Large-Scale Sparse Kernel Logistic Regression

— with a comparative study on optimization algorithms

Zhiwei (Tony) Qin
Columbia University
New York, NY
zq2107@columbia.edu

Bo Huang
Columbia University
New York, NY
bh2359@columbia.edu

Shyam S. Chandramouli
Columbia University
New York, NY
sc3102@columbia.edu

Junfeng He
Columbia University
New York, NY
jh2700@columbia.edu

Sanjiv Kumar
Google Research
New York, NY
sanjivk@google.com

ABSTRACT

Kernel Logistic Regression (KLR) is a powerful probabilistic classification tool, but its training and testing both suffer from severe computational bottlenecks when used with large-scale data. Traditionally, L1-penalty is used to induce sparseness in the parameter space for fast testing. However, most of the existing optimization methods for training l_1-penalized KLR do not scale well in large-scale settings. In this work, we present highly scalable training of KLR model via three first-order optimization methods: Fast Shrinkage Thresholding Algorithm(FISTA), Coordinate Gradient Descent (CGD), and a variant of Stochastic Gradient Descent (SGD) method. To further reduce the space and time complexity, we apply a simple kernel linearization technique which achieves similar results at a fraction of the computational cost. While SGD appears the fastest in training large-scale data, we show that CGD performs considerably better in some cases on various quality measures. Based on this observation, we propose a multi-scale extension of FISTA which improves its computational performance significantly in practice while preserving the theoretical global convergence rate. We further propose a two-stage active set training scheme for CGD and FISTA, which boosts the prediction accuracies by up to 4%. Extensive experiments on several data sets containing up to millions of samples demonstrate the effectiveness of our approach.

Categories and Subject Descriptors

1 [**Algorithms/Models**]: Classification

Keywords

large-scale classification, kernel logistic regression, kernel linearization, L1-regularization, first-order optimization methods, two-stage active set training

1. INTRODUCTION

1.1 Kernel Logistic Regression

Kernel Logistic Regression (KLR) [25, 30] is a powerful probabilistic classification tool [7]. Given N training points, the following minimization is used to train KLR:

$$\min_w F(w) = -\sum_{i=1}^{N} \log(\sigma(y_i w^T k_i)) \tag{1}$$

where k_i is the i-th column of the kernel matrix K of the training data, and $\sigma(v) = 1/(1+\exp^{-v})$. The possibility of using different kernels allows one to learn a nonlinear classifier in the feature space.

However, its training and testing both suffer from severe computational bottlenecks when used with large-scale data. Moreover, when the number of training data is small compared to the number of the features, straightforward KLR training may lead to over-fitting. KLR with L1-regularization in which an extra $\lambda\|\cdot\|_1$ is added to penalize large "w" has received considerable attention since it enforces sparsity in w. Specifically, L1-regularized KLR often yields a sparse solution w whose nonzero components correspond to the underlying more "meaningful" features. Therefore, it greatly saves the testing time especially when the feature dimension is huge for only a few features are used in computation. Indeed, L1-regularized KLR makes it possible to learn and apply KLR at large scale with similar performance as of L2-regularization. In [16, 26], it is shown that L1-regularization can outperform L2-regularization especially when number of observations is smaller than the number of features. Thus, L1-regularization technique has been widely used in many other problems, such as compressed sensing [3, 4] and Lasso [20], despite the fact that it is more challenging to solve L1-regularization than L2-regularization due to the non-smoothness of $\|\cdot\|_1$. In this paper, we consider the following Sparse Kernel Logistic Regression (SKLR) model.

$$\min_w F(w) = -\sum_{i=1}^{N} \log(\sigma(y_i w^T k_i)) + \lambda\|w\|_1 \tag{2}$$

We choose Radial Basis Function (RBF) kernel to form the Kernel matrix K, i.e.,

$$k(i,j) = \exp^{-\frac{\|x_i - x_j\|^2}{\sigma^2}} \tag{3}$$

1.2 Kernel linearization

Non-linear kernel classifiers are attractive because they can approximate the decision boundary better given enough training data. Unfortunately, methods that operate on the nonlinear kernel matrix scale poorly with the size of the training data, thus unsuitable for the large-scale computation. In [19], the advantages of the linear and nonlinear approaches are combined using the "Fourier Kernel Linearization". In this paper, we will follow this line to speed up the method and make large-scale computation possible.

1.3 Algorithms for L1-regularization

Various first-order optimization algorithms have been proposed to solve the L1-regularized problems. In this paper, we explore three algorithms, (Fast) Iterative Shrinkage-Thresholding algorithm (FISTA/ISTA) [1], the Coordinate Gradient Descent algorithm (CGD) [22] and the Stochastic gradient descent (SGD) [2, 23]. FISTA is an accelerated proximal-gradient algorithm which was originally proposed to solve the linear inverse problem arising in image processing. In [13, 15], it has been proven that its complexity bound is optimal among all the first-order methods. CGD is a coordinate-descent type of algorithm. [29] extended the former work to L1-regularized convex minimization problems which achieves the aforementioned goals. [18] and [12] applied a block-coordinate version of CGD to solve the Group Lasso problems with least squares and logistic regression respectively. SGD uses approximate gradients from subsets of the training data and updates the parameters in an online fashion. In many applications this results in less training time in practice than batch training algorithms. The SGD-C method that we use in our study is based on the Stochastic sub-Gradient Descent algorithm [23] originally proposed to solve the L1-regularized log-linear models. It uses the cumulative penalty heuristic to improve the sparsity of w. SGD-C also incorporates the gradient-averaging idea from the dual averaging algorithm [27], which extends Nesterov's work [14] to L1-regularized kernel logistic regression.

1.4 Our contribution

In this paper, we explore and analyze three algorithms FISTA, CGD and SGD for solving the L1-regularized large-scale KLR, which has never been performed to the best of our knowledge. We observe that CGD performs surprisingly better than FISTA in the number of iterations to convergence. For large-scale data with size up to millions, SGD appears faster in terms of the training time, but it comes with the loss of optimality guarantee upon termination, and SGD has lower prediction accuracy or sparsity compared to the deterministic methods (i.e. FISTA and CGD) in some cases. We also study the effect of various values of the regularization parameter on the training time, prediction accuracy, and sparsity. In the algorithmic aspect, we propose a two-stage active-set training approach which can boost the prediction accuracy by up to 4% for the deterministic algorithms. Based on the observation that CGD converges faster than FISTA on SKLR problems, we adopt the feature of CGD and propose a multiple-scaled FISTA which improves its performance significantly while preserving the theoretical global convergence rate. We have also applied the gradient-averaging technique in [27, 14, 28] and the cumulative penalty heuristic in [23] to SGD to make it well-suited for large-scale SKLR training.

The subsequent sections of our paper are organized as follows. In Section 2, we discuss the technical details of the three optimization algorithms and the relevant extensions to them. We then talk about kernel linearization and the two-stage active set training in Section 3. The experiment results on various data sets are presented in Section 4. We end the paper with some concluding remarks.

2. OPTIMIZATION ALGORITHMS

2.1 Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

To solve the L1-regularized KLR, we need to minimize the sum of two convex functions

$$\min_x \{F(x) \equiv f(x) + g(x) : x \in R^m\} \tag{4}$$

where $g(x) = \|x\|_1$ is a continuous convex function and $f(x) = -\sum_{i=1}^{N} \log(\sigma(y_i w^T k_i))$ is a smooth function with Lipschitz continuous gradient such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L(f)\|x - y\|$$

where $L(f) > 0$ is the Lipschitz constant of ∇f but it is often expensive to compute $L(f)$.

In [1], Beck and Teboulle solve $F(x)$ with $g(x)$ unchanged and $f(x)$ replaced by its quadratic approximation, i.e,

$$\begin{aligned} Q_\mu(x; x_k) &:= f(y) + \langle x - x_k, \nabla f(x_k)\rangle + \frac{1}{2\mu}\|x - x_k\|^2 + g(x) \\ p_\mu(x_k) &= \arg\min_x Q_\mu(x; x_k) \end{aligned}$$

However, we can extend the original FISTA and make a better quadratic approximation by introducing the approximate Hessian matrix H_k. It is not surprising that it will result in better computational performance, and the improvement highly depends on the quality of H_k. More specifically,

$$\begin{aligned} Q_\mu(x; x_k) &:= f(y) + \langle x - x_k, \nabla f(x_k)\rangle \\ &+ \frac{1}{2\mu}(x - x_k)^T H_k (x - x_k) + \lambda\|x\|_1 + g(x), \\ p_\mu(x_k) &= \arg\min_x Q_\mu(x; x_k) \end{aligned}$$

where $p_\mu(x_k)$ has the closed form when H_k is a diagonal matrix, and its ith component is computed as followed

$$p_\mu(x_k)^i = \max(|z^i - \frac{\lambda\mu}{H_k^{ii}}, 0|, 0) \cdot \mathrm{sgn}(z^i) \tag{5}$$

where H_k^{ii} is the ith diagonal element of H_k, and z^i is defined as

$$z^i = x_k^i - \frac{\mu}{H_k^{ii}}(\nabla f(x))^i.$$

The diagonal approximated Hessian can be chosen in the following ways:

$$H_k = I \tag{6}$$

$$H_k^{ii} = \min(\frac{\partial^2 f(x_k)}{\partial (x_k^j)^2}, H_{k-1}^{ii}) \tag{7}$$

$$H_k^{ii} = \frac{\partial^2 f(x_k)}{\partial (x_k^j)^2} \tag{8}$$

Note that the original FISTA proposed in [1] simply uses (6). We propose to use (7) or (8) to extend FISTA to Multiple-scale FISTA. It is so named because the choice of H_k is

equivalent to choosing a different and tailored step-length for each coordinate (feature). The Multiple-scale FISTA performs much better than the original FISTA as shown in Section 4.3 Table 7 since it carries more information of the second order derivative. In order to make our convergence

Algorithm 1 Multiple-scale FISTA (with backtracking)

1: Given $\mu > 0$, some $\eta < 1$, and $y_0 \in R^n$. Set $x_1 = y_0,\ t_1 = 1$.
2: **for** $k = 1, 2, \cdots$ **do**
3: Find the smallest nonnegative integers i_k such that with $\bar{\mu} = \eta^{i_k}\mu$

$$F(p_{\bar{\mu}}(x_k)) \leq Q_{\bar{\mu}}(p_{\bar{\mu}}(x_k), x_k).$$

4: Set $\mu \leftarrow \eta^{i_k}\mu$ and compute

$$y_k \leftarrow p_\mu(x_k)$$
$$t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$$
$$x_{k+1} \leftarrow y_k + (\frac{t_k - 1}{t_{k+1}})(y_k - y_{k-1})$$

5: **end for**

hold, the following conditions are critical.

$$F(p_{\bar{\mu}}(x_k)) \leq Q_{\bar{\mu}}(p_{\bar{\mu}}(x_k), x_k). \quad (9)$$
$$H_{k-1} - H_k \succ 0 \quad (10)$$

Condition (9) indicates that the quadratic approximation lies above the true function, which implies that the true objective value will decrease even more while we minimize the quadratic approximation. Condition (10) is also necessary for the proof of the following theorem, and it is satisfied by first two choices of H_k.

THEOREM 1. *Suppose the sequences $\{x_k\}$ and $\{y_k\}$ are generated by FISTA with $H_{k-1} - H_k \succ 0$ for $k = 1, 2\cdots$ and x^* is the optimal solution, then we have*

$$F(y_k) - F(x^*) \leq \frac{C}{(k+2)^2}$$

where $C = 4(F(y_0) - F(x^*)) + 2L(f)\|x^* - y_0\|^2_{H_0}$.

PROOF. See Section 6. □

2.2 Coordinate Gradient Descent (CGD)

CGD is a simple method to conceptualize, very similar in structure to the gradient descent methods. Theoretical framework for solving non-smooth separable convex minimization using a coordinate gradient method was developed in [22]. They provide global convergence results as well as a local linear rate of convergence for the CGD method (which otherwise lacks a global complexity result). [29] extended the aforementioned work to least squares and logistic regression with L1-regularization.

CGD is viewed as a hybrid of gradient descent and coordinate descent methods with connection to gradient distribution method for unconstrained smooth optimization. To explain it further, we use $\nabla f(x)$ to build a quadratic approximation of f at x and apply coordinate descent to generate an improving direction d at x. We choose a subset of coordinate indices $\{1, 2, ..., m\}$ as J_k (using the Gauss-Southwell rule) and a symmetric positive definite H_k (approximation of $\nabla^2 f(x)$) and move x along the descent direction,

$$d_H(x; J) = arg\min_d\{\nabla f(x)^T d + \frac{d^T H d}{2} + g(x+d) | d_j = 0 \forall j \neq J\}$$

It is worth mentioning that solving $d_H(x; J)$ is essentially the same as solving $p_\mu(x_k)$ of FISTA, and it also admits a simple closed form solution. Once we have the descent direction, a suitable step length satisfying the Armijo's conditions (as indicated below) is used for the descent.

In the numerical implementation of CGD, one could approximate the Hessian, $H_k = I$ or $\theta_k I$ where $\theta_k \in R$. Alternately, as suggested by [29], we use the similar trick (which Multiple-scale FISTA is inspired from) for the approximation of Hessian at each iteration,

$$H_k = diag[\min\{\max\{\nabla^2 f(x_k)_{jj}, 10^{-10}\}, 10^{10}\}]_{j=1,2,...m}$$

and it performs extremely well in terms of convergence rate on the data sets that we experimented on. This considerable gain in performance can be used as an advantage while handling huge data sets.

Algorithm 2 CGD

1: Choose $x^0 \in R^n$.
2: **for** $k = 0, 1, \cdots$ **do**
3: Choose a $H_k \succ 0$ and a non empty set $J \subseteq N$
4: Solve for $d^k = d_{H_k}(x, J^k)$ using $x = x_k, J = J_k, H = H_k$
5: Choose α_k using Amijo's rule given below
6: Set $x_{k+1} = x_k + \alpha^k d^k$ with $\alpha^k > 0$
7: **end for**

Algorithm 3 Armijo Rule

1: Choose $\alpha^k_{init} > 0$, a large value.
2: Reduce α^k gradually till we find the largest one, that satisfies
3: $F(x_k + \alpha d^k) \leq F(x_k) + \alpha^k \delta^k$
4: $\delta^k = \nabla f(x_k)^T d^k + \frac{{d^k}^T H d^k}{2}$

CGD is a Gauss-Seidel type of algorithm, while FISTA is of the Jacobi type. However, the iterations of the two algorithms have an intimate relationship, as discussed in [18]. Basically, the backtracking line-search in FISTA is equivalent to choosing an appropriate H_k (but limited to a scalar multiple of the identity) in Algorithm 2 so that the Armijo's descent condition is automatically satisfied.

2.3 Stochastic Gradient Descent (SGD)

The basic idea of a stochastic gradient descent method is to use a subset of the training samples to approximate the current gradient of the objective function in each iteration and update the feature vector in an online fashion. Since the L1-regularized KLR is non-smooth, the method applied here is a sub-gradient method. As pointed out in [23], a direct application of the simple SGD does not result in efficient

training and quality solutions because the naive SGD does not enforces sparsity well.

Here we adapt the SGD algorithm with cumulative penalty (SGD-C) proposed by [23]. The two key features of SGD-C are the clipping-at-zero technique and the cumulative penalties. The former ensures that during a gradient update, the L_1 penalty is applied to the features to the extent that it does not change their signs. (See Lines 11 and 13 in Algorithm 4.) Note that this is equivalent to applying the shrinkage operator (5) in ISTA. The second feature keeps track of the total penalty that could have been applied to a feature and the actual penalty that has been applied so that the feature is updated in each iteration based on the difference of the two. (See Lines 4 and 15 in Algorithm 4.) The goal of this approach is to smooth out the effects of noisy gradients. We remark that we did not implement the "lazy update" approach, i.e. updating only the features used in the current sample. The reason is that the kernel matrix is usually dense for non-linear kernels, so the "lazy update" does not actually help speed up the algorithm.

For estimating the gradient of the logistic loss function $l(w) = -\sum_{i=1}^{N} \log(\sigma(y_i w^T k_i))$, we take the a convex combination of the gradients using the current sample point and the previous gradient vector, as similarly done in [28] and [27] for the Regularized Dual Averaging (RDA) method. Note that in RDA, the average gradients on all previous data points is used for the gradient step in each iteration. In our implementation, we took a slightly different approach and put more weight on the gradient estimated by the current point, i.e.

$$g^k = \frac{k-1}{k} g(a_k) + \frac{1}{k} g^{k-1}, \tag{11}$$

where g^k is the gradient used in iteration k, and $g(a_k)$ is the gradient of l on the data point a_k. The intuition behind the gradient averaging technique is inline with that of the cumulative penalties - we try to smooth out the randomness in the gradients by taking into account the information from the previous iterations.

We follow [23] to use exponential decay

$$\eta_k = \eta_0 \alpha^{-k/N} \tag{12}$$

for the learning rates, where N is the total number of samples. We formally state the procedures discussed above in Algorithm 4.

Algorithm 4 SGD-C

1: $u \leftarrow 0, w_i^0 \leftarrow 0, q_i \leftarrow 0 \forall i$.
2: **for** $k = 0, 1, \cdots$ **do**
3: $\quad \eta \leftarrow \eta_0 \alpha^{-k/N} \quad (\eta_0 = 0.5, \alpha = 2)$.
4: $\quad u \leftarrow u + \eta \frac{\lambda}{N}$
5: $\quad$ Select a sample with index k_j.
6: $\quad h^k \leftarrow \frac{1}{k} h^{k-1} + \frac{k-1}{k} \nabla l(w^k; a_k)$
7: $\quad w^k \leftarrow w^k - \eta h^k$
8: $\quad$ **for** $i = 1, \cdots, m$ **do**
9: $\quad\quad z \leftarrow w_i^k$
10: $\quad\quad$ **if** $w_i^k > 0$ **then**
11: $\quad\quad\quad w_i^k \leftarrow \max(0, w_i^k - (u + q_i))$
12: $\quad\quad$ **else if** $w_i^k < 0$ **then**
13: $\quad\quad\quad w_i^k \leftarrow \min(0, w_i^k + (u - q_i))$
14: $\quad\quad$ **end if**
15: $\quad\quad q_i \leftarrow q_i + (w_i^k - z)$
16: $\quad$ **end for**
17: **end for**

3. IMPLEMENTATION

3.1 Kernel linearization

Nonlinear kernel classifiers are attractive because they can approximate the decision boundary better given enough training data. However, they do not scale well in both training time and storage in large-scale settings, and it may take days to train on data sets with millions of points. On the other hand, linear classifiers run much more quickly, especially when the number of features is small, but behave relatively poorly when the underlying decision boundaries are nonlinear. Kernel linearization [19] combines the advantages of the linear and nonlinear classifiers. The key idea is to map the training data to a low-dimensional Euclidean space by a randomized feature map $\mathbf{z} : R^n \rightarrow R^D, D \ll n$, so that

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \approx z(x)' z(y). \tag{13}$$

Therefore, we can directly put the transformed data $z(x)$ to the linear classifier and speed up the computation. To calculate $z(x)$, randomized Fourier transform is used as,

$$z_j(x) = \sqrt{\frac{2}{D}} \cos(\omega_j x + b). \tag{14}$$

Since we use the RBF kernel, i.e.

$$k(i, j) = \exp^{-\frac{\|x_i - x_j\|^2}{\sigma^2}},$$

ω_j and b are sampled from the following distribution

$$w_{jk} \sim N(0, \frac{2}{\sigma^2}), \qquad b \sim U(0, 2\pi).$$

The following theorem (Claim 1 in [19]) establishes the convergence of the approximation.

THEOREM 2. *Let $\mathcal{M}$ be a compact subset of $\mathbb{R}^d$ with diameter $diam(\mathcal{M})$. Then, for the mapping z defined in* (14)*, we have*

$$Pr\left[\sup_{x,y \in \mathcal{M}} |z(x)' z(y) - k(y, x)| \geq \epsilon\right] \leq 2^8 \left(\frac{\sigma diam(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{D\epsilon^2}{4(d+2)}\right), \tag{15}$$

where $\sigma^2 \equiv E[\omega' \omega]$ is the second moment of the Fourier transform of k. Further, $\sup_{x,y \in \mathcal{M}} |z(x)' z(y) - k(y, x)| \leq \epsilon$ with any constant probability when $D = \Omega\left(\frac{d}{\epsilon^2} \log \frac{\sigma diam(\mathcal{M})}{\epsilon}\right)$.

3.2 A two-stage active-set training scheme

To boost the prediction performance of the deterministic algorithms discussed in Section 2, we propose a two-stage active-set approach on top of the algorithms. Specifically, we first use CGD/FISTA to solve the kernel logistic regression problem to optimality as usual. We then record the support of the optimal solution, i.e. the indices of the nonzero entries or the active set. Next, we optimize the problem

Data set	training	test	no. features	factor of λ_{max}
mnist10k	10,000	1000	784	0.5,0.2,0.1,0.05
gisette	6,000	1000	5,000	0.5,0.2,0.1,0.05
cod-rna	483,565	5000	8	0.001
mnist8M	1,560,154	19,000	784	0.1

Table 1: Various data sets used in the experiments.

again with regard to only the active set, while keeping the entries outside the active set at zero throughout the second stage. The regularization parameter λ for the second stage is chosen to be smaller than the that in the first stage, for example, 10% of the original to improve the prediction performance of the solution while preserving the current sparsity. Note that choosing a λ equal to the value in the first stage does not make sense since the solution from the first stage is optimal for that particular λ. Setting the second stage λ larger than that in the first stage may improve sparsity, but it may very likely impact prediction accuracy. We have observed through our experiments that the proposed approach can usually boost the prediction accuracy by up to 4%. Preliminary numerical results for medium-scale data are summarized in Table 6. We remark that the work involved in the second stage is far less than the first stage, since the active set is only a fraction of the entire feature set.

4. EXPERIMENTS

4.1 Data sets

We selected four data sets for our experiments, with two at medium-scale and the other two at large-scale. The attributes of the data sets are summarized in Table 1. mnist10k is from the MNIST data set [9] for classifying the handwritten digit '3'. The gisette data is one of five datasets of the NIPS 2003 feature selection challenge [5] for separating the highly confusable digits '4' and '9'. mnist8 is the augmented version of the original MNIST dataset through performing careful elastic deformation [11]. We chose to classify the handwritten digit '8'. cod-rna is a bio-medical data set introduced in [24].

4.2 Methodology

All experiments were run in Matlab R2009b environment on a 64-bit computer with a quad-core Intel Xeon 2GHz CPU and 32G memory. In the subsequent sections, whenever FISTA is mentioned, it refers to the one with our multi-scale extension, unless we specifically state that it is the original FISTA.

4.2.1 Medium-scale data sets

We first used mnist10k and gisette data to illustrate the effectiveness of kernel linearization, by testing the algorithms with the RBF kernel, and the linearized RBF kernel. For fair comparisons, we ran either FISTA or CGD first till the solution produced by the algorithm satisfied the termination condition, which we will discuss in Section 4.2.4. We then recorded the objective function value of this solution and used it as the reference objective value for the remaining two algorithms, i.e. we stopped the other two algorithms on the same data set only when they had returned a solution with an objective value less or equal to the reference value.[1]

[1]For SGD-C, we stopped the algorithm when the average objective value of the most recent 20 iterations was within 1% higher than the reference value, i.e. $\left|\frac{\bar{F}_{SGD}-F_{ref}}{F_{ref}}\right| \leq 0.01$.

For the reason discussed in Section 4.2.2, we ran SGD-C twice. In the first run, the objective value was computated for each iteration, and the algorithm was terminated with the aforementioned condition. We then ran SGD-C again for the same number of iterations as in the first run but without computing the objective values. The reported CPU times are from the second run.

4.2.2 Large-scale data sets

For the large-scale data sets (i.e. MNIST and cod-rna), we ran the experiments with only the linearized RBF kernels. For comparisons, again we tested CGD first till the termination criterion was met. We then recorded the reference objective value as done for the medium-scale data sets. FISTA was run till either the stopping criterion was met or the objective value reached the reference value. We allowed a maximum of 3000 iterations for FISTA and 5000 iterations for CGD. Here, we note that SGD-C does not actually require objective function evaluation to compute the solution of the current iteration – we need the objective values only when we are to determine the stopping point. The running time of SGD-C should be independent of the size of the data set. Since computing the objective function value involves the entire data set, the work load is very high, and it does not make sense to do the extra work in each SGD-C iteration. So, instead of stopping SGD-C by checking the objective values, we ran the algorithm for a large number of iterations, for example, as the size of the data set or several times of that. The running time in this case does not reflect the actual time required for SGD-C to produce a solution of the same quality as FISTA or CGD. Nevertheless, it still allows us to compare the other aspects of the algorithms.

4.2.3 The regularization parameter λ

One of the key parameters to set for the SKLR problem is the penalty λ on the L_1-norm. There are two major approaches for choosing λ. One is through computing the regularization path, as done in [30]. Specifically, we start with a large λ, on which the optimization algorithms converge faster. We then gradually decrease λ and use the solution from the previous λ as the starting solution for the current λ. This technique is also known as continuation or warm-starting [6]. [10] uses 10-fold cross validation to compute the hold-out accuracy and select the λ that achieves the highest accuracy.

With a given kernel matrix K, we can also explicitly compute an upper bound λ_{max} on the meaningful range of λ following the formulation given in [8],

$$\lambda_{max} = \|K^T \tilde{b}\|_\infty, \tag{16}$$

where

$$\tilde{b}_i = \begin{cases} N_-/N, & \text{if } y_i = 1, \\ -N_+/N, & y_i = -1, \end{cases} \quad i = 1 \cdots N.$$

N_- and N_+ above denote the number of positive and negative training labels respectively. The optimal solution is the zero vector for any λ larger than λ_{max}. For medium-scale datasets, we ran the three algorithms with four values of λ: $0.5\lambda_{max}$, $0.2\lambda_{max}$, $0.1\lambda_{max}$, and $0.05\lambda_{max}$. We show in Section 4.3 that usually a good balance between accuracy

Alg/Data set	mnist10k	$\lambda = 0.5\lambda_{max}$	$\lambda = 0.2\lambda_{max}$	$\lambda = 0.1\lambda_{max}$	$\lambda = 0.05\lambda_{max}$
SGD-C	RBF	84.6/3.0e+001/0.1	88.1/8.2e+001/0.2	90.6/1.3e+002/0.3	91.3/1.4e+002/0.6
	linearRBF	84.5/4.1e+000/2.6	90.3/6.7e+000/6.8	91.5/9.3e+000/13.2	92.8/1.2e+001/23.8
FISTA	RBF	85.3/4.8e+003/0.1	88.7/1.1e+004/0.3	90.4/1.1e+004/0.6	91.9/6.3e+003/2.7
	linearRBF	84.6/4.4e+000/1.4	89.6/2.2e+001/4.6	91.5/3.3e+001/9.2	92.7/3.5e+001/16.0
CGD	RBF	85.4/6.1e+002/0.3	88.5/3.5e+003/14.3	90.2/5.7e+003/37.5	92.4/6.2e+003/100.0
	linearRBF	84.8/3.0e-001/1.4	89.4/3.6e-001/4.0	91.3/9.8e-001/9.6	92.6/2.6e+000/20.2

Table 2: Results for mnist10k: each cell reads (accuracy(%)/cpu/sparsity(%)). The results were obtained without using the two-stage active set approach.

and sparsity occurs at around $0.2\lambda_{max} to 0.1\lambda_{max}$. In Table 1, we have specified the actual factors of λ_{max} that we used in our experiments. This approach is simple and the chosen parameter values are usually appropriate enough for ensuring that we test the optimization algorithms in the relevant regime for classification.

4.2.4 Stopping criterion: duality gap

Our stopping criterion for the deterministic algorithms (i.e. FISTA and CGD) is based on the duality gap of the current solution. The duality gap is the difference between the primal and the dual objective values. Since strong duality holds for the L1-regularized kernel logistic regression problems, the optimal solution for a given λ has zero duality gap. Hence, the duality gap is a measure of the optimality of the current solution.

Computing the duality gap for L1-regularized logistic regression has been discussed in [8], and a more general version has also been discussed in [17]. The same method applies to the kernel version in our case. Note that computing the duality gap requires evaluation of the true gradient, which is avoided in SGD-C. So, for SGD-C, we either use the user-supplied maximum number of iterations or the reference objective value as the termination criterion.

4.3 Results

We present the results for different data sets in Tables 2, 3, and 5[2]. We compare the relative efficiency of the three optimization algorithms through plotting the objective function values against the CPU time (Figure 1) and the duality gap against the CPU time for the RBF and the linearized RBF kernels. We have also plotted the sparsity against CPU time graph (Figure 2) to gain some insight on how the sparsity changes as the algorithms progress in optimization. Here, we define sparsity as the percentage of non-zero entries in the solution. To illustrate the effect of the regularization parameter λ on the test accuracy and the sparsity of the solutions, we have plotted the graph of the prediction accuracy/sparsity/training time against the four values of λ for the medium-scale data sets. (See Figures 3 and 4.)

We can see from Table 2 that by using RBF kernels with the kernel linearization technique, we can achieve very similar prediction accuracies as RBF kernels but at only a fraction of the computational cost. For the gisette data, by cross-referencing Table 6 with Table 3, we see that our two-stage active set approach can also boost the prediction accuracies of the linearized RBF kernels up to the level inline with the RBF kernels. Indeed, the two-stage active set approach

[2]We have omitted the results of FISTA on mnist8M for the training time exceeded the maximum limit of ten hours.

Alg/Data set	gisette	$\lambda = 0.5\lambda_{max}$	$\lambda = 0.2\lambda_{max}$	$\lambda = 0.1\lambda_{max}$	$\lambda = 0.05\lambda_{max}$
SGD-C	RBF	81.6/3.8e+001/0.6	88.8/6.5e+001/2.1	92.5/1.3e+002/16.6	89.1/7.6e+001/54.9
	linearRBF	79.5/2.0e+001/5.8	86.3/6.2e+001/16.6	88.7/8.2e+001/26.3	89.6/2.3e+002/37.9
FISTA	RBF	82.2/6.0e+003/1.2	89.6/6.7e+003/11.0	91.0/9.5e+003/28.6	93.0/1.1e+004/62.8
	linearRBF	79.6/8.3e+000/3.6	86.2/1.1e+001/12.4	88.6/1.3e+001/20.0	90.2/2.1e+001/31.3
CGD	RBF	82.8/5.9e+002/35.9	89.6/2.2e+003/100.0	91.1/2.9e+003/100.0	92.3/4.0e+003/100.0
	linearRBF	78.8/6.6e-001/2.6	86.0/2.5e+000/12.2	88.7/6.1e+000/20.0	90.1/1.2e+001/33.2

Table 3: Results for gisette: each cell reads (accuracy(%)/cpu/sparsity(%)). The results were obtained without using the two-stage active set approach.

Model	Software	Accuracy (%)
L2-regularized Logistic regression	LIBLINEAR	86.59
L1-regularized KLR with linearization	CGD	88.5
L2-regularized KLR with linearization	SGD	88.0

Table 4: Comparison on prediction accuracy of different learning models on cod-rna.

has improved the accuracies of both FISTA and CGD by up to 4% on both medium-scale data sets across all four values of λ. We have also compared the prediction performance of SKLR with kernel linearization with other popular classification methods in Table 4. SKLR has produced better results on the cod-rna data. In particular, compared to L2-regularized KLR, SKLR has a 100-time less test computation work load, taking into account the 1% sparsity in its solution.

To compare and analyze the performance of the three optimization algorithms for solving the SKLR problem, we consider three aspects: convergence speed, sparsity, and prediction accuracy. In terms of convergence speed, it is clear from Figure 1 that CGD has considerable advantage over FISTA and SGD-C. Although this is surprising from the theoretical point of view, we believe that the excellent computational performance of CGD should be attributed to the use of the diagonal approximate Hessian matrix as well as the tailored Armijo line-search rule. In fact, we have applied the same diagonal approximate Hessian matrix to FISTA in our experiments (multi-scale FISTA). This indeed has yielded significant improvement in computational performance for FISTA as we show in Table 7. The advantage of cheap per iteration cost of SGD-C is more obvious when dealing with large-scale data. It appears from Table 5 that SGD-C is able to produce a solution of similar quality as CGD and FISTA at only less than 10% of the training time. However, we need to keep in mind that the gain in speed for SGD-C comes with the condition of not computing any objective function values, thus sacrificing the optimality guarantee upon termination.

Tables 5 and 3 show that CGD and FISTA are able to produce solutions of better sparsity than SGD-C with the linearized RBF kernels. Figure 2 also shows that CGD keeps

Data set	Alg	Accuracy (%)	Training time	Iters	Sparsity (%)
cod-rna	SGD-C	88.5	1.08e+003	1450695	3.8
	FISTA	88.5	2.71e+004	3000	1.6
	CGD	88.5	1.48e+004	3993	1.4
mnist8M	SGD-C	83.4	2.36e+003	3120308	16.4
	CGD	85.5	2.74e+004	1991	17.6

Table 5: Results for large-scale data sets. FISTA and CGD were run with the two-stage active set approach. The kernel type is RBF with kernel linearization.

Data set	Alg	$\lambda = 0.5\lambda_{max}$	$\lambda = 0.2\lambda_{max}$	$\lambda = 0.1\lambda_{max}$	$\lambda = 0.05\lambda_{max}$
mnist10k	FISTA	84.90 (+0.3)	90.90 (+1.3)	92.90 (+1.4)	93.30 (+0.6)
	CGD	85.20 (+0.4)	90.80 (+1.4)	91.90 (+0.6)	93.20 (+0.6)
gisette	FISTA	82.90 (+3.3)	88.40 (+2.2)	90.60 (+2)	90.60 (+0.4)
	CGD	83.00 (+4.2)	88.60 (+2.6)	90.80 (+2.1)	90.90 (+0.8)

Table 6: The boosted prediction accuracies (%) with the two-stage active set approach on the medium-scale data sets. The numbers in the brackets are the improvements over the corresponding entries in Tables 2 and 3. The kernel type is RBF with kernel linearization.

Data set	Alg	$\lambda = 0.5\lambda_{max}$	$\lambda = 0.2\lambda_{max}$	$\lambda = 0.1\lambda_{max}$	$\lambda = 0.05\lambda_{max}$
mnist10k	FISTA	61.64	52.82	26.13	25.27
gisette	FISTA	40.72	24.81	30.17	20.45

Table 7: Improvement in the number of iterations with multi-scale FISTA on mnist10k and gisette. The number in the table are computed as (no. iters multi-scale FISTA / no. iters original FISTA)$\times$100%.

the sparsity at a much lower level than the other two methods throughout the training process (except for mnist8M). However, for RBF kernels, CGD performed particularly bad in terms of sparsity. FISTA and SGD-C appear to be more consistent in sparsity performance. Another interesting observation here is that CGD always started with a very low sparsity level (very few non-zero entries in the solution) and converged to a higher sparsity level, while SGD-C and

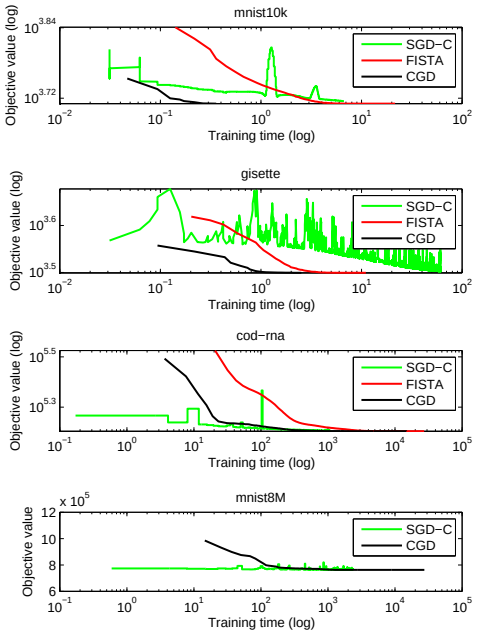

Figure 1: Objective values against training time for RBF kernels with linearization. $\lambda = 0.2\lambda_{max}$ for the plots of medium-scale data. $\lambda = 0.001\lambda_{max}$ for cod-rna, and $\lambda = 0.1\lambda_{max}$ for mnist8M.

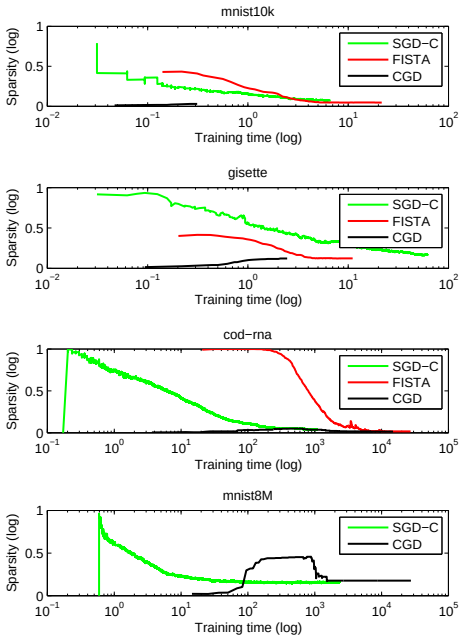

Figure 2: Sparsity against Training time for RBF kernels with linearization. $\lambda = 0.2\lambda_{max}$ for the plots of medium-scale data. $\lambda = 0.001\lambda_{max}$ for cod-rna, and $\lambda = 0.1\lambda_{max}$ for mnist8M.

FISTA did exactly the opposite.

The three algorithms have generally yielded similar results in terms of prediction accuracy. The deterministic algorithms have a slight edge in that aspect because of the optimality guarantee upon termination. These observations are evident from Tables 2, 3, 5, and Figures 3 and 4.

Figures 3 and 4 illustrate the effect of the regularization parameter λ on the training time, prediction accuracy, and sparsity. The general trend for all three algorithms is that as λ increases, we need less training time and obtain better sparsity with the price of a decreasing accuracy. The decrease in training time, i.e. faster convergence for a larger λ was discussed in [6] on L_1-minimization for compressed sensing, and that forms the basis for the continuation scheme mentioned in Section 4.2.3. Our observation here confirms that it is possible to use continuation to solve SKLR too. We also observe from the plots that a good tradeoff between sparsity and accuracy is around $0.1\lambda_{max}$ to $0.2\lambda_{max}$.

5. CONCLUSION

We have carried out a comparative study of three popular first-order optimization algorithms on medium and large-scale L_1-regularized kernel logistic regression problems. We have also proposed a multiple-scale extension to FISTA and a two-stage active set training scheme, which helps improve the prediction accuracies of the deterministic learning algorithms. Through our experiments, We demonstrated the effectiveness of kernel linearization and the computational

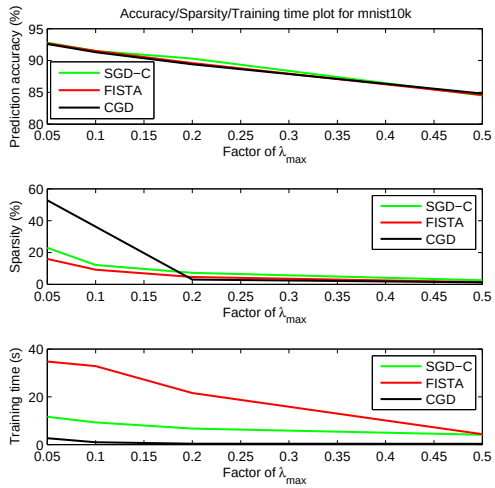

Figure 3: Comparisons of prediction accuracy/sparsity/training time across different values of λ for mnist10k with linearized RBF kernels. The plots are based on Table 2.

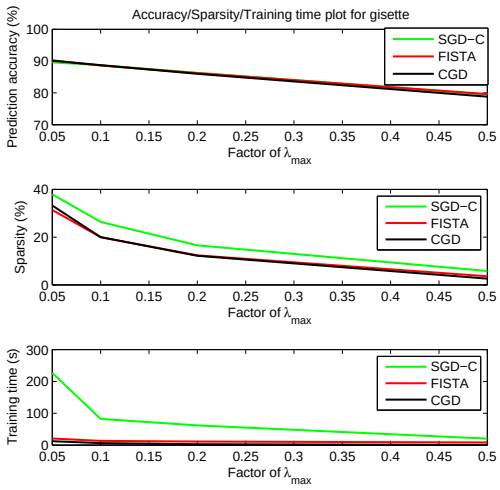

Figure 4: Comparisons of prediction accuracy/sparsity/training time across different values of λ for gisette with linearized RBF kernels. The plots are based on Table 3.

advantage that it brings, which makes large-scale kernel learning highly feasible. Our observations show that while SGD-C remains to be the most efficient algorithm for large-scale data, the deterministic algorithms (especially CGD) are also well-capable of handling a wide range of data with optimality guarantee. Finally, the success of the algorithms that we have introduced in this paper demonstrates the potential and computational advantage of sparse modeling in large-scale classification.

6. PROOF OF CONVERGENCE RATE FOR MULTI-SCALE FISTA

In this section, we give the proof for Theorem 1. Since it's similar to ideas in [21], we only state the main results and modifications made on the original proof.

$$F(x) = f(x) + g(x) \tag{17}$$

LEMMA 1. *Let $\{x_k\}$ and $\{y_k\}$ be generated by multi-scaled FISTA. Define*

$$\theta_k := \frac{1}{t_k}$$

such that

$$x_{k+1} = y_k + \theta_k(\theta_{k-1}^{-1} - 1)(y_k - y_{k-1}).$$

. If x^* is the optimal solution of (17), then

$$\begin{array}{l} \frac{1}{\theta_k^2}(F(y_{k+1}) - F(x^*)) \le \frac{1}{\theta_{k-1}^2}(F(y_k) - F(x^*)) \\ + \frac{L(f)}{2}(\|x^* - w_k)\|_{H_k}^2 - \|x^* - w_{k+1}\|_{H_k}^2) \end{array} \tag{18}$$

where

$$\begin{aligned} w_k &= y_{k-1} + \theta_{k-1}^{-1}(y_k - y_{k-1}) \\ \|x\|_H &= x^T H x \end{aligned}$$

PROOF. The proof is very similar to Proposition 2 of [21]. We first replace all the squared norms $\|\cdot\|^2$ by $\|\cdot\|_{H_k}$. Then we add the function $g(y_{k+1})$ to the first inequality and $g(y)$ to the second inequality. By using the fact $\frac{1-\theta_{k+1}}{\theta_{k+1}^2} \le \frac{1}{\theta_k^2}$ and the convexity of $g(y)$, i.e.,

$$g(y) \le (1 - \theta_k)g(x_k) + \theta_k g(x^*)$$

for $y = (1 - \theta_k)x_k + \theta_k x^*$, we conclude that (18) is true. □

We now prove Theorem 1 in the following.

PROOF. Sum up (18) from 1 to k, we have the following

$$\begin{array}{rcl} \frac{1}{\theta_k^2}(F(y_k) - F(x^*)) & \le & (F(y_0) - F(x^*)) \\ & & + \frac{L(f)}{2}\|x^* - x_0\|_{H_0}^2 \end{array} \tag{19}$$

We use the relation $H_k - H_{k+1} \succ 0$ to achieve the last term in (19) since

$$\|x^* - w_k\|_{H_{k-1}} - \|x^* - w_k\|_{H_k} = \|x^* - w_k\|_{H_{k-1} - H_k} \ge 0$$

It's not hard to verify that $\theta_k \le \frac{2}{k+2}$, therefore by multiplying both sides by θ_k^2, we have

$$F(y_k) - F(x^*) \le C/(k+2)^2$$

where $C = 4(F(y_0) - F(x^*)) + 2L(f)\|x^* - y_0\|_{H_0}^2$. □

7. REFERENCES

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[2] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier and G. Saporta, editors, *Proceedings of the 19th*

International Conference on Computational Statistics (COMPSTAT'2010), pages 177–187, Paris, France, August 2010. Springer.

[3] E. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

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

[5] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. *Advances in Neural Information Processing Systems*, 17:545–552, 2005.

[6] E. Hale, W. Yin, and Y. Zhang. Fixed-Point Continuation for L1-Minimization: Methodology and Convergence. *SIAM Journal on Optimization*, 19:1107, 2008.

[7] S. Hoi, M. Lyu, and E. Chang. Learning the unified kernel machines for classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 187–196. ACM, 2006.

[8] K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine learning research*, 8(8):1519–1555, 2007.

[9] Y. LeCun and C. Cortes. The MNIST database of handwritten digits, 1998.

[10] S. Lee, H. Lee, P. Abbeel, and A. Ng. Efficient L1 Regularized Logistic Regression. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 401. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[11] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

[12] L. Meier, S. Van De Geer, and P. Buhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[13] Y. Nesterov. A method of solving a convex programming problem with convergence rate O (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[14] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[15] Y. Nesterov and I. Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer Netherlands, 2004.

[16] A. Ng. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[17] J. Obozinski and F. Bach. Network Flow Algorithms for Structured Sparsity. 2010.

[18] Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient Block-coordinate Descent Algorithms for the Group Lasso. 2010.

[19] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20:1177–1184, 2008.

[20] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[21] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2008.

[22] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1):387–423, 2009.

[23] Y. Tsuruoka, J. Tsujii, and S. Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics, 2009.

[24] A. Uzilov, J. Keegan, and D. Mathews. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7(1):173, 2006.

[25] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

[26] M. Wainwright, P. Ravikumar, and J. Lafferty. High-Dimensional Graphical Model Selection Using l˜ 1-Regularized Logistic Regression. *Advances in Neural Information Processing Systems*, 19:1465, 2007.

[27] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Advances in Neural Information Processing Systems*, 23, 2009.

[28] H. Yang, Z. Xu, I. King, and M. Lyu. Online learning for group lasso. In *27th Intąrl Conf. on Machine Learning (ICML2010)*. Citeseer.

[29] S. Yun and K. chuan Toh. A coordinate gradient descent method for l1-regularized convex minimization. Technical report, Department of Mathematics, National University of Singapore, 2008.

[30] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 14(1):185–205, 2005.