# Least Squares Revisited: Scalable Approaches for Multi-class Prediction

**Alekh Agarwal** ALEKHA@MICROSOFT.COM
Microsoft Research New York, NY

**Sham M. Kakade** SKAKADE@MICROSOFT.COM
Microsoft Research Cambridge, MA

**Nikos Karampatziakis** NIKOSK@MICROSOFT.COM
Microsoft Cloud and Information Services Lab, Redmond, WA

**Le Song** LSONG@CC.GATECH.EDU
College of Computing, Georgia Tech, Atlanta, Georgia

**Gregory Valiant** VALIANT@STANFORD.EDU
Computer Science Department, Stanford University, CA

## Abstract

This work provides simple algorithms for multi-class (and multi-label) prediction in settings where both the number of examples $n$ and the data dimension $d$ are relatively large. These robust and parameter free algorithms are essentially iterative least-squares updates and very versatile both in theory and in practice. On the theoretical front, we present several variants with convergence guarantees. Owing to their effective use of second-order structure, these algorithms are substantially better than first-order methods in many practical scenarios. On the empirical side, we show how to scale our approach to high dimensional datasets, achieving dramatic computational speedups over popular optimization packages such as Liblinear and Vowpal Wabbit on standard datasets (MNIST and CIFAR-10), while attaining state-of-the-art accuracies.

## 1. Introduction

The aim of this paper is to develop robust and scalable algorithms for multi-class classification problems with $k$ classes, where the number of examples $n$ and the number of features $d$ is simultaneously quite large. Typically, such problems have been approached by the minimization of a convex surrogate loss, such as the multiclass hinge-loss or the multiclass logistic loss, or reduction to convex binary subproblems such as one-versus-rest. Given the size of the problem, (batch or online) first-order methods are typically the methods of choice to solve these underlying optimization problems. First-order updates easily scale to large $d$. To deal with the large number of examples, online methods are very appealing in the single machine setting, while batch methods are often preferred in distributed settings.

Empirically however, these first-order approaches are often found to be lacking. Many natural high-dimensional data such as images, audio, and video typically result in ill-conditioned optimization problems. While each iteration of a first-order method is fast, the number of iterations needed unavoidably scale with the condition number of the data matrix (Nemirovsky & Yudin, 1983), even for simple generalized linear models (henceforth GLM).

A natural alternative in such scenarios is to use second-order methods, which are robust to the conditioning of the data. In this paper, we present simple second-order methods for multiclass prediction in GLMs. The methods are parameter free, robust in practice and admit easy extensions. As an example, we show a more sophisticated variant which learns the unknown link function in the GLM simultaneously with the weights. Finally, we also present a wrapper algorithm which tackles the difficulties typically encountered in applying second-order methods to high-dimensional problems. This can be viewed as a block-coordinate descent style stagewise regression pro-

*Proceedings of the $31^{st}$ International Conference on Machine Learning*, Beijing, China, 2014. JMLR: W&CP volume 32. 

cedure that incrementally solves least-squares problems on small batches of features. The result of this overall development is a suite of techniques that are simple, versatile and substantially faster than several other state-of-the-art optimization methods. Finally, we empirically find that in ill-conditioned datasets, such as images, these methods consistently outperform first-order methods in generalization.

**Our Contributions:** Our work has three main contributions. Working in the GLM framework: $\mathbb{E}[y \mid x] = g(Wx)$, where $y$ is a *vector* of predictions, $W$ is the weight matrix, and $g$ is the vector valued link function, we present a simple second-order update rule. The update is based on a majorization of the Hessian, and uses a scaled version of the empirical second moment $\frac{1}{n}\sum_i x_i x_i^T$ as the preconditioner. Our algorithm is parameter-free and does not require a line search for convergence. Furthermore our computations only involve a $d \times d$ matrix unlike IRLS and other Hessian related approaches where matrices are $\mathcal{O}(dk \times dk)$ for multiclass problems[1]. Theoretically, the proposed method enjoys an iteration complexity independent of the condition number of the data matrix.

We extend our algorithm to simultaneously estimate the weights as well as the link function in GLMs under a parametric assumption on the link function, building on ideas from isotonic regression (Kalai & Sastry, 2009; Kakade et al., 2011). We provide a global convergence guarantee for this algorithm despite the non-convexity of the problem. Practically this enables, for example, the use of our current predictions as features to improve the predictions in subsequent iterations. Similar procedures are common for binary SVMs (Platt, 1999) and re-ranking(Collins & Koo, 2000).

Both the above algorithms are metric free but scale poorly with the dimensionality of the problem.Following ideas in the block-coordinate descent and stagewise regression literature, we generate batches of features (through projection or subsampling) and perform one of the above second-order updates on that batch only. We then repeat this process, successively fitting the residuals. In settings where the second order information is relevant, such as MNIST and CIFAR-10, we find that stagewise variants can be highly effective, providing orders of magnitude speed-ups over online methods and other first-order approaches. This is particularly noteworthy since we compare a simple MATLAB implementation of our algorithms with sophisticated C software for the alternative approaches. In contrast, for certain text problems where the data matrix is well conditioned, online methods are highly effective. Notably, we also achieve state of the art accuracy results on MNIST and CIFAR-10, outperforming the "dropout" neural net (Hinton et al., 2012), where our underlying optimization procedures are entirely based on simple least squares approaches. These promising results highlight that this is a fruitful avenue for the development of further theory and algorithms, which we leave for future work.

[1]This is a critical distinction as we focus on tasks involving increasingly complex class hierarchies, particularly in the context of computer vision problems.

**Related Work:** A large chunk of the work on large-scale optimization builds on and around online and stochastic optimization, leveraging the ability of these algorithms to ensure a very rapid initial reduction of test error (see e.g. (Bottou & Bousquet, 2008; Shalev-Shwartz, 2012)). These methods can be somewhat unsuited though for ill-conditioned problems, leading to recent works on hybrid methods (Shalev-Shwartz & Zhang, 2013; Roux et al., 2012). There has also been a renewed interest in Quasi-Newton methods scalable to statistical problems using stochastic approximation ideas (Byrd et al., 2011; Bordes et al., 2009). High-dimensional problems have also led to natural consideration of block coordinate descent style procedures, both in serial (Nesterov, 2012) and distributed (Richtárik & Takác, 2012; Recht et al., 2011) settings. Indeed, in some of our text experiments, our stagewise procedure comes quite close to a block-coordinate descent type update. There are also related approaches for training SVMs that extract the most information out of a small subset of data before moving to the next batch (Chapelle, 2007; Matsushima et al., 2012; Yu et al., 2012).

On the statistical side, our work most directly generalizes past works on learning in GLMs for binary classification, when the link function is known or unknown (Kalai & Sastry, 2009; Kakade et al., 2011; Friedman, 2001). In the statistics literature, the iteratively reweighed least squares algorithm (IRLS) is the workhorse for fitting GLMs and also works by recasting the optimization problem to a series of least squares problems. However, IRLS can diverge, while the proposed algorithms are guaranteed to make progress on each iteration. In IRLS (and some other majorization algorithms e.g., (Jebara & Choromanska, 2012)) each iteration needs to work with a new Hessian since it depends on the parameters. In contrast, our algorithms use the same matrix throughout their run.

## 2. Setting and Algorithms

We begin with the simple case of binary GLMs, before addressing the more challenging multi-class setting.

### 2.1. Warmup: Binary GLMs

The canonical definition of a GLM in binary classification (where $y \in \{0, 1\}$) setup posits the probabilistic model

$$\mathbb{E}[y \mid x] = g(w^{*T} x), \tag{1}$$

where $g : \mathbb{R} \mapsto \mathbb{R}$ is a monotone increasing function, and $w^* \in \mathbb{R}^d$. To facilitate the development of better algorithms, assume that $g$ is a $L$-Lipschitz function of its univariate argument. Since $g$ is a monotone increasing univariate function, there exists a convex function $\Phi : \mathbb{R} \mapsto \mathbb{R}$ such that $\Phi' = g$. Based on this convex function, let us define a convex loss function.

**Definition 1** (Calibrated loss)**.** *Given the GLM* (1)*, define the associated convex loss*

$$\ell(w; (x, y)) = \Phi(w^T x) - y w^T x. \tag{2}$$

Up to constants independent of $w$, this definition yields the least-squares loss for the identity link function, $g(u) = u$, and the logistic loss for the logit link function, $g(u) = e^u/(1 + e^u)$. The loss is termed calibrated: for each $x$, minimizing the above loss yields a consistent estimate of the weights $w^*$. Trivially:

$$\begin{aligned}\mathbb{E}[\nabla \ell(w^*; (x, y)) \mid x] &= \mathbb{E}[\nabla \Phi({w^*}^T x) - xy|x] \\ &\overset{(a)}{=} \mathbb{E}[g({w^*}^T x)x \mid x] - g({w^*}^T x)x = 0, \end{aligned} \tag{3}$$

where the equality (a) follows since $\Phi' = g$ and $\mathbb{E}[y \mid x] = g(w^* x)$ by the probabilistic model (1). Similar observations have been previously noted for the binary case (see Kakade et al. (2011)). Computing the optimal $w^*$ amounts to using a standard convex optimization procedure. We now discuss these choices for multi-class prediction.

### 2.2. Multi-class GLMs and Minimization Algorithms

The first question in the multi-class case concerns the definition of a generalized linear model; monotonicity is not immediately extended in the multi-class setting. Following the definition in the recent work of Agarwal (2013), we extend the binary case by defining the model:

$$\mathbb{E}[y \mid x] = \nabla \Phi(W^* x) := g(W^* x) \tag{4}$$

where $W^* \in \mathbb{R}^{k \times d}$ is the weight matrix, $\Phi : \mathbb{R}^k \mapsto \mathbb{R}$ is a proper and convex lower semicontinuous function of $k$ variables and $y \in \mathbb{R}^k$ is a vector with 1 for the correct class and zeros elsewhere. This definition essentially corresponds to the link function $g = \nabla \Phi$ satisfying (maximal and cyclical) monotonicity (Rockafellar, 1966).

This formulation immediately yields an analogous definition for a calibrated multi-class loss.

**Definition 2** (Calibrated multi-class loss)**.** *Given the GLM* (4)*, define the associated convex loss*

$$\ell(W; (x, y)) = \Phi(Wx) - y^T W x. \tag{5}$$

The loss function is convex as before and yields, for example, the multi-class logistic loss when the probabilistic model (4) is a multinomial logit model. It is Fisher consistent: the minimizer of the expected loss is $W^*$ (as in (3)).

As before, existing convex optimization algorithms can be utilized to estimate the weight matrix $W$. First-order methods applied to the problem have per-iteration complexity of $\mathcal{O}(dk)$, but can require a large number of iterations as discussed before. Here, the difficulty in utilizing second-order approaches is that the Hessian is of size $dk \times dk$ (e.g. as in IRLS); any direct matrix inversion method is now much more computationally expensive even for moderate $k$.

Algorithm 1 provides a simple variant of least squares regression — which repeatedly fits the residual error — that exploits the second order structure in $x$. The algorithm uses a block-diagonal upper bound on the Hessian matrix in order to preserve the correlations between the covariates $x$, but does not consider interactions across the different classes to have a more computationally tractable update. The algorithm has several attractive properties. Notably, (i) the algorithm is parameter free[2] and (ii) the algorithm only inverts a $d \times d$ matrix. Furthermore, this matrix is independent of the weights $W$ (and the labels) and can be computed once ahead of time. In that spirit, the algorithm can also be viewed as *preconditioned gradient descent*, with a block diagonal preconditioner whose diagonal blocks are all equal to the matrix $\widehat{\Sigma}^{-1}$. At each step, we utilize the residual error $\hat{\mathbb{E}}[(\hat{y} - y)x^T]$, akin to a gradient update on least-squares loss. Note the "stepsize" here is determined by $L$, a parameter entirely dependent on the loss function and not on the data. For the case of logistic regression, simply $L = 1$ satisfies this Lipchitz constraint. Also observe that for the square loss, where $L = 1$, the generalized least squares algorithm reduces to least squares (and terminates in one iteration).

We now describe the convergence properties of Algorithm 1. The results are stated in terms of the sample loss

$$\ell_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(W; (x_i, y_i)). \tag{7}$$

The following additional assumptions regarding the link function $\nabla \Phi$ are natural for characterizing convergence rates. Assuming that the link function $g = \nabla \Phi$ is $L$-Lipschitz amounts to the condition

$$\|g(u) - g(v)\|_2 \leq L \|u - v\|_2, \text{ where } u, v \in \mathbb{R}^k. \tag{8}$$

If we want a linear convergence rate, we must further assume $\mu$-strong monotonicity, meaning for all $u, v \in \mathbb{R}^k$:

$$\langle g(u) - g(v), u - v \rangle \geq \mu \|u - v\|_2^2. \tag{9}$$

**Theorem 1.** *Define* $W^* = \arg\min_W \ell_n(W)$. *Suppose that the link function* $\nabla \Phi$ *is L-Lipschitz* (8)*. Using the generalized Least Squares updates (Algorithm 1) with* $W_0 = 0$*,*

[2]Here and below we refer to parameter free algorithms from the point of view of optimization: no learning rates, backtracking constants etc. The overall learning algorithms may still require setting other parameters, such as the regularizer.

**Algorithm 1** Generalized Least Squares

**Input:** Initial weight matrix $W_0$, data $\{(x_i, y_i)\}$, Lipschitz constant $L$, link $g = \nabla\Phi$.

Define the (vector valued) predictions $\hat{y}_i^{(t)} = g(W_t x_i)$ and the empirical expectations:

$$\widehat{\Sigma} = \hat{\mathbb{E}}[x_i x_i^T] = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$$

$$\hat{\mathbb{E}}[(\hat{y}^{(t)} - y)x^T] = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i^{(t)} - y_i)x_i^T$$

**repeat**

Update the weight matrix $W_t$:

$$W_{t+1}^T = W_t^T - \frac{1}{L}\,\widehat{\Sigma}^{-1}\,\hat{\mathbb{E}}[(\hat{y}^{(t)} - y)x^T] \qquad (6)$$

**until** convergence

*then for all* $t = 1, 2, \ldots$

$$\ell_n(W_t) - \ell_n(W^*) \le \frac{2L\|W^*\|^2}{t+4}.$$

*If, in addition, the link function is* $\mu$*-strongly monotone* (9) *and let* $\kappa_\Phi = L/\mu$*. Then*

$$\ell_n(W_t) - \ell_n(W^*) \le \frac{L}{2}\left(\frac{\kappa_\Phi - 1}{\kappa_\Phi + 1}\right)^t \|W^*\|_F^2.$$

The proof rests on demonstrating that the block-diagonal matrix formed by copies of $L\widehat{\Sigma}$ provides a majorization of the Hessian matrix, along with standard results in convex optimization (see e.g. (Nesterov, 2004)) and is deferred to the long version (Agarwal et al., 2013). Also, observe that the convergence results in Theorem 1 are completely independent of the conditioning of the data matrix $\widehat{\Sigma}$. Indeed they depend only on the smoothness and strong convexity properties of $\Phi$ which is a function we know ahead of time and control. This is the primary benefit of these updates over first-order updates.

In order to understand these issues better, let us quickly contrast these results to the analogous ones for gradient descent. In that case, we get qualitatively similar dependence on the number of iterations. However, in the case of Lipschitz $\nabla\Phi$, the convergence rate is $\mathcal{O}\left(\frac{L}{t}\sigma_{\max}\left(\frac{XX^T}{n}\right)\|W^*\|^2\right)$. Under strong monotonicity, the rate improves to $\mathcal{O}\left(L\left(\frac{\kappa_\Phi\kappa_{XX^T}-1}{\kappa_\Phi\kappa_{XX^T}+1}\right)^t\right)$. That is, the convergence rate is slowed down by factors depending on the singular values of the empirical covariance in both the cases. Similar comparisons can also be made for accelerated versions of both our and vanilla gradient methods.

**Algorithm 2** Calibrated Least Squares

**Input:** Initial weight matrix $W_0$, set of calibration functions $G = \{g_1, \ldots g_m\}$ $(g_i : \mathbb{R}^k \to \mathbb{R}^k)$

Initialize the predictions: $\hat{y}_i^{(0)} = W_0 x_i$

**repeat**

Fit the residual:

$$W_t = \arg\min_W \sum_{i=1}^{n} \|y_i - \hat{y}_i^{(t-1)} - Wx_i\|_2^2, \quad \text{and}$$

$$\tilde{y}_i^{(t)} = \hat{y}_i^{(t-1)} + W_t x_i. \qquad (10)$$

Calibrate $\tilde{y}^{(t)}$: Define $u_i = [g_1(\tilde{y}_i^{(t)}), \ldots, g_m(\tilde{y}_i^{(t)})]^\top$

$$\tilde{W}_t = \arg\min_{\tilde{W}\in\mathbb{R}^{k\times mk}} \sum_{i=1}^{n} \|y_i - \tilde{W}u_i\|_2^2, \quad \text{and}$$

$$\hat{y}_i^{(t)} = \text{clip}(\tilde{W}_t u_i), \qquad (11)$$

where clip$(v)$ is the Euclidean projection of $v$ onto the probability simplex in $\mathbb{R}^k$.

**until** convergence

### 2.3. Unknown Link Function for Multi-class

The more challenging case is when the link function is unknown. Our approach will generalize the Isotron algorithm for the binary case (Kalai & Sastry, 2009). In particular we will iterate between fitting $y$ and finding the best link function to calibrate our predictions. This raises two main difficulties: the statistical one of how to restrict the complexity of the class of link functions and the computational one of ensuring that this procedure convergences globally.

With regards to the former, a natural restriction is to consider the class of link functions realized as the derivative of a convex function in $k$-dimensions. This naturally extends Isotron but, unfortunately, this is an extremely rich class; the sample complexity of estimating a uniformly bounded convex, Lipschitz function in $k$ dimensions grows exponentially with $k$ (Bronshtein, 1976). To avoid this curse of dimensionality, assume that there is a finite basis $G$ such that $g^{-1} = (\nabla\Phi)^{-1} \in \text{lin}(G)$, $(\nabla\Phi)^{-1}$ is the functional inverse of $\nabla\Phi$. Without loss of generality, we also assume that $G$ always contains the identity function. We do not consider the issue of approximation error here.

Before presenting the algorithm, let us provide some more intuition about our assumption $g^{-1} = (\nabla\Phi)^{-1} \in \text{lin}(G)$. Clearly the case of $G = g^{-1}$ for a fixed function $g$ puts us in the setting of the previous section. More generally, let us consider that $G$ is a dictionary of $p$ functions so that $g^{-1}(y) = \sum_{i=1}^{p} \widetilde{w}_i G_i(y)$, where $G_i : \mathbb{R}^k \mapsto \mathbb{R}^k$. In the

GLM (4), this yields an overall linear-like model[3]

$$\sum_{i=1}^{p} \widetilde{W}_i G_i(\mathbb{E}[Y|x]) = W^* x.$$

If we let $p = k$ and $G_i(y)$ be the $i_{th}$ class indicator $y_i$, then the above equation boils down to

$$\widetilde{W}^T \mathbb{E}[Y|x] = W^* x, \tag{12}$$

meaning that an unknown linear combination of the class-conditional probabilities is a linear function of the data. More generally, we consider $G_i$ to also have higher-order monomials such as $y_i^2$ or $y_i^3$ so that the LHS is some low-degree polynomial of the class-conditional probability with unknown coefficients.

Now, the computational issue is to efficiently form accurate predictions (as in the binary case (Kalai & Sastry, 2009), the problem is not convex). We now describe a simple strategy for simultaneously learning the weights as well as the link function, which not only improves the square loss at every step, but also converges to the optimal answer, and empirically in very few steps. The strategy maintains two sets of weights, $W_t \in \mathbb{R}^{k \times d}$ and $\tilde{W}_t \in \mathbb{R}^{k \times \dim(G)}$ and maintains our current predictions $\hat{y}_i^{(t)} \in \mathbb{R}^k$ for each data point $i = 1, 2, \ldots, n$. After initializing all the predictions and weights to zero, the updates shown in Algorithm 2 involve two alternating least squares steps. The first step fits the residual error to $x$ using the weights $W_t$. The second step then fits $y$ to functions of updated predictions, i.e. to $G(\tilde{y}^{(t)})$. Finally, we project onto the unit simplex to obtain the new predictions, which can only decrease the squared error and can be done in $O(k)$ time (Duchi et al., 2008).

In the context of the examples of $G_i$ mentioned above, the algorithm boils down to predicting the conditional probability of $Y = i$ given $x$, based not only on $x$, but also on our current predictions for all the classes (and higher degree polynomials in these predictions).

For the analysis of Algorithm 2, we focus on the noiseless case to understand the optimization issues. Analyzing the statistical issues, where there is noise, can be handled using ideas in (Kalai & Sastry, 2009; Kakade et al., 2011).

**Theorem 2.** *Suppose that $y_i = g(W^* x_i)$ and that the link function $g = \nabla\Phi$ satisfies the Lipschitz and strong monotonicity conditions* (8) *and* (9) *with constants $L$ and $\mu$ respectively. Suppose also that $\nabla\Phi(0) = \mathbb{1}/k$. Using the (calibrated) Least Squares updates (Algorithm 2) with $W_0 = 0$, for all $t = 1, 2, \ldots$ we have the bound*

$$\frac{1}{n} \sum_{i=1}^{n} \|\hat{y}_i^{(t)} - y_i\|_2^2 \leq \frac{22 k \kappa_\Phi^2}{t}.$$

[3]It is not a linear model since the statistical noise passes through the functions $G_i$ rather than being additive.

**Algorithm 3** Stagewise Regression

**Input:** data $\{(x_i, y_i)\}$, batch generator GEN, batch size $p$, iterations $T$

Initialize predictions: $\hat{y}_i^{(1)} = 0$
**for** $t = 1, \ldots, T$ **do**
  Generate $p$ features from the original ones

$$\{\tilde{x}_i\} = \text{GEN}(\{x_i\}, p)$$

  Run Algorithm 1 or 2 on $\{(\tilde{x}_i, y_i - \hat{y}_i^{(t)})\}$ and get $W_t$
  Update predictions: $\hat{y}_i^{(t+1)} = \hat{y}_i^{(t)} + W_t \tilde{x}_i$
**end for**

We again emphasize the fact that the updates (10) and (11) only require the solution of least-squares problems in a similar spirit as Algorithm 1. Finally, we note that the rules to compute predictions in our updates( (10) and (11)) require previous predictions (i.e. the learned model is not *proper* in that it does not actually estimate $g$, yet it is still guaranteed to make accurate predictions).

### 2.4. A Scalable Wrapper Algorithm

When the number of features is large, any optimization algorithm that scales superlinearly with the problem dimension faces serious computational issues. Here we propose a simple way to scale the previous algorithms to high dimensional problems. We adopt a block coordinate descent style approach. To keep the presentation fairly general, we assume that we have an algorithm GEN that returns a small set of $m$ features, where $m$ is small enough so that least squares fitting with $m$ features is efficient (e.g. we typically use $m \approx 1000$). The GEN procedure can be as simple as sampling $m$ of the original features (with or without replacement) or other schemes such as random Fourier features (Rahimi & Recht, 2007). We call GEN and fit a model on the $m$ features using either Algorithm 1 or Algorithm 2. We then compute residuals and repeat the process on a fresh batch of $m$ features returned by GEN. In Algorithm 3 we provide pseudocode for this stagewise regression procedure. We stress that this algorithm is purely a computational convenience. It can be thought as the algorithm that would result by a block-diagonal approximation of the second moment matrix $\Sigma$ (not just across classes, but also groups of features). Algorithm 3 bears some resemblance to boosting and related coordinate descent methods, with the crucial difference that GEN is not restricted to searching for the best set of features. Indeed, in our experiments GEN is either sampling from the features without replacement or randomly projecting the data in $m$ dimensions and transforming each of the $m$ dimension by a simple non-linearity. Despite its simplicity, more work needs to be done to theoretically understand the properties of this

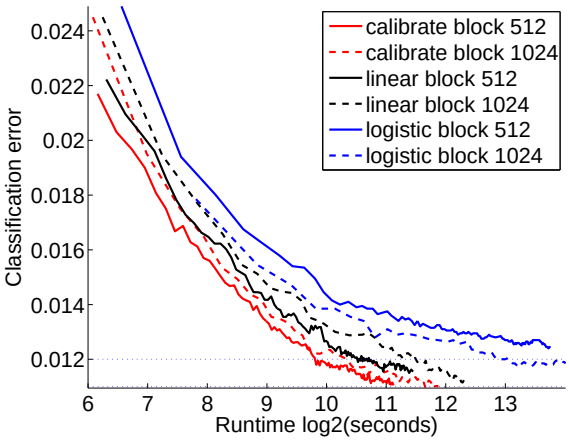



*Figure 1.* Runtime versus test error for different algorithms

variant as clearly as those of Algorithm 1 or Algorithm 2. Practically, stagewise regression can have useful regularization properties but these can be subtle and greatly depend on the GEN procedure. In text classification, for example, fitting the most frequent words first leads to better models than fitting the least frequent words first.

## 3. Experiments

We consider four datasets MNIST, CIFAR-10, 20 Newsgroups, and RCV1 that capture many of the challenges encountered in real-world learning tasks. We believe that the lessons gleaned from our analysis and comparisons of performance on these datasets apply more broadly.

For MNIST, we compare our algorithms with a variety of standard algorithms. Both in terms of classification accuracy and optimization speed, we achieve close to state of the art performance among permutation-invariant methods (1.1% accuracy, improving upon methods such as the "dropout" neural net). For CIFAR-10, we also obtain nearly state of the art accuracy ($> 85\%$) using standard features. Here, we emphasize that it is the computational efficiency of our algorithms which enables us to achieve higher accuracy without novel feature-generation.

The story is rather different for the two text datasets, where the performance of our methods is less competitive with online approaches, though we do demonstrate substantial reduction in error rate in one of the problems. As mentioned in the introduction, second order learning is less appealing for well conditioned datasets such as text.

### 3.1. MNIST

Nonlinear classifiers are needed to achieve state-of-the-art performance in MNIST dataset. Although it only contains 60K data points, the requirement for nonlinearity makes this dataset computationally challenging. For instance, a nonlinear support vector machine with a Gaussian RBF kernel needs to manipulate a 60K×60K kernel matrix; requiring substantial computation and memory in most modern desktop machines. Hence we use an explicit feature representation and train our classifiers in the primal space. Specifically we construct random fourier features which are known to approximate the Gaussian kernel $k(x, x') = \exp(-\|x - x'\|^2/s)$ (Rahimi & Recht, 2007) (more details in the long version (Agarwal et al., 2013)).

We start by comparing linear regression with Algorithm 1 (Logistic), and Algorithm 2 (Calibration). For Calibration, we use a basis $G(y)$ consisting of $y$, $y^2$ and $y^3$ (applied elementwise to the vector $y$). We compare these algorithms on raw pixel features, as well as small number of random Fourier features described above. As seen in Table 1, the performance of Logistic and Calibration seems similar and consistently superior to plain linear regression.

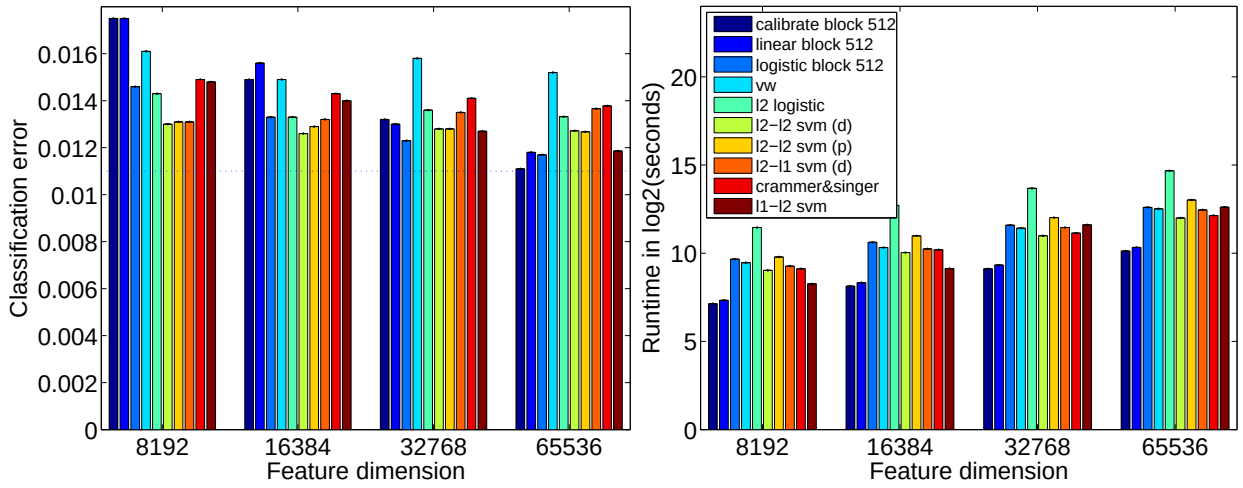
Next, we move to improve accuracy by using Algorithm 3, which allows us to scale up to larger number of random Fourier features. Concretely, we fit blocks of features (either 512 and 1024) with Algorithm 3 with three alternative update rules on each stage: linear regression, Calibration, and Logistic (50 inner loop iterations). Our calibrated variant again uses the functions $y$, $y^2$ and $y^3$ of previous predictions as additional features in our new batch of features.

Our next experiment demonstrates that all three (extremely simple and parameter free) algorithms quickly achieve state of the art performance. Figure 1 shows the relation between feature block size, classification test error, and runtime for these algorithm variants. Importantly, while the linear and Calibration algorithms do not achieve as low an error for a fixed feature size (cf. Table 1), they are faster to optimize and are more effective overall. Recall that we used 50 inner loop iterations for Logistic hence it appears to not make as much progress as the other methods. Given enough time however, Logistic can reach a test error of 1.1% (not shown). In general we find that linear regression and relatively small size feature batches achieves better runtime and error trade-off than logistic regression while Calibration further improves upon these results.

In our next experiment, we compare the previous three variants to other state-of-the-art algorithms in terms of classification test error and runtime (Figure 2(a) and (b)). We used the default convergence criteria for these other implementations. Perhaps, given enough time, they could also reach

*Table 1.* Linear Regression vs. logistic regression vs. (polynomial) calibration. For the polynomial calibration, we refit our predictions with $\hat{y}$, $\hat{y}^2$ and $\hat{y}^3$.

| Algorithm | Linear | Logistic | (poly.) Calibration |
|---|---|---|---|
| Raw pixels | 14.1% | 7.8% | 8.1% |
| 4000 dims | 1.83 % | 1.48 % | 1.54 % |
| 8000 dims | 1.48% | 1.33% | 1.36% |




(a) Comparable Errors (b) Time Comparisons

*Figure 2.* (a) Error comparison between our variants, VW and six variants of Liblinear: L2-regularized logistic regression, L2-regularized L2-loss support vector classification (dual), L2-regularized L2-loss support vector classification (primal), L2-regularized L1-loss support vector classification (dual), multi-class support vector classification by Crammer and Singer, L1-regularized L2-loss support vector classification. Dotted line is 1.1% test error. For competing methods, we picked best value of regularization parameter. (b) Runtime comparison in **log** time.

the same quality solutions that our methods achieve, but they seem to have been tuned for well-conditioned problems. The comparison includes VW, and six algorithms implemented in Liblinear (Fan et al., 2008) (see figure caption). We searched for regularization paremeters. We timed these algorithms to measure their computation time, rather than their loading of the features (which can be rather large, making it very time consuming to run these experiments); our stagewise algorithms generate new features on the fly so this is not an issue (further discussion in the long version).

From Figure 2 (a) and (b), Logistic is competitive with all the other algorithms, in terms of its error (while for lower dimensions linear and Calibration fared a little worse). In comparison to VW and Liblinear, the generalization error of our methods drops quickly as more features are added. We suspect that more features make the conditioning of the data worse and the first order methods have to give up almost all of the improvement in approximation error as excess optimization error. This raises an interesting issue as sometimes these methods are believed to be "exact" (in contrast to our stagewise approach which is obviously an approximate optimizer). In practice, bad enough conditioning can completely defeat first order methods, which may be forced to terminate early based on lack of progress. Finally, all of our algorithms were substantially faster. (Note the *logarithmic* scaling of the runtime axes).

In conclusion, our methods produce models with a test error of 1.1% while none of the competitors achieve this test error (only L1-L2 SVM comes close) Runtime wise, the stagewise linear and Calibration variants are extremely fast, consistently at least 10 times faster than the other highly optimized algorithms. This is particularly notable given the simplicity of this approach.

### 3.2. CIFAR-10

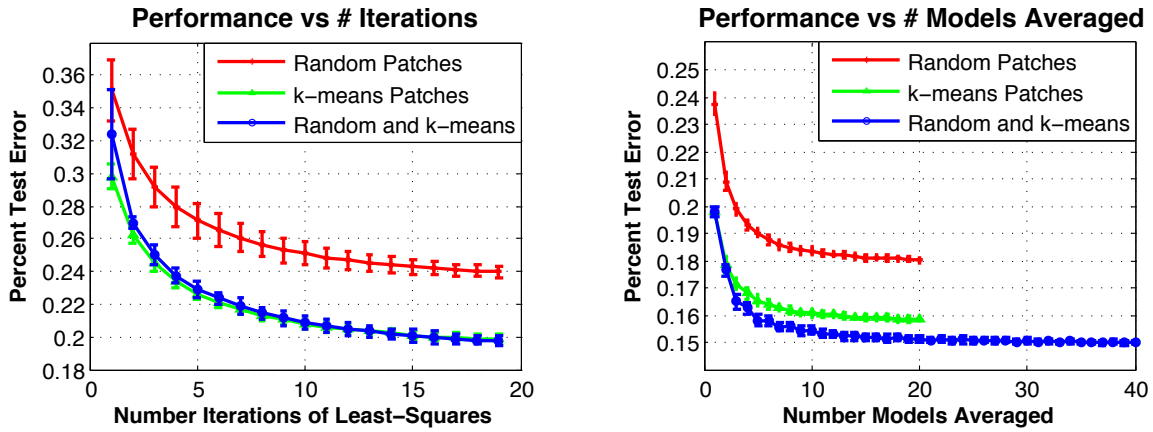
The CIFAR-10 dataset is more challenging and many image recognition algorithms have been tested (primarily illustrating different methods of feature generation; our work instead focuses on the optimization component). Algorithms such as the "dropout" and "maxout" (Hinton et al., 2012; Goodfellow et al., 2013) achieve accuracies of 84% and 87% without dataset augmentation (through jitter or other transformations). We are able to robustly achieve over 85% accuracy with linear regression on standard convolution features without dataset augmentation.

Figure 3 illustrates the performance when we use two types of features: convolutions with random masks, or K-means masks (as in (Coates et al., 2011), though we do *not* use contrast normalization). The induced representations are ill-conditioned and fitting them with Liblinear is extremely slow. Our methods are simple and easy to embed in a tight loop together with this (or any) feature generation.

We find that using only about 400 filters, along with polynomial features, is sufficient to obtain over 80% accuracy very quickly. Hence, using thousands of generated features, it is rather fast to build multiple models with disjoint features and model average them, obtaining 85% accuracy.

### 3.3. Well-Conditioned Problems

We close by observing that in certain cases, it might be wasteful to employ our methods. To illustrate, we exam-




(a) (b)

*Figure 3.* CIFAR-10 results using two types of convolutional features random masks and masks derived from K-means masks. (a) Generalization vs least square iterations. (b) Generalization vs number of models averaged.

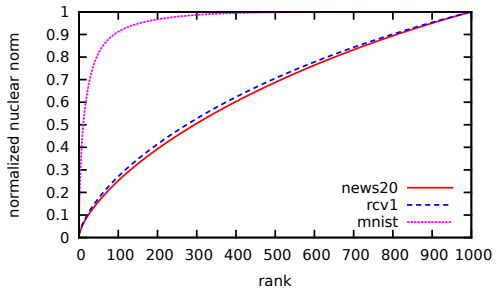



*Figure 4.* The fraction of the sum of the top 1000 singular values that is captured by the top $x$ singular values.

ine two popular multiclass text datasets: 20 newsgroups[4] (henceforth NEWS20), which is a 20 class dataset and a four class version of Reuters Corpus Volume 1 (Lewis et al., 2004) (henceforth RCV1). We use a (log) term frequency representation of the data (more details in the long version). These data are sparse and very well conditioned: The ratio of the 2nd singular value to the 1000th one (as a proxy for the condition number) is 19.8 for NEWS20 and 14 for RCV1. In contrast, for MNIST, this number is about 72000 (computed with 3000 random Fourier features). Figure 4 shows the respective normalized spectra.

As expected, online first-order methods (VW) fare far more favorably in this setting, as seen in Table 2. We use a simple greedy procedure for our stagewise ordering (as discussed in the long version, though random also works well). Note that this data is well suited for online methods: it is sparse (making the updates cheap) and well conditioned (making the gap in convergence between first and second order methods small). Developing hybrid approaches applicable to both cases is an interesting direction.

[4]http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

*Table 2.* Running times and test errors in text datasets for VW, Liblinear, and Stagewise regression

| | NEWS20 | | RCV1 | |
|---|---|---|---|---|
| Method | Time | %Error | Time | %Error |
| VW | 2.5 | 12.4 | 2.5 | 2.75 |
| Liblinear | 27 | 13.8 | 120 | 2.73 |
| Stagewise | 40 | 11.7 | 240 | 2.77 |

## 4. Discussion

We presented a suite of fast and simple algorithms for tackling large-scale multiclass prediction problems. We stress that the key upshot of the methods developed in this work is their conceptual simplicity and ease of implementation. Indeed these properties make the methods quite versatile and easy to extend in various ways. We showed an instance of this in Algorithm 2. Similarly, it is straightforward to develop accelerated variants (Nesterov, 2004), by using the distances defined by the matrix $\widehat{\Sigma}$ as the prox-function in Nesterov's work. These variants enjoy the usual improvements of $\mathcal{O}(1/t^2)$ iteration complexity in the smooth and $\sqrt{\kappa_\Phi}$ dependence in the strongly convex setting, while retaining the metric-free nature of Algorithm 1.

It is also easy to extend the algorithms to multi-label settings, with the only difference being that the vector $y$ now lives on the hypercube instead of the simplex. This amounts to a minor modification of (11) in Algorithm 2.

Overall, we believe that our approach revisits many old and deep ideas to develop algorithms that are practically very effective. We believe that it will be quite fruitful to understand these methods better both theoretically and empirically in further research.

**Acknowledgement**

L. Song was supported by NSF/NIH BIGDATA 1R01GM108341-01, NSF IIS1116886, and a Raytheon faculty fellowship.

## References


Agarwal, A. Selective sampling algorithms for cost-sensitive multiclass prediction. In *ICML*, 2013.

Agarwal, A., Kakade, S. M., Karampatziakis, N., Song, L., and Valiant, G. Least squares revisited: Scalable approaches for multi-class prediction. *arXiv preprint arXiv:1310.1949*, 2013.

Bordes, A., Bottou, L., and Gallinari, P. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, July 2009.

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *NIPS*. 2008.

Bronshtein, E.M. $\epsilon$-entropy of convex sets and functions. *Siberian Mathematical Journal*, 17(3):393–398, 1976.

Byrd, R. H., Chin, G. M., Neveitt, W., and Nocedal, J. On the use of stochastic hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3): 977–995, 2011.

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

Coates, A., Ng, A. Y., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011.

Collins, M. and Koo, T. Discriminative reranking for natural language parsing. In *ICML*, 2000.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the $\ell_1$-ball for learning in high dimensions. In *ICML*, 2008.

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

Friedman, J. H. Greedy function approximation: a gradient boosting machine.(english summary). *Ann. Statist*, 29(5):1189–1232, 2001.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. Maxout networks. *CoRR*, 2013.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Jebara, T. and Choromanska, A. Majorization for crfs and latent likelihoods. In *NIPS*, 2012.

Kakade, S. M., Kalai, A., Kanade, V., and Shamir, O. Efficient learning of generalized linear and single index models with isotonic regression. In *NIPS*, 2011.

Kalai, A. T. and Sastry, R. The isotron algorithm: High-dimensional isotonic regression. In *COLT '09*, 2009.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

Matsushima, S., Vishwanathan, S. V. N., and Smola, A. J. Linear support vector machines via dual cached loops. In *KDD*, 2012.

Nemirovsky, A. S. and Yudin, D. B. *Problem Complexity and Method Efficiency in Optimization*. New York, 1983.

Nesterov, Y. *Introductory Lectures on Convex Optimization*. New York, 2004.

Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Adavances in large margin classifiers*, pp. 61–74. MIT Press, 1999.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20:1177–1184, 2007.

Recht, B., Re, C., Wright, S. J., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pp. 693–701, 2011.

Richtárik, P. and Takác, M. Parallel coordinate descent methods for big data optimization. 2012. URL `http://arxiv.org/abs/1212.0873`.

Rockafellar, R.T. Characterization of the subdifferentials of convex functions. *Pac. J. Math.*, 17:497–510, 1966.

Roux, N. L., Schmidt, M., and Bach, F. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, pp. 2672–2680. 2012.

Shalev-Shwartz, S. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2), 2012.

Shalev-Shwartz, S. and Zhang, T. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization. *Journal of Machine Learning Reearch*, 14:567–599, 2013.

Yu, H.-F., Hsieh, C.-J., Chang, K.-W., and Lin, C.-J. Large linear classification when data cannot fit in memory. *TKDD*, 5(4), 2012.