Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

A Reduction of the Elastic Net to Support Vector Machines with an Application to GPU Computing

Quan Zhou*†**, Wenlin Chen**‡**, Shiji Song***†**, Jacob R. Gardner**‡**,**
Kilian Q. Weinberger‡**, Yixin Chen**‡
* Tsinghua National Laboratory for Information Science and Technology (TNList)
† Department of Automation, Tsinghua University, Beijing, 100084 China
‡ Washington University in St. Louis, 1 Brookings Drive, MO 63130
zhouq10@mails.tsinghua.edu.cn, shijis@mail.tsinghua.edu.cn
{wenlinchen, gardner.jake, kilian, ychen25}@wustl.edu

Abstract

Algorithmic reductions are one of the corner stones of theoretical computer science. Surprisingly, to-date, they have only played a limited role in machine learning. In this paper we introduce a formal and practical reduction between two of the most widely used machine learning algorithms: from the Elastic Net (and the Lasso as a special case) to the Support Vector Machine. First, we derive the reduction and summarize it in only 11 lines of MATLAB$^{\text{TM}}$. Then, we demonstrate its high impact potential by translating recent advances in parallelizing SVM solvers directly to the Elastic Net. The resulting algorithm is a parallel solver for the Elastic Net (and Lasso) that naturally utilizes GPU and multi-core CPUs. We evaluate it on twelve real world data sets, and show that it yields identical results as the popular (and highly optimized) *glmnet* implementation but is up-to two orders of magnitude faster.

Introduction

The discovery and rigorous analysis of Algorithmic reductions is arguably amongst the biggest achievements in theoretical computer science (Cormen et al. 2001). It enabled the rise of modern complexity theory and the establishment of complexity equivalence-classes (such as P and NP). Reductions provide important links between seemingly different algorithms, and often allow for theoretical results about one algorithm to be transferred to another.

Despite this undeniable success in traditional theoretical computer science, reductions have found little traction within the machine learning community (with only few notable exceptions (Langford and Zadrozny 2005)). In this paper, we present a formal reduction of the Elastic Net (Zou and Hastie 2005) and Lasso (as a special case) (Hastie, Tibshirani, and Friedman 2009) to the Support Vector Machine with squared hinge loss[1] (Cortes and Vapnik 1995). In other words, we show how any Elastic Net regression problem can be transformed into a binary SVM classification problem, which lead to identical weight vectors (up to scaling). We make several concrete contributions:

We provide a *rigorous derivation* of the reduction from the Elastic Net to the SVM. This reduction sheds new light onto both algorithms and offers a valuable opportunity to translate results from one research community to the other. Although our contribution is first and foremost theoretical, we derive a *practical* linear-time algorithm to perform this reduction and state it in 11-lines of Matlab^{TM} code.

We also demonstrate the high impact potential of our reduction by *transferring* recent breakthroughs in parallel SVM solvers with graphics processing unit (GPU) support (Tyree et al. 2014; Cotter, Srebro, and Keshet 2011) directly to the Elastic Net and the Lasso. We thoroughly *evaluate the correctness and speed* of the resulting parallel Elastic Net implementation on *twelve* real-world data sets (covering the $p \gg n$ and the $n \gg p$ scenarios) and show that it outperforms the most efficient existing implementations consistently in *all* settings by up-to two orders of magnitude — making it the fastest Elastic Net and Lasso solver that we are aware of.

We believe that our result should be of interest to many members of the machine learning community. Our theoretical reduction could give rise to similar reductions of variations of the Elastic Net to other known versions of the SVM. It may also lead to entirely new algorithmic discoveries on both sides. Our application to utilize GPU hardware and its associated speedup could be of great use to practitioners, *e.g.* in computational biology, neuroscience or sparse coding. Finally, the fact that utilizing SVM implementations for the Elastic Net leads to drastically faster implementations proves that there is still a lot of room to improve Elastic Net solvers by incorporating parallelism and modern hardware.

Notation and Background

Throughout this paper we type vectors in bold ($\mathbf{x}$), scalars in regular (C or b), matrices in capital bold ($\mathbf{X}$). Specific entries in vectors or matrices are scalars and follow the cor-

[1]In this work, as we do not use the standard SVM with linear hinge loss, we refer to the SVM with squared hinge loss simply as *SVM*.

responding convention, *i.e.* the i^{th} dimension of vector $\mathbf{x}$ is x_i. In contrast, depending on the context, $\mathbf{x}^{(i)}$ refers to the i^{th} *column* in matrix $\mathbf{X}$ and $\mathbf{x}_i$ refers to the transpose of its i^{th} *row*. $\mathbf{1}$ is a column vector of all 1. In the remainder of this section we briefly review the Elastic Net and SVM. In derivations we highlight changes in equations in red.

Elastic Net. In the regression scenario we are provided with a data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each $\mathbf{x}_i \in \mathcal{R}^p$ and the labels are real valued, *i.e.* $y_i \in \mathcal{R}$. Let $\mathbf{y} = (y_1, \ldots, y_n)^\top$ be the response vector and $\mathbf{X} \in \mathcal{R}^{n \times p}$ be the design matrix where the (transposed) i^{th} row of $\mathbf{X}$ is $\mathbf{x}_i$. As in (Zou and Hastie 2005), we assume throughout that the response vector is centered and all features are normalized.

The Elastic Net (Zou and Hastie 2005) learns a (sparse) linear model to predict y_i from $\mathbf{x}_i$ by minimizing the squared loss with L2-regularization and an L1-norm constraint,

$$\min_{\boldsymbol{\beta} \in \mathcal{R}^p} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 \qquad \text{such that } |\boldsymbol{\beta}|_1 \leq t, \quad (1)$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]^\top \in \mathcal{R}^p$ denotes the weight vector, $\lambda_2 \geq 0$ is the L2-regularization constant and $t > 0$ the L1-norm budget. In the case where $\lambda_2 = 0$, the Elastic Net reduces to the Lasso (Hastie, Tibshirani, and Friedman 2009) as a special case. The L1 constraint encourages the solution to be sparse. The L2 regularization coefficient has several desirables effects: 1. it makes the problem strictly convex and therefore yields a unique solution; 2. if features are highly correlated it assigns non-zero weights to all of them (making the solution more stable); 3. if $p \gg n$ the optimization does not become unstable for large values of t.

SVM with squared hinge loss. In the classification setting we are given a training dataset $\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i=1}^m$ where $\hat{\mathbf{x}}_i \in \mathcal{R}^d$ and $\hat{y}_i \in \{+1, -1\}$. The linear SVM with squared hinge (Schölkopf and Smola 2002) learns a separating hyperplane, parameterized by a weight vector $\mathbf{w} \in \mathcal{R}^d$—the unique solution of the following optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1}^m \xi_i^2 \quad \text{s.t.} \quad \hat{y}_i \mathbf{w}^\top \hat{\mathbf{x}}_i \geq 1 - \xi_i \; \forall i. \quad (2)$$

Here, $C > 0$ denotes the regularization constant. Please note that we do not include any bias term, *i.e.* we assume that the separating hyperplane passes through the origin.

This problem is often solved in its dual formulation, which is equivalent to solving (2) directly due to strong duality. Without replicating the derivation (Hsieh et al. 2008; Schölkopf and Smola 2002), we state the dual of (2) as:

$$\min_{\alpha_i \geq 0} \|\hat{\mathbf{Z}}\boldsymbol{\alpha}\|_2^2 + \frac{1}{2C}\sum_{i=1}^m \alpha_i^2 - 2\sum_{i=1}^m \alpha_i, \quad (3)$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$ denote the dual variables and $\hat{\mathbf{Z}} = (\hat{y}_1\hat{\mathbf{x}}_1, \ldots, \hat{y}_m\hat{\mathbf{x}}_m)$ is a $d \times m$ matrix, of which the i^{th} column $\mathbf{z}^{(i)}$ consists of input $\hat{\mathbf{x}}_i$ multiplied by its corresponding label $\hat{y}_i$, *i.e.* $\mathbf{z}^{(i)} = \hat{y}_i\hat{\mathbf{x}}_i$. The two formulations (2) and (3) are equivalent and the solutions connect via $\mathbf{w} = \sum_{i=1}^m \alpha_i \hat{y}_i \hat{\mathbf{x}}_i$.

In (3) the data is only accessed through $\hat{\mathbf{Z}}^\top\hat{\mathbf{Z}}$, which corresponds to the inner-product matrix of the input rescaled by the labels, *i.e.* $[\hat{\mathbf{Z}}^\top\hat{\mathbf{Z}}]_{ij} = \hat{y}_i\hat{\mathbf{x}}_i^\top\hat{\mathbf{x}}_j\hat{y}_j$. In scenarios with $d \gg m$, this matrix can be pre-computed and cached in a kernel matrix in $O(m^2)$ memory and $O(d)$ operations, which makes the remaining running time independent of the dimensionality (Schölkopf and Smola 2002). Historically, the dual formulation is most commonly used to achieve non-linear decision boundaries, with the help of the kernel-trick (Schölkopf and Smola 2002). In our case, however, we will only need the linear setting and restrict the kernel (inner-product matrix) to be linear, too.

Both formulations of the SVM can be solved particularly efficiently on modern hardware with Newton's Method (Chapelle 2007; Fan et al. 2008), which offloads the majority of the computation onto matrix operations and therefore can be vectorized and parallelized to achieve near peak computing performance (Tyree et al. 2014).

The Reduction of Elastic Net to SVM

In this section, we derive the equivalence between Elastic Net and SVM, and reduce problem (1) to a specific instance of the SVM optimization problem (3).

Reformulation of the Elastic Net. We start with the Elastic Net formulation as stated in (1). First, we divide the objective and the constraint by t and substitute in a rescaled weight vector, $\boldsymbol{\beta} := \frac{1}{t}\boldsymbol{\beta}$. This step allows us to absorb the constant t entirely into the objective and rewrite (1) as

$$\min_{\boldsymbol{\beta}} \left\|\mathbf{X}\boldsymbol{\beta} - \frac{1}{t}\mathbf{y}\right\|_2^2 + \lambda_2\|\boldsymbol{\beta}\|_2^2 \qquad s.t.\ |\boldsymbol{\beta}|_1 \leq 1. \quad (4)$$

To simplify the L1 constraint, we follow (Schmidt 2005) and split $\boldsymbol{\beta}$ into two sets of non-negative variables, representing positive components as $\boldsymbol{\beta}^+ \geq 0$ and negative components as $\boldsymbol{\beta}^- \geq 0$, *i.e.* $\boldsymbol{\beta} = \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-$. Then we concatenate $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$ together and form a new weight vector $\hat{\boldsymbol{\beta}} = [\boldsymbol{\beta}^+; \boldsymbol{\beta}^-] \in \mathcal{R}_{\geq 0}^{2p}$. The regularization term $\|\boldsymbol{\beta}\|_2^2$ can be expressed as $\sum_{i=1}^{2p} \hat{\beta}_i^{\,2}$, and (4) can be rewritten as

$$\min_{\hat{\beta}_i \geq 0} \left\|[\mathbf{X}, -\mathbf{X}]\,\hat{\boldsymbol{\beta}} - \frac{1}{t}\mathbf{y}\right\|_2^2 + \lambda_2 \sum_{i=1}^{2p} \hat{\beta}_i^{\,2} \qquad s.t. \sum_{i=1}^{2p} \hat{\beta}_i \leq 1. \quad (5)$$

Here the set $\mathcal{R}_{\geq 0}^{2p}$ denotes all vectors in $\mathcal{R}^{2p}$ with all non-negative entries. Please note that, as long as $\lambda_2 \neq 0$, the solution to (5) is unique and satisfies that $\beta_i^+ = 0$ or $\beta_i^- = 0$ for all i.

Barring the (uninteresting) case with extremely large $t \gg 0$, the L1-norm constraint in (1) will always be tight (Boyd and Vandenberghe 2004), *i.e.* $|\hat{\boldsymbol{\beta}}| = 1$.[2] We can incorporate

[2]If t is extremely large, (1) is equivalent to ridge regression (Hastie, Tibshirani, and Friedman 2009), which typically yields completely dense (non-sparse) solutions. There is little reason to ever use such (pathologically) large values of t, as the solution can be found much more efficiently with standard ridge regression and does not do feature selection.

this equality constraint into (5) and obtain

$$\min_{\hat{\beta}_i \geq 0} \left\| [\mathbf{X}, -\mathbf{X}]\, \hat{\boldsymbol{\beta}} - \frac{1}{t}\mathbf{y} \right\|_2^2 + \lambda_2 \sum_{i=1}^{2p} \hat{\beta}_i^{\,2} \quad s.t. \sum_{i=1}^{2p} \hat{\beta}_i = 1. \tag{6}$$

We construct a matrix $\hat{\mathbf{Z}} = [\hat{\mathbf{X}}_1, -\hat{\mathbf{X}}_2]$ such that $\hat{\mathbf{Z}}\hat{\boldsymbol{\beta}} = [\mathbf{X}, -\mathbf{X}]\, \hat{\boldsymbol{\beta}} - \frac{1}{t}\mathbf{y}$. As $\mathbf{1}^\top \hat{\boldsymbol{\beta}} = 1$, we can expand $\mathbf{y} = \mathbf{y}\mathbf{1}^\top \hat{\boldsymbol{\beta}}$ and define $\hat{\mathbf{X}}_1 = \mathbf{X} - \frac{1}{t}\mathbf{y}\mathbf{1}^\top$ and $\hat{\mathbf{X}}_2 = \mathbf{X} + \frac{1}{t}\mathbf{y}\mathbf{1}^\top$. If we substitute $\hat{\mathbf{Z}}$ into (6) it becomes

$$\min_{\hat{\beta}_i \geq 0} \|\hat{\mathbf{Z}}\hat{\boldsymbol{\beta}}\|_2^2 + \lambda_2 \sum_{i=1}^{2p} \hat{\beta}_i^{\,2} \qquad s.t. \sum_{i=1}^{2p} \hat{\beta}_i = 1. \tag{7}$$

In the remainder of this section we show that one can obtain the optimal solution $\hat{\boldsymbol{\beta}}^*$ for (7) by carefully constructing a binary classification data set $\hat{\mathbf{X}}, \hat{\mathbf{y}}$ such that $\hat{\boldsymbol{\beta}}^* = \boldsymbol{\alpha}^*/|\boldsymbol{\alpha}^*|_1$, where $\boldsymbol{\alpha}^*$ is the solution for the SVM dual (3) for $\hat{\mathbf{X}}, \hat{\mathbf{y}}$.

Data set construction. We construct a binary classification data set with $m = 2p$ samples and $d = n$ features consisting of the columns of $\hat{\mathbf{X}} = [\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2]$. Let us denote this set as $\{(\hat{\mathbf{x}}^{(1)}, \hat{y}_1), \ldots, (\hat{\mathbf{x}}^{(2p)}, \hat{y}_{2p})\}$, where each $\hat{\mathbf{x}}^{(i)} \in \mathcal{R}^n$ and $\hat{y}_1, \ldots, \hat{y}_p = +1$ and $\hat{y}_{p+1}, \ldots, \hat{y}_{2p} = -1$. In other words, the columns of $\hat{\mathbf{X}}_1$ are of class $+1$ and the columns of $\hat{\mathbf{X}}_2$ are of class -1. It is straight-forward to see that for $\hat{\mathbf{Z}} = [\hat{\mathbf{X}}_1, -\hat{\mathbf{X}}_2]$, as used in (7), we have $\hat{\mathbf{Z}} = (\hat{y}_1\hat{\mathbf{x}}_1, \ldots, \hat{y}_m\hat{\mathbf{x}}_m)$, matching the definition in (3). In other words, the solution of (3) with $\hat{\mathbf{Z}}$ is the SVM classifier when applied to $\hat{\mathbf{X}}, \hat{\mathbf{y}}$.

Optimal solution. Let $\boldsymbol{\alpha}^*$ denote the optimal solution of (3), when optimized with this matrix $\hat{\mathbf{Z}}$ and $C = \frac{1}{2\lambda_2}$. We will now reshape the SVM optimization problem (3) into the Elastic Net (7) without changing the optimal solution, $\boldsymbol{\alpha}^*$ (up to scaling). First, knowing the optimal solution to (3), we can add the constraint $\sum_{i=1}^{2p} \alpha_i = |\boldsymbol{\alpha}^*|_1$, which is trivially satisfied at the optimum, $\boldsymbol{\alpha}^*$, and (3) becomes:

$$\min_{\alpha_i \geq 0} \|\hat{\mathbf{Z}}\boldsymbol{\alpha}\|_2^2 + \lambda_2 \sum_{i=1}^{2p} \alpha_i^2 - 2\sum_{i=1}^{2p} \alpha_i. \quad s.t. \sum_{i=1}^{2p} \alpha_i = |\boldsymbol{\alpha}^*|_1. \tag{8}$$

Because of this equality constraint, the last term in the objective function in (8), $-2\sum_{i=1}^{2p} \alpha_i = -2|\boldsymbol{\alpha}^*|_1$, becomes a constant and can be dropped. Removing this constant term does not affect the solution and leads to the following equivalent optimization:

$$\min_{\alpha_i \geq 0} \|\hat{\mathbf{Z}}\boldsymbol{\alpha}\|_2^2 + \lambda_2 \sum_{i=1}^{2p} \alpha_i^2 \quad s.t. \sum_{i=1}^{2p} \alpha_i = |\boldsymbol{\alpha}^*|_1. \tag{9}$$

Note that the only difference between (9) and (7) is the scale of design variables. If we divide[3] the objective by $|\boldsymbol{\alpha}^*|_1^2$ and the constraint by $|\boldsymbol{\alpha}^*|_1$ and introduce a change of variable, $\hat{\beta}_i = \alpha_i / |\boldsymbol{\alpha}^*|_1$ we obtain exactly (7) and its optimal solution $\hat{\boldsymbol{\beta}}^* = \boldsymbol{\alpha}^* / |\boldsymbol{\alpha}^*|_1$.

[3]This is not well defined if $|\alpha^*|_1 = 0$, which is the over-regularized case when $\lambda_2 \to \infty$. Here, the SVM selects no support vectors, *i.e.* $\boldsymbol{\alpha} = \mathbf{0}$, and the solution to (1) is $\boldsymbol{\beta} = \mathbf{0}$.

Implementation details. To highlight the fact that this reduction is not just of theoretical value but highly practical, we summarize it in Algorithm 1 in MATLAB™ code.[4] We refer to our algorithm as *Support Vector Elastic Net (SVEN)*. As mentioned in the *background* section, the dual and primal formulations of SVM have different time complexities and we choose the faster one depending on whether $2p > n$ or vice versa. Line 7 converts the primal variables $\mathbf{w}$ to the dual solution $\boldsymbol{\alpha}$ (Schölkopf and Smola 2002). Many solvers (Bottou 2010; Chapelle 2007; Fan et al. 2008; Tyree et al. 2014) have been developed for the linear SVM problem (2). In practice, it is no problem to find an implementation with no bias term. Some implementations we investigate do not use a bias by default (*e.g. liblinear* (Fan et al. 2008)) and for others it is trivial to remove (Chapelle 2007). In our experiments we use an SVM implementation based on Chapelle's original exact linear SVM implementation (Chapelle 2007) (which can solve the dual and primal formulation respectively). The resulting algorithm is exact and uses a combination of conjugate gradient (until the number of potential support vectors is sufficiently small) and Newton steps. The majority of the computation time is spent in the Newton updates. As pointed out by Tyree et al. 2014, the individual Newton steps can be parallelized trivially by using parallel BLAS libraries (which is the default in MATLAB™), as it involves almost exclusively matrix operations. We also create a GPU version by casting several key matrices as *gpuArray*, a MATLAB™ internal variable type that offloads computation onto the GPU.[5]

Feature selection and Lasso. It is worth considering the algorithmic interpretation of this reduction. Each input $\hat{\mathbf{x}}_i$ in the SVM data set corresponds to a feature of the original Elastic Net problem. Support Vectors correspond to features that are selected, *i.e.* $\beta_i \neq 0$. If $\lambda_2 \to 0$ the Elastic Net becomes LASSO (Hastie, Tibshirani, and Friedman 2009), which has previously been shown to be equivalent to the hard-margin SVM (Jaggi 2013). It is reassuring to see that our formulation recovers this relationship as a special case, as $\lambda_2 \to 0$ implies that $C \to \infty$, converting (2) into the hard-margin SVM. (In practice, to avoid numerical problems with very large values of C, one can treat this case specially and call a hard-margin SVM implementation in lines 6 and 9 of Algorithm 1.)

Time complexity. The construction of the input only requires $O(np)$ operations and the majority of the running time will, in all cases, be spent in the SVM solver. As a result, our algorithm has great flexibility, and for any dataset

[4]For improved readability, some variable names are mathematical symbols and would need to be substituted in clear text (*e.g.* $\boldsymbol{\alpha}$ should be *alpha*)

[5]The *gpuArray* was introduced into MATLAB™ in 2013.

Algorithm 1 MATLAB™ implementation of SVEN.

```
function $\boldsymbol{\beta}$ = SVEN(X, y, t, $\lambda_2$);
[n p] = size(X);
Xnew = [bsxfun(@minus, X, y./t); bsxfun(@plus, X, y./t)]';
Ynew = [ones(p,1); -ones(p,1)];
if $2p > n$ then
    w = SVMPrimal(Xnew, Ynew, C = 1/(2$\lambda_2$));
    $\boldsymbol{\alpha}$ = C * max(1-Ynew.*(Xnew*w),0);
else
    $\boldsymbol{\alpha}$ = SVMDual(Xnew, Ynew, C = 1/(2$\lambda_2$));
end if
$\boldsymbol{\beta}$ = t * ($\boldsymbol{\alpha}$(1:p) - $\boldsymbol{\alpha}$(p+1:2p)) / sum($\boldsymbol{\alpha}$);
```

with n inputs and p dimensions, we can choose an SVM implementation with a running time that is advantageous for that dataset. Chapelle's MATLAB™ implementation can scale in the worst case either $O(n^3)$ (primal mode) or $O(p^3)$ (dual mode) (Chapelle 2007).[6] In the typical case the time complexity is known to be much better. Especially for the dual formulation we can in practice achieve a running time far below $O(p^3)$, as the worst case assumes that all points are support vectors. In the Elastic Net setting, this would correspond to all features being kept. A more realistic practical running time is on the order of $O(\min(p,n)^2)$, depending on the number of features selected (as regulated by t).

SVM implementations with other time complexities can easily be adapted to our setting, for example (Joachims 2006) would allow training in time $O(np)$ and recent work even suggests solvers with sub-linear time complexity (Hazan, Koren, and Srebro 2011) (although the solution might be insufficiently exact for our purposes in practice).

Related Work

The main contribution of this paper is the practical reduction from Elastic Net to SVMs. Our work is inspired by a recent theoretical contribution, (Jaggi 2013), which reveals the close relation between Lasso and hard-margin SVMs. The first equivalence results between sparse approximation and SVMs were discovered by (Girosi 1998), who established a connection in noise free learning settings. We extend this line of work and prove a non-trivial equivalence between the Elastic Net and the soft-margin SVM and we derive a practical algorithm, which we validate experimentally.

Large scale Elastic Net. Our parallel implementation of the Elastic Net showcases the immediate usefulness of our reduction. Although the Elastic Net has been widely deployed in many machine learning applications only little effort has been made towards efficient parallelization. The coordinate gradient descent algorithm has become the dominating strategy for the optimization.

[6]As it is slightly confusing it is worth re-emphasizing that if the original data has n samples with p dimensions, the constructed SVM problem has $2p$ samples with n dimensions.

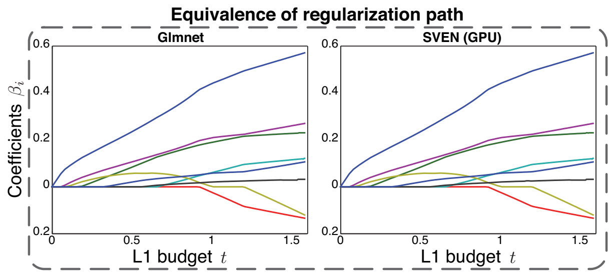

Figure 1: The regularization paths of *glmnet* (left) and SVEN (GPU) (right) on the *prostate* dataset. Each line corresponds to the value of β_i^* as a function of the L1 budget t. The two algorithms match exactly for all values of t.

The state-of-the-art single-core implementation for solving the Elastic Net problem is the *glmnet* package, developed by Friedman (Friedman, Hastie, and Tibshirani 2010). As far as we know, it is the fastest off-the-shelf solver for the Elastic Net, which can be primarily attributed to the very careful Fortran implementation and code optimizations. Due to its inherent sequential nature, the coordinate descent algorithm is extremely hard to parallelize. The *Shotgun* algorithm, (Bradley et al. 2011), is among the first to parallelize coordinate descent for Lasso. This implementation can handle extremely sparse large scale datasets that other software, including *glmnet*, cannot cope with due to memory constraints. The *L1_LS* algorithm (Kim et al. 2007), transforms the Lasso to its dual form directly and uses a log-barrier interior point method for optimization. The optimization is based on the Preconditioned Conjugate Gradient (PCG) method to solve Newton steps, which is suitable for sparse large scale compressed sensing problems. Liu *et al.* (2009) propose the SLEP algorithm to solve Elastic Net by efficiently computing the Euclidean projection onto a closed convex set including L1 ball. Shalev-Shwartz *et al.* (2014) use a proximal version of stochastic dual coordinate ascent, which allows non-smooth regularization function such as Elastic Net. Different than applying gradient based methods, Zou and Hastie (2005) introduce the use of LARS (Efron et al. 2004) to solve the Elastic Net problem.

On the SVM side, one of the most popular and user-friendly implementations is the *libsvm* library (Chang and Lin 2011). However, it is optimized to solve kernel SVM problems using sequential minimal optimization (SMO) (Platt and others 1998), which is not efficient for the specific case of linear SVM. The *liblinear* library (Fan et al. 2008) is specially tailored for linear SVMs, including the squared hinge loss version. However, we find that on modern multi-core platforms (with and without GPU acceleration) algorithms that actively seek updates through matrix operations (Tyree et al. 2014) tend to be substantially faster (in both settings, $p \gg n$ and $n \gg p$).

Experimental Results

In this section, we conduct extensive experiments to evaluate SVEN on twelve real world data sets. We first provide a brief description of the experimental setup and the data sets, then

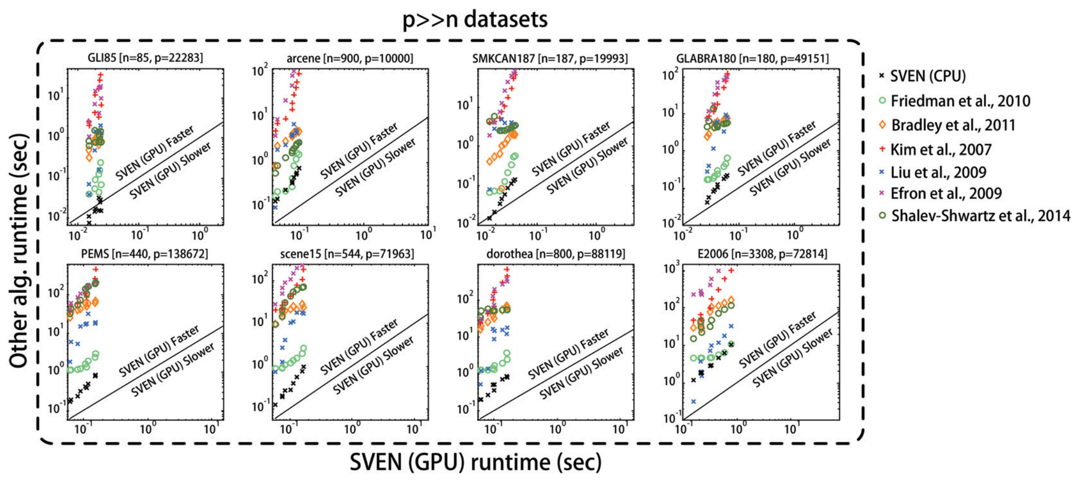

Figure 2: Training time comparison of various algorithms in $p \gg n$ scenarios. Each marker compares an algorithm with SVEN (GPU) on one (out of eight) datasets and one parameter setting. The X,Y-axes denote the running time of SVEN (GPU) and that particular algorithm on the same problem, respectively. All markers are above the diagonal line (except SVEN (CPU) for GLI-85), indicating that SVEN (GPU) is faster than all baselines in all cases.

we investigate the two common scenarios $p \gg n$ and $n \gg p$ separately.

Experimental Setting. We test our method on GPU and (multi-core) CPU under the names of SVEN (GPU) and SVEN (CPU), respectively. We compare against a large number of Elastic Net and Lasso implementations. *glmnet* (Friedman, Hastie, and Tibshirani 2010) is a popular and highly optimized Elastic Net software package. The *Shotgun* algorithm by Bradley et al. (Bradley et al. 2011) parallelizes coordinate gradient descent. *L1_LS* is a parallel MATLAB solver (for Lasso) implemented by Kim et al. (Kim et al. 2007). *SLEP* is an implementation by (Liu, Ji, and Ye 2009). We also compare against an implementation of LARS (Efron et al. 2004). Finally, we compare against the implementation of (Shalev-Shwartz and Zhang 2014). All the experiments were performed on an off-the-shelve desktop with two 8-core Intel(R) Xeon(R) processors of 2.67 GHz and 96GB of RAM. The attached NVIDIA GTX TITAN graphics card contains 2688 cores and 6 GB of global memory.

Regularization path. On all data sets we compare 20 different settings for λ_2 and t. We obtain these by first solving for the full solution path with *glmnet*. The *glmnet* implementation enforces the L1 budget not as a constraint, but as an L1-regularization penalty with a regularization constant λ_1. We obtain the solution path by slowly decreasing $\lambda = \lambda_1 + \lambda_2$ and sub-sampling 20 evenly spaced settings along this path that lead to solutions with distinct number of selected features. If the *glmnet* solution for a particular parameter setting is β^* we obtain t by computing $t = |\beta^*|_1$. This procedure provides us with 20 parameter pairs (λ_2, t) for each data set on which we compare all algorithms. (For the pure Lasso implementations, *shotgun* and *L1_LS*, we set $\lambda_2 = 0$.)

Correctness. Throughout all experiments and all settings of λ_2 and t we find that *glmnet* and SVEN obtain identical results up to the tolerance level. To illustrate the equivalence, Figure 1 shows the regularization path of SVEN (GPU) and *glmnet* on the *prostate cancer* data used in (Zou and Hastie 2005). As mentioned in the previous paragraph, we obtain the original solution path from *glmnet* and evaluate SVEN (GPU) on these parameter settings. The data has eight clinical features (*e.g.* log(cancer volume), log(prostate weight)) and the response is the log of prostate-specific antigen (lpsa). Each line in Figure 1 corresponds to the β_i^* value of some feature $i = 1, \ldots, 8$ as a function of the L1 budget t. The two algorithms lead to exactly matching regularization paths as the budget t increases.

Data sets with $p \gg n$. The $p \gg n$ scenario may be the most common application setting for the Elastic Net and Lasso and there is an abundance of real world data sets. We evaluate all methods on the following eight of them: GLI-85, a dataset that screens a large number of diffuse infiltrating gliomas through transcriptional profiling; SMK-CAN-187, a gene expression dataset from smokers w/o lung cancer; GLA-BRA-180, a dataset concerning analysis of gliomas of different grades; Arcene, a dataset from the NIPS 2003 feature selection contest, whose task is to distinguish cancer versus normal patterns from mass spectrometric data; Dorothea, a sparse dataset from the NIPS 2003 feature selection contest, whose task is to predict which compounds bind to Thrombin.[7] Scene15, a scene recognition data set (Lazebnik, Schmid, and Ponce 2006; Xu, Weinberger, and Chapelle 2012) where we use the binary class 6 and 7 for feature selection; PEMS (Bache and Lichman 2013), a dataset that describes the occupancy rate, between 0 and 1, of different car lanes of San Francisco bay

[7]We removed features with all-zero values across all inputs.

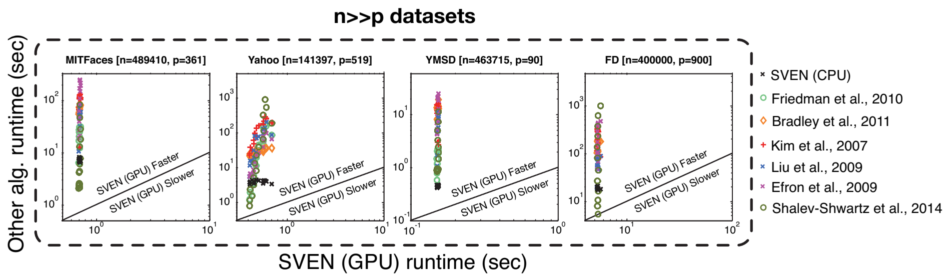

Figure 3: Training time comparison of various algorithms in $n \gg p$ scenarios. Each marker compares an algorithm with SVEN (GPU) on one (out of four) datasets and one parameter setting. The X,Y-axes denote the running time of SVEN (GPU) and that particular algorithm on the same problem, respectively. All markers are above the diagonal line, as SVEN (GPU) is faster in all cases.

area freeways. E2006-tfidf, a sparse dataset whose task is to predict risk from financial reports based on TF-IDF feature representation.[8]

Evaluation ($p \gg n$). Figure 2 depicts training time comparisons of the six baseline algorithms and SVEN (CPU) on the eight datasets with SVEN (GPU). Each marker corresponds to a comparison of one algorithm and SVEN (GPU) in one particular setting along the regularization path. It's y-coordinate corresponds to the training time required for the corresponding algorithm and its x-coordinate corresponds to the training time required for SVEN (GPU) with the exact same L1-norm budget and λ_2 value. All markers above the diagonals corresponds to runs where SVEN (GPU) is faster, and all markers below the diagonal corresponds to runs where SVEN (GPU) is slower.

We observe several general trends: 1. Across *all* eight data sets SVEN (GPU) always outperforms *all* baselines. The only markers below the diagonal are from SVEN (CPU) on the GLI-85 data set, which is the smallest and where the transfer time for the GPU is not offset by the gains in more parallel computation. 2. Even SVEN (CPU) outperforms or matches the performance of the fastest baseline across all data sets. 3. As the L1-budget t increases, the training time increase for all algorithms, but much more strongly for the baselines than for SVEN (GPU). This can be observed by the fact that the markers of one color (*i.e.* one algorithm) follow approximately lines with much steeper slope than the diagonal.

Data sets with $n \gg p$. For the $n \gg p$ setting, we evaluate all algorithms on four additional datasets. MITFaces, a facial recognition dataset; the Yahoo learning to rank dataset, a dataset concerning the ranking of webpages in response to a search query; YearPredictionMSD (YMSD), a dataset of songs with the goal to predict the release year of a song from audio features; and FD, another face detection dataset.

Evaluation ($n \gg p$). A comparison to all methods on all four datasets can be found in Figure 3. The speedups of SVEN (GPU) are even more pronounced in this setting. The training time of SVEN (GPU) is completely dominated by the kernel computation and therefore almost identical for all values of t and λ_2. Consequently all markers follow vertical lines in all plots. The massive speedup of up to two orders of magnitude obtained by SVEN (GPU) over the baseline methods squashes all markers at the very left most part of the plot. *glmnet* fails to complete the FD dataset due to memory constraints and therefore we evaluate on the λ_2 and t values along the solution path from the other face recognition data set, MITFaces. As in the $p \gg n$ case, all solutions returned by both versions of SVEN match those of *glmnet* exactly.

Conclusion

This paper provides a rare case of an initially purely theoretical contribution with strong practical implications. Algorithmic reductions are very promising for the machine learning literature for many theoretical and practical reasons. In particular, the use of algorithmic reductions to obtain parallelization and improved scalability has several intriguing benefits: 1. no new learning algorithm has to be implemented and optimized by hand (besides the small transformation code); 2. the burden of code maintenance reduces to the single highly optimized algorithm (in this case SVM); 3. the implementation is very reliable from the start as almost the entire execution time is spent in a well established and tested implementation. The squared hinge-loss SVM formulation can be solved almost entirely with large matrix operations, which are already parallelized (and maintained) by high-performance experts through BLAS libraries (*e.g.* CUBLAS for NVIDIA GPUs).

We hope that this paper will benefit the community in at least two ways: Practitioners will obtain a new stable and blazingly fast implementation of Elastic Net and Lasso; and machine learning researchers might identify and derive different algorithmic reductions to facilitate similar performance improvements with other learning algorithms. SVEN is available at http://sven.weinbergerweb.com.

Acknowledgements. QZ and SS are supported by National Natural Science Foundation of China under Grants 41427806 and 61273233, National Key Technology R&D Program under grant 2012BAF01B03, Research Fund for the Doctoral Program of Higher Education under Grant 20130002130010, the Project of China Ocean Association under Grant DY125-25-02, and Tsinghua University Initia-

[8]Here, we reduce the training set size by subsampling to match the size of the test set, $n = 3308$.

tive Scientific Research Program under Grant 20131089300. KQW, JRG, YC, and WC are supported by NIH grant U01 1U01NS073457-01 and NSF grants IIA-1355406, IIS-1149882, EFRI-1137211, CNS-1017701, CCF-1215302 and IIS-1343896. The authors thank Martin Jaggi for clarifying discussions and suggestions, and John Tran from NVIDIA for providing us with GPU cards.

References

Bache, K., and Lichman, M. 2013. UCI machine learning repository.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer. 177–186.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Bradley, J.; Kyrola, A.; Bickson, D.; and Guestrin, C. 2011. Parallel coordinate descent for l1-regularized loss minimization. In *ICML*, 321–328.

Chang, C., and Lin, C. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27.

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

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C.; et al. 2001. *Introduction to algorithms*, volume 2. MIT press Cambridge.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Cotter, A.; Srebro, N.; and Keshet, J. 2011. A gpu-tailored approach for training kernelized svms. In *SIGKDD*, 805–813.

Efron, B.; Hastie, T.; Johnstone, I.; and Tibshirani, R. 2004. Least angle regression. *The Annals of statistics* 32(2):407–499.

Fan, R.; Chang, K.; Hsieh, C.; Wang, X.; and Lin, C. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research* 9:1871–1874.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software* 33(1):1.

Girosi, F. 1998. An equivalence between sparse approximation and support vector machines. *Neural computation* 10(6):1455–1480.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The elements of statistical learning*, volume 2. Springer.

Hazan, E.; Koren, T.; and Srebro, N. 2011. Beating sgd: Learning svms in sublinear time. In *NIPS*, 1233–1241.

Hsieh, C.; Chang, K.; Lin, C.; Keerthi, S.; and Sundararajan, S. 2008. A dual coordinate descent method for large-scale linear svm. In *ICML*, 408–415. ACM.

Jaggi, M. 2013. An equivalence between the lasso and support vector machines. *arXiv preprint arXiv:1303.1152*.

Joachims, T. 2006. Training linear svms in linear time. In *SIGKDD*, 217–226. ACM.

Kim, S.; Koh, K.; Lustig, M.; Boyd, S.; and Gorinevsky, D. 2007. An interior-point method for large-scale l 1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of* 1(4):606–617.

Langford, J., and Zadrozny, B. 2005. Relating reinforcement learning performance to classification performance. In *ICML*, 473–480.

Lazebnik, S.; Schmid, C.; and Ponce, J. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, 2169–2178.

Liu, J.; Ji, S.; and Ye, J. 2009. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University.

Platt, J., et al. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. *technical report msr-tr-98-14, Microsoft Research*.

Schmidt, M. 2005. Least squares optimization with l1-norm regularization. *CS542B Project Report*.

Schölkopf, B., and Smola, A. 2002. Learning with kernels.

Shalev-Shwartz, S., and Zhang, T. 2014. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *ICML*, 64–72.

Tyree, S.; Gardner, J.; Weinberger, K.; Agrawal, K.; and Tran, J. 2014. Parallel support vector machines in practice. *arXiv preprint arXiv:1404.1066*.

Xu, Z.; Weinberger, K.; and Chapelle, O. 2012. The greedy miser: Learning under test-time budgets. In *ICML*, 1175–1182.

Zou, H., and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67(2):301–320.