares

```
parameters
   Matrix A
   Vector b
variables
   Vector x
min
   norm1(x)
st
   A*x == b
```

Figure 11: Compressed Sensing

References

[1] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[3] Emmanuel J Candès and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[4] Emmanuel J. Candès and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Information Theory*, 52(12):5406–5425, 2006.

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[6] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

[7] Hui Han Chin, Aleksander Madry, Gary L Miller, and Richard Peng. Runtime guarantees for regression problems. In *Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 269–282, 2013.

[8] Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *ACM Symposium on Theory of Computing (STOC)*, pages 273–282, 2011.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[10] Aaron Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems (NIPS)*, pages 676–684, 2016.

[11] Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1646–1654, 2014.

[12] Van Hoan Do, Mislav Blazevic, Pablo Monteagudo, Luka Borozan, Khaled M. Elbassioni, Sören Laue, Francisca Rojas Ringeling, Domagoj Matijevic, and Stefan Canzar. Dynamic pseudo-time warping of complex single-cell trajectories. In *Research in Computational Molecular Biology (RECOMB)*, pages 294–296, 2019.

[13] Alexander Domahidi, Eric Chu, and Stephen P. Boyd. ECOS: An SOCP Solver for Embedded Systems. In *European Control Conference (ECC)*, 2013.

[14] David Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[15] Alina Ene and Adrian Vladu. Improved convergence for ℓ_1 and ℓ_∞ regression via iteratively reweighted least squares. In *International Conference on Machine Learning (ICML)*, 2019. To appear.

[16] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[17] Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[18] Gurobi Optimization, Inc. Gurobi Optimizer. http://www.gurobi.com.

[19] Dongmin Kim, Suvrit Sra, and Inderjit S Dhillon. A non-monotonic method for large-scale non-negative least squares. *Optimization Methods and Software*, 28(5):1012–1039, 2013.

[20] Charles L. Lawson and Richard J. Hanson. *Solving least squares problems*. Classics in Applied Mathematics. SIAM, 1995.

[21] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pages 2348–2358, 2017.

[22] Hongzhou Lin, Julien Mairal, and Zaïd Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3384–3392, 2015.

[23] Mingrui Liu, Zhe Li, Xiaoyu Wang, Jinfeng Yi, and Tianbao Yang. Adaptive negative curvature descent with applications in non-convex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4858–4867, 2018.

[24] MOSEK ApS. The MOSEK optimization software. http://www.mosek.com.

[25] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN USSR (translated as Soviet Math. Docl.)*, 269, 1983.

[26] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

[27] Meenu Rani, S. B. Dhok, and R. B. Deshmukh. A systematic review of compressive sensing: Concepts, implementations and applications. *IEEE Access*, 6:4875–4894, 2018.

[28] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[29] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, 2013.

[30] Martin Slawski. Problem-specific analysis of non-negative least squares solvers with a focus on instances with sparse solutions. https://sites.google.com/site/slawskimartin/code, March 2013.

[31] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *arXiv preprint*, November 2017.

[32] Zhihui Zhu, Xiao Li, Kai Liu, and Qiuwei Li. Dropping symmetry for fast symmetric nonnegative matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5160–5170, 2018.

[33] Yong Zhuang, Yu-Chin Juan, Guo-Xun Yuan, and Chih-Jen Lin. Naive parallelization of coordinate descent methods and an application on multi-core l1-regularized classification. In *International Conference on Information and Knowledge Management (CIKM)*, pages 1103–1112, 2018.

[34] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.