SCIENCE CHINA
Information Sciences

• **Supplementary File** •

# Neural Network Equivalent Model for Highly Efficient Massive Data Classification

Siquan Yu[1,2,3], Zhi Han[2,3*], Yandong Tang[2,3] & Chengdong Wu[4]

[1]*School of Information Science and Engineering, Northeastern University, Shenyang* 110819, *China ;*
[2]*Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang* 110016, *China;*
[3]*State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang* 110016, *China;*
[4]*Faculty of Robot Science and Engineering, Northeastern University, Shenyang* 110819, *China*

This supplementary document mainly includes five parts. In Appendix A, we introduce the details of the proposed equivalent model. In Appendixes B and C, we analyze the convergence issue of ADMM, and provide the proof of Theorem 1. In Appendixes D and E, we conduct a series of experiments including the numerical experiments, the real word experiments and the massive data classification experiment, which are used to verify the effectiveness of our model and the correctness of theoretical assertions.

## Appendix A The proposed equivalent model.

Our equivalent model mainly consists of two parts which are the hidden parameter generation and the classification hyperplane learning. We first analyze the rationality of the hidden parameter generation and use it to map data to the feature space. Then, we explain the solution process of the classification hyperplane based on ADMM. The proposed equivalent model is shown in Figure A1.

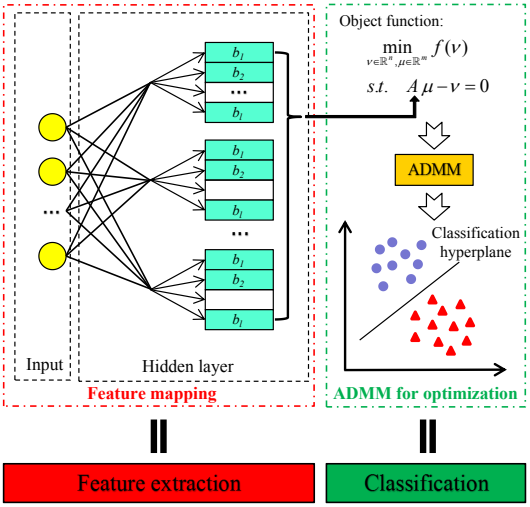



**Figure A1** The demonstration of the proposed algorithm.

Training neural networks is a typical research topic in machine learning. Generally speaking, there is inconsistency between network optimization and learning. On the one hand, to guarantee network convergence, numerous free parameters are often required. On the other hand, the numerous free parameters and large capacity of the neural network often lead to the over-fitting phenomenon. Thus, over-parameterizing improves the expression ability of the network, but it reduces the learnability of the network. To avoid the inconsistency between optimizing and learning a neural network, we borrow the idea of the deterministic assignment strategy proposed in [5] to reduce the capacity of a neural networks model in an off-line manner. Another advantage of assigning parameters instead of training them is that it can reduce the computational burden.

In [5], LtDaHPs regards the weights of a neural network as the minimal Riesz energy point on the sphere, and sets the biases as equally spaced points (ESPs) in an interval. The generated network almost has the same generalization capacity as the trained network in regression tasks. Compared with [5], the novelty of our work is that it maps the classification samples to the feature space by using the generated weights.

Given the training samples $D := \{(x_i, y_i)\}_{i=1}^m$, the feature space is,

$$\mathcal{H}_{l,n,\sigma} := \{\sum_{j=1}^{n}\sum_{k=1}^{l} a_{jk}\sigma(\alpha_j x - b_k) : a_{jk} \in \mathbb{R}^1\}, \tag{A1}$$

* Corresponding author (email: hanzhi@sia.cn)

where $\sigma$ is a nonlinear function, $n \times l$ denotes splitting of $N$. The inner weights $\{\alpha_j\}_{j=1}^n$ is obtained by minimizing the Riesz $\tau$-energy point of $\mathbb{S}^{d-1}$ with $\tau \geqslant d-1$, and $\{b_k\}_{k=1}^l$ is the ESPs in the interval $[-1/2, 1/2]$.

After feature mapping, since the maximal margin principle is not considered in massive data classification, it can be transformed to an empirical risk minimization problem.

$$f_D = \arg \min_{f \in \mathcal{H}_{l,n,\sigma}} \left\{ \frac{1}{m} \sum_{i=1}^{m} (1 - y_i f(x_i))_+ \right\}, \tag{A2}$$

where $t_+$ is the hinge loss function which is defined as $max\{t, 0\}$ for $t \in \mathbb{R}$ .

Due to the non-smoothness of (A2), the classical optimization methods (e.g. gradient decent method) cannot be feasibly applied to solve this problem. Concerning the sub-gradient methods, [8] showed that $\mathcal{O}(1/\epsilon)$ iterations are required and $\mathcal{O}(mn)$ float computations are needed in each iteration, where $\epsilon$ is the approximation accuracy between the estimator generated by a sub-gradient method, $m$ is the number of training samples, $n$ is the dimension of the data feature. Faced with massive data classification task, $\epsilon$ should be extremely small and thus sub-gradient methods involve extremely high computational burden. Since ADMM is a powerful optimization algorithm that can solve the non-smooth convex problem, we try to adopt ADMM to solve the un-regularized optimization problem (A2). First, we reformulate the problem (A2) as an unconstrained form and introduce a new variable $\nu$.

$$\begin{aligned} &\min_{\nu \in \mathbb{R}^n, \mu \in \mathbb{R}^m} f(\nu), \\ &\text{s.t.} \quad A\mu - \nu = 0, \end{aligned} \tag{A3}$$

where $f(\nu) := \frac{1}{m} \sum_{i=1}^m (1 - y_i \nu_i)_+$, and $A \in \mathbb{R}^{m \times n}$ is the feature matrix with $A_{ij} = \sigma(\alpha_j x_i - b_{kj})$. The augmented Lagrangian function of (A3) is as follows,

$$L_\beta(\mu, \nu, w) = f(\nu) + \langle \omega, A\mu - \nu \rangle + \frac{\beta}{2} ||A\mu - \nu||_2^2, \tag{A4}$$

where $\beta > 0$ is the augmented Lagrangian coefficient, and $\omega \in \mathbb{R}^m$ is a multiplier variable. Thus, the detailed ADMM processes for problem (A2) are shown as follows.

Given an initialization $\mu^0, \nu^0, \omega^0, \alpha > 0$ and $\beta > 0$, for $t = 0, 1, ...,$ ,

$$\mu^{t+1} = \arg\min_{\mu \in \mathbb{R}^n} \left\{ L_\beta(\mu, \nu^t, \omega^t) + \frac{\alpha}{2} ||\mu - \mu^t||_2^2 \right\}, \tag{A5}$$

$$\nu^{t+1} = \arg\min_{\mu \in \mathbb{R}^n} L_\beta(\mu^{t+1}, \nu, \omega^t), \tag{A6}$$

$$w^{t+1} = \omega^t + \beta(A\mu^{t+1} - v^{t+1}), \tag{A7}$$

where $t = 0, 1, 2, ....$

According to [7], the closed form solutions of A5 and A6 can be expressed as,

$$\mu^{t+1} = (\beta A^T A + \alpha I_n)^{-1} (\alpha \mu^t + \beta A^T \nu^t - A^T \omega^t), \tag{A8}$$

$$\nu^{t+1} = \mathrm{Hinge}_{\mathrm{m}\beta}(\mathrm{y}, \mathrm{A}\mu^{\mathrm{t}+1} + \beta^{-1}\omega^{\mathrm{t}}), \tag{A9}$$

where $\mathrm{Hinge}_\gamma(\zeta, \xi) = (\mathrm{hinge}_\gamma(\zeta(1), \xi(1)), ..., \mathrm{hinge}_\gamma(\zeta(\mathrm{m}), \xi(\mathrm{m})))$, $\zeta = (\zeta(1), ..., \zeta(m))^T$ for $\zeta \in \mathbb{R}^m$, $\gamma > 0$ and

$$\mathrm{hinge}_\gamma(\mathrm{a}, \mathrm{b}) =, \begin{cases} b, & \text{if } a = 0, \\ b + \gamma^{-1} a, & \text{if } a \neq 0 \text{ and } ab \leqslant 1 - \gamma^{-1} a^2, \\ a^{-1}, & \text{if } a \neq 0 \text{ and } 1 - \gamma^{-1} a^2 < ab < 1, \\ b, & \text{if } a \neq 0 \text{ and } ab > 1. \end{cases}$$

We summarize the specific algorithm in Algorithm A1.

**Algorithm A1** Neural Network Equivalent Model for Massive Data Classification

**Input:** training samples $D := \{(x_i, y_i)\}_{i=1}^m$, the nonlinear function $\sigma$, initialization of ADMM, $p^0 = (\mu^0, \nu^0, \omega^0) = (0, y, 0)$, $\alpha = \beta = 1$, a splitting $N = n \times l$ and the stopping tolerance parameter $tol > 0$.

**Output:** $f_{D,l,n,\sigma} = \sum_{j=1}^n \mu_j^{output} A(l, n, \sigma, \cdot)$;

1: **Generation inner weights and bias:**
Take the inner weights $\{\alpha_j\}_{j=1}^n$ as the minimal Riesz $\tau$-energy point of $\mathbb{S}^{d-1}$ with $\tau \geqslant d-1$, and set $\{b_k\}_{k=1}^l$ to $b_k = -\frac{1}{2} + \frac{k}{l}, k = 1, 2, ..., l$ in the range of $[-1/2, 1/2]$,

2: **Obtain the parameterized hypothesis space:**
$A_{l,n,\sigma} := \{\sum_{j=1}^n \sum_{k=1}^l a_{jk} \sigma(\alpha_j x - b_k) : a_{jk} \in \mathbb{R}^1\}$

3: **Update:**
$\mu^{t+1} = (\beta A^T A + \alpha I_n)^{-1} (\alpha \mu^t + \beta A^T \nu^t - A^T \omega^t)$,
$\nu^{t+1} = \mathrm{Hinge}_{\mathrm{m}\beta}(\mathrm{y}, \mathrm{A}\mu^{\mathrm{t}+1} + \beta^{-1}\omega^{\mathrm{t}})$,
$\omega^{t+1} = \omega^t + \beta(A\mu^{t+1} - \nu^{t+1})$.

4: **End:**
The first $t$ satisfying $||p^{t+1} - p^t||^2 < tol$.

## Appendix B Convergence of ADMM

We first analyze the convergence issue of ADMM. The closed form of $(\mu^k, \nu^k, \omega^k)$ can be deduced easily from their definitions. In particular, from [7, Sec.3], we can directly derive the lemma.

**Lemma 1.** Denote $(\mu^t, \nu^t, \omega^t)$ as the $t$-th iteration of ADMM. There holds

$$\mu^{t+1} = (\beta A^T A + \alpha \mathbf{I}_n)^{-1}(\alpha\mu^t + \beta A^T \nu^t - A^T\omega^t) \tag{B1}$$

and

$$v^{t+1} = \mathrm{Hinge}_{m\beta}(y, A\mu^{t+1} + \beta^{-1}\omega^t), \tag{B2}$$

where $\mathbf{I}_n$ is an identity matrix with the dimension of $n$,

$$\mathrm{Hinge}_\gamma(\eta, \epsilon) = (\mathrm{hinge}_\gamma(\eta(1), \epsilon(1)), \ldots, \mathrm{hinge}_\gamma(\eta(m), \epsilon(m)))^T,$$

$\eta = (\eta(1), \ldots, \eta(m))^T$ for $\eta \in \mathbb{R}^m$, $\gamma > 0$ and

$$\mathrm{hinge}_\gamma(\mu, \nu) = \begin{cases} \nu, & \text{if } \mu = 0, \\ \nu + \gamma^{-1}\mu, & \text{if } \mu \neq 0 \text{ and } \mu\nu \leqslant 1 - \gamma^{-1}\mu^2, \\ \mu^{-1}, & \text{if } \mu \neq 0 \text{ and } 1 - \gamma^{-1}\mu^2 < \mu\nu < 1, \\ \nu, & \text{if } \mu \neq 0 \text{ and } \mu\nu \geqslant 1. \end{cases}$$

In the following lemma, whose proof is standard and can be found in [7], we demonstrate the rationality of the ADMM algorithm, and explain that it can obtain a global minimum solution when solving the optimization problem (A4). In addition, the lemma also interprets that the convergence of (A4) is independent of the values of $\alpha, \beta$ and $(\mu^0, \nu^0, \omega^0)$.

**Lemma 2.** Let $\{p^t := (\mu^t, \nu^t, \omega^t)\}$ be the sequence generated by (A5), (A6) and (A7) for any $\alpha > 0$, $\beta > 0$ and finite initial point $(\mu^0, \nu^0, \omega^0)$. If there is a solution to problem (A4). Then $p^t$ converges to some $p^* = (\mu^*, \nu^*, \omega^*)$ and $\mu^*$ is a global minimizer of (A4).

The above lemma presents that the ADMM algorithms converge to the global minimum of the optimization problem (A4). Thus, it is feasible and efficient to solve the optimization problem (A4).

## Appendix C Proof of Theorem 1

The main tools used in our proof are the approximation error analysis in [5] and the oracle inequality in [7]. Denote $\phi(z) := (1-z)_+$, $\mathcal{E}(h) := \int_Z \phi(yh(x))d\rho$ for $h \in L^2_{\rho_X}$. We can find the relationship in [6] that

$$\mathcal{R}(\mathrm{sgn}(h)) - \mathcal{R}(h_c) \leqslant \mathcal{E}(h) - \mathcal{E}(h_\rho). \tag{C1}$$

Thus, it can be used to bound $\mathcal{E}(\pi h_{D,\ell,n,\sigma}) - \mathcal{E}(h_\rho)$, where $\pi h$ satisfy $\pi h = \mathrm{sgn}(h) \cdot \min\{1, h\}$. It denote a truncation of $h \in \mathbb{R}$ to $[-1, 1]$, which $\mathrm{sgn}(h)$ denote the sign function. Then, the empirical formula of $\mathcal{E}(h)$ can be defined by $\mathcal{E}_D(h) := \frac{1}{m}\sum_{i=1}^m \phi(y_i h(x_i))$. Furthermore, we can derive the following error decomposition.

**Lemma 3.** For $f_0 \in \mathcal{H}_{\ell,n,\sigma}$, there holds

$$\mathcal{E}(\pi f_{D,\ell,n,\sigma}) - \mathcal{E}(f_\rho) \leqslant \mathcal{D}(f_0) + \mathcal{S}_D(f_0) - \mathcal{S}_D(\pi f_{D,\ell,n,\sigma}), \tag{C2}$$

where

$$\mathcal{D}(f_0) := \mathcal{E}(f_0) - \mathcal{E}(f_\rho), \tag{C3}$$

and

$$\mathcal{S}_D(f) := [\mathcal{E}_D(f) - \mathcal{E}_D(f_\rho)] - [\mathcal{E}(f) - \mathcal{E}(f_\rho)]. \tag{C4}$$

Thus, in order to estimate the generalization ability, it need to bound the approximation error and sample error respectively. The main tool for the former one is the following lemma, which is derived in [5, Lemma 6]

**Lemma 4.** Let $0 \leqslant v \leqslant \frac{b+1}{2}$. If $f_\rho \in W_2^r$ and $n \sim \ell^{b-1}$, there exist a function $f_0 \in \mathcal{H}_{l,n,\phi_K}$ such that

$$\|f_\rho - f_0\|_{L^2(\mathbf{B}^b)} \leqslant C(n\ell)^{-v/b},$$

where $C$ is a constant and depends only on $v$, $b$, $\phi$ and $f_\rho$.

Based on Lemma 4, we derive from

$$\mathcal{E}(f_0) - \mathcal{E}(f_\rho) \leqslant \|f_0 - f_\rho\|_{L^1_{\rho_X}}$$

and the definition of $D_{\rho_X}$ that

$$\mathcal{D}(f_0) \leqslant \|f_0 - f_\rho\|_{L^1_{\rho_X}} \leqslant D_{\rho_X}\|f_\rho - f_0\|_{L^2(\mathbf{B}^b)} \leqslant D_{\rho_X} C(n\ell)^{-v/b}. \tag{C5}$$

The bound of $\mathcal{S}_D(f_0)$ is also standard. Thus, in the same way as proof [7], the lemma that follow can be easily derived.

**Lemma 5.** Let $\theta \in (0, 1)$, with confidence at least $1 - \theta/2$, then

$$\mathcal{S}_D(f_0) \leqslant \frac{8\log(2/\theta)}{3m} + \left(\frac{2\log(2/\theta)}{m}\right)^{\frac{1}{2}} + \frac{1}{2}D(f_0).$$

Then, we consider to bound $-\mathcal{S}_D(\pi f_{D,\ell,n,\sigma})$. The following lemma is a covering number estimate, whose proof can be found in [5]

**Lemma 6.** Let $\pi\mathcal{H}_{\ell,n,\sigma} := \{\pi f : f \in \mathcal{H}_{\ell,n,\sigma}\}$. For any $\varepsilon > 0$, it satisfy

$$\mathcal{H}_\varepsilon(\pi\mathcal{H}_{\ell,n,\sigma}, G(X)) \leqslant c_2' n^d \ell^d \log\frac{1}{\varepsilon},$$

where $c_2'$ is positive and depends only on $d$, $G(X)$ denotes the set of all continuous functions defined on $X$.

Based on this covering number estimate, we can get the following error estimate, whose proof can be found in [7].

**Lemma 7.** Let $\theta \in (0,1)$, with confidence at least $1-\theta/2$, then

$$-\mathcal{S}_D(\pi f_{D,\ell,n,\sigma}) \leqslant \frac{1}{2}(\mathcal{E}(\pi f_{D,\ell,n,\sigma}) - \mathcal{E}(f_\rho)) + c_3\left[\frac{\ell n\log m}{m}\right]^{\frac{1}{2}}\log\frac{4}{\theta},$$

where $c_3$ is a constant independent of $m$, $\ell$, $n$ or $\theta$.

*Proof.* [Proof of Theorem 1] With confidence $1-\theta$, combining the aforemention Lemma 3 with Lemma 5, Lemma 7 and (C5), there holds

$$\mathcal{E}(\pi f_{D,\ell,n,\sigma}) - \mathcal{E}(f_\rho) \leqslant 6D_{\rho_X}G(n\ell)^{-r/d} + \frac{8\log(4/\theta)}{3m} + \left(\frac{2\log(4/\theta)}{m}\right)^{\frac{1}{2}} + \frac{1}{2}(\mathcal{E}(\pi f_{D,\ell,n,\sigma}) - \mathcal{E}(f_\rho)) + c_3\left[\frac{n\ell\log m}{m}\right]^{\frac{1}{2}}\log\frac{4}{\theta}.$$

Thus,

$$\mathcal{E}(\pi f_{D,\ell,n,\sigma}) - \mathcal{E}(f_\rho) \leqslant c_4\left[D_{\rho_X}C(n\ell)^{-r/d} + m^{-1}\log(4/\theta) + \left(m^{-1}\log(4/\theta)\right)^{\frac{1}{2}} + (n\ell)^{\frac{1}{2}}\left(m^{-1}\log m\right)^{\frac{1}{2}}\log\frac{4}{\theta}\right],$$

for some constant $c_4 > 0$. Thus, if $n\ell \sim \left(\frac{m}{\log m}\right)^{\frac{d}{2r+d}}$, then

$$\mathcal{E}(\pi f_{D,\ell,n,\sigma}) - \mathcal{E}(f_\rho) \leqslant cD_{\rho_X}\left(\frac{m}{\log m}\right)^{-\frac{r}{2r+d}}\log\frac{4}{\theta}$$

for $c$ which is a positive constant . Therefore, Theorem 1 follows from (C1).

## Appendix D Toy simulations

In this part, we carry out a series of numerical simulations to study the effect of the algorithm parameters and verify our theoretical statements. The simulations are done with three purposes. In the first simulations, we investigate the effect of the tolerance of ADMM to select a suitable stopping criterion for our algorithm. In the second and the third simulations, we study the generalization abilities and training time of our methods with different sized training samples. In addition, we conduct experiments on training samples with different levels of noise. All the toy simulations and all of the experiments are performed on a desktop workstation with an Intel(R) Core i7-4790 3.6GHz CPU, 32G of RAM, Ubuntu 14.04 operating system and Matlab 2014a environment. The implementation of neural network is derived from the publicly available Tensorflow-1.4.0 framework by using AdamOptimizer.

## Appendix D.1 Experiment setting

### Appendix D.1.1 *Samples and labels*

***Sample A***: Given a discrete signal $x = [x_1, x_2]$ with dimension $N = 2$, the labels of the samples are defined by a nonlinear Bayes rule.

$$h(t) = ((1-5t)_+^5(32 * t^2 + 20t + 1) + 1)/2 \tag{D1}$$

We define the labels with the following rules: given a sample $x_i = [x_i(1), x_i(2)]$, its label $y_i = 1$ if $x_i(2) \geqslant h(x_i(1))$; and otherwise, $y_i = -1$. The generated data is shown in Figure D1 (a).

***Sample B***: To verify the stability and robustness of our model, we use the second synthetic dataset in all numerical experiments. This dataset consists of two interleaved classes which each one shows a sinusoid pattern and the details of the data are shown in Figure D1 (b). Different from ***Sample A***, this dataset adds some standard Gaussian noise to the input samples, which makes the two categories have no obvious boundary [1].

### Appendix D.1.2 *Implementation and Evaluation*

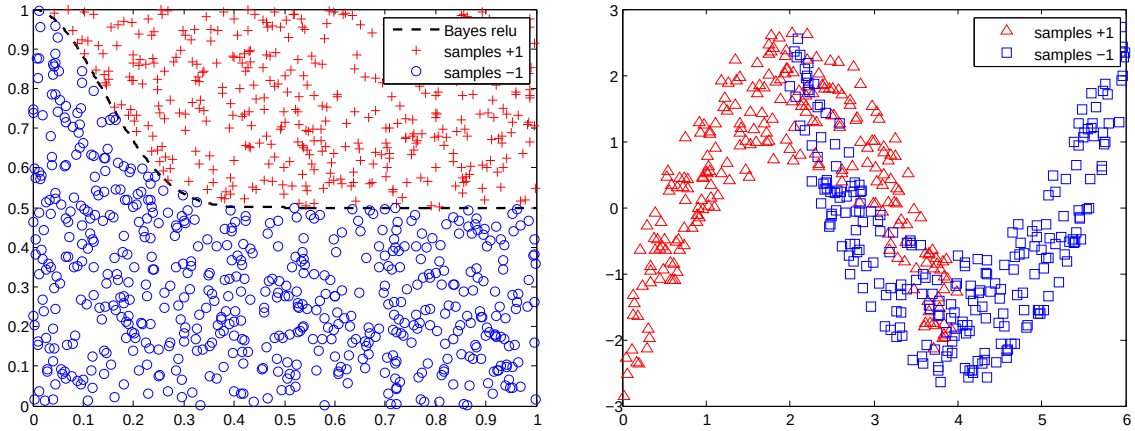
In this part, we conduct three simulations to analyze the effects of the parameters. First, we start with an experiment to study effect of the stopping criterion of ADMM. Then, the effects of different numbers of training samples on the performance of the algorithm are studied. Finally, we study the generalization of our algorithm on noisy data. We repeat each experiment 20 times and record the average training and test errors.

## Appendix D.2 Results

### Appendix D.2.1 *The effect of the tolerance of ADMM*

The initial parameters and tolerance of ADMM are two crucial factors for the optimization time and accuracy of our algorithms. In this part, we study the effects of these parameters on the algorithm and choose an appropriate stopping criterion for our algorithm. The numbers of training sets and test sets are set to $m_{train} = m_{test} = 1000$. For the initialization of ADMM, we set $p^0 = (0, y, 0)$, $\beta = \alpha = 1$ and the range of tolerance was selected from $10^{-7}$ to $10^0$ with a step size of $10^{-0.2}$.

The curves of test error, training error and the required maximal number of iterations are reported in Figure D2 and D3. It can be observed that the errors are stable when the tolerance is chosen from $10^{-4}$ to $10^0$. However, when the tolerance is less than $10^{-3}$, the required maximal iterations increases dramatically. Therefore, according to the test error and training time consumption, we select $Tolerance = 10^{-4}$ as the stopping criterion of ADMM.




(a) Data with boundary (b) Data without boundary

**Figure D1** The generated data used in simulations.

### Appendix D.2.2 *The effect of the number of training samples*

In this simulation, we mainly study the performance of our model under various sized training samples. We verify 14 training sets, and the number of samples used for training is in the form of $fix[10^r]$, where $fix[\cdot]$ denotes the integral function and $r$ is set as the equidistant points from 1.95 to 3.9 at a step size of 0.15. The range of the number of training samples is $[89, 7943]$.

Figure D4 and Figure D5 show the variation curves of the training error, test error and the maximum number of iterations required with the size of training samples. As reported in Figures D4 and D5, as the number of samples increases, the test error always decreases, and the required iterations increases at the beginning and decreases later. Furthermore, the required maximum number of iterations will eventually be maintained at a stable value. These results show that our algorithm tends to be stable as the number of samples increases. This is the main focus of the paper, our algorithm achieves a high efficiency in the training phase when the size of data increases.

### Appendix D.2.3 *The effect of different levels of noise*

In the third numerical experiment, we study the effect of noise on the proposed algorithm. The training error curve and test error curve with respect to the noise ratio are shown in Figures D6 and D7, respectively. We conduct experiments on samples with different noise ratios. For *Sample A* and *Sample B*, the noise samples are randomly distributed in $[0, 1] \times [0, 1]$ and $[0, 6] \times [0, 3]$, respectively. The setting of this simulation is the same as that of the previous sections. Specifically, we set the parameters $\alpha = 1$, $\beta = 1$, $tol = 0.0005$.

As shown in Figures D6 and D7, the training errors are almost proportional to the noise level, which means that our method is generally robust to different levels of noise. All the aforementioned simulations verify the correctness of our theoretical assertions in Section 4.

## Appendix E Real world data experiments

## Appendix E.1 UCI datasets experiments

### Appendix E.1.1 *Experiment setting*

In this part, we carry out a series of experiments on real world datasets. To make a reasonable comparison, we set the experimental parameters similar to those in literature [7]. The detailed settings are described as follows.

• Our method: We set $\alpha = \beta = 1$ and the initialization of ADMM is $p^0 = (0, y, 0)$. The the training time of the algorithm is considered when setting the tolerance. The tolerance is $tol = 5 \times 10^{-5}$ which generally ensures the convergence of the algorithm as shown in previous numerical simulations.

• SVM methods: The range of parameters $(c, g)$ are selected by the grid search of the region $[2^{-5}, 2^5] \times [2^{-5}, 2^5]$ in a logarithmic scale. The poly kernel $s$ is determined in the interval $[1, 10]$ with a step size of 1.

• Random forest: The number of trees is selected from a group of candidates based on experience, which is in the interval $[2, 20]$ with a step size of 2.

• Neural network: There are two parameters that affect the performance of the network, which are the number of neurons in the hidden layer and the learning rate. Empirically, we set the number of neurons to about twice the input dimension. For each dataset, we tried learning rates of 0.0001, 0.0005 and 0.001 on a fixed network architecture. The optimal network architectures and learning rates for each dataset are shown in Table E1.

Following the common evaluation procedure, we independently repeat each algorithm 20 times. The averages and standard deviations (std) of classification errors will be reported in the next section. We also report the average of training time and analyze the advantages of the proposed model.

### Appendix E.1.2 *Samples*

In these experiments, all the data are obtained from the UCI dataset: http:// archive.ics. uci.edu/ml/data sets.php. We include 6 problems covering different fields and summarize the detailed information of each dataset in Table E2 and # stands for the number of samples. In the following experiments, the training, verification and test sets accounted for 50%, 25% and 25% of the total samples, respectively.

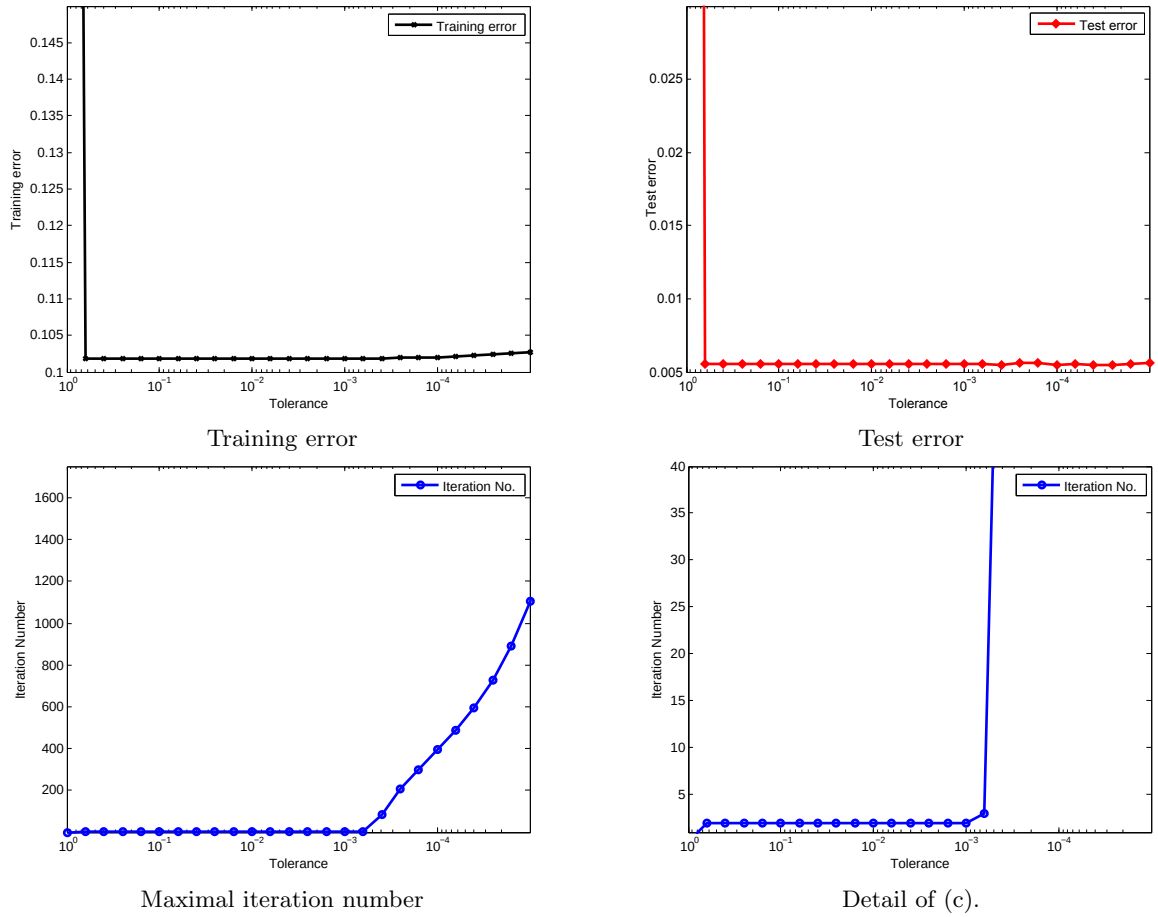



**Figure D2** The effect of tolerance of ADMM on *Sample A*.

**Table E1** Network architecture in UCI experiments

| Datasets | Network architectures | Learning rates |
|---|---|---|
| banknote | [4, 10, 2] | 0.0005 |
| seismic | [18, 40, 2] | 0.001 |
| musk2 | [166, 320, 2] | 0.0001 |
| HTRU2 | [8, 20, 2] | 0.001 |
| MAGIC | [10, 20, 2] | 0.001 |
| default | [24, 50, 2] | 0.0005 |

**Table E2** Information of UCI datasets.

| Datasets | Data size # | Attributions |
|---|---|---|
| banknote | 1372 | 4 |
| seismic | 2584 | 18 |
| musk2 | 6598 | 166 |
| HTRU2 | 17898 | 8 |
| MAGIC | 19020 | 10 |
| default | 30000 | 24 |

### Appendix E.1.3 *Competitors*

We compare the performance to typical SVM methods, the random forest (RF) and the neural network (NN). The difference between the SVM methods is the kernel function. Specifically, we set the kernel functions as the radial basis function (SVM-rbf) and the polynomial kernel function (SVM-poly). In terms of implementing these SVM methods, we use the well-known libsvm toolbox obtained from the following website: https://www.csie.ntu.edu.tw/ cjlin/libsvm/.

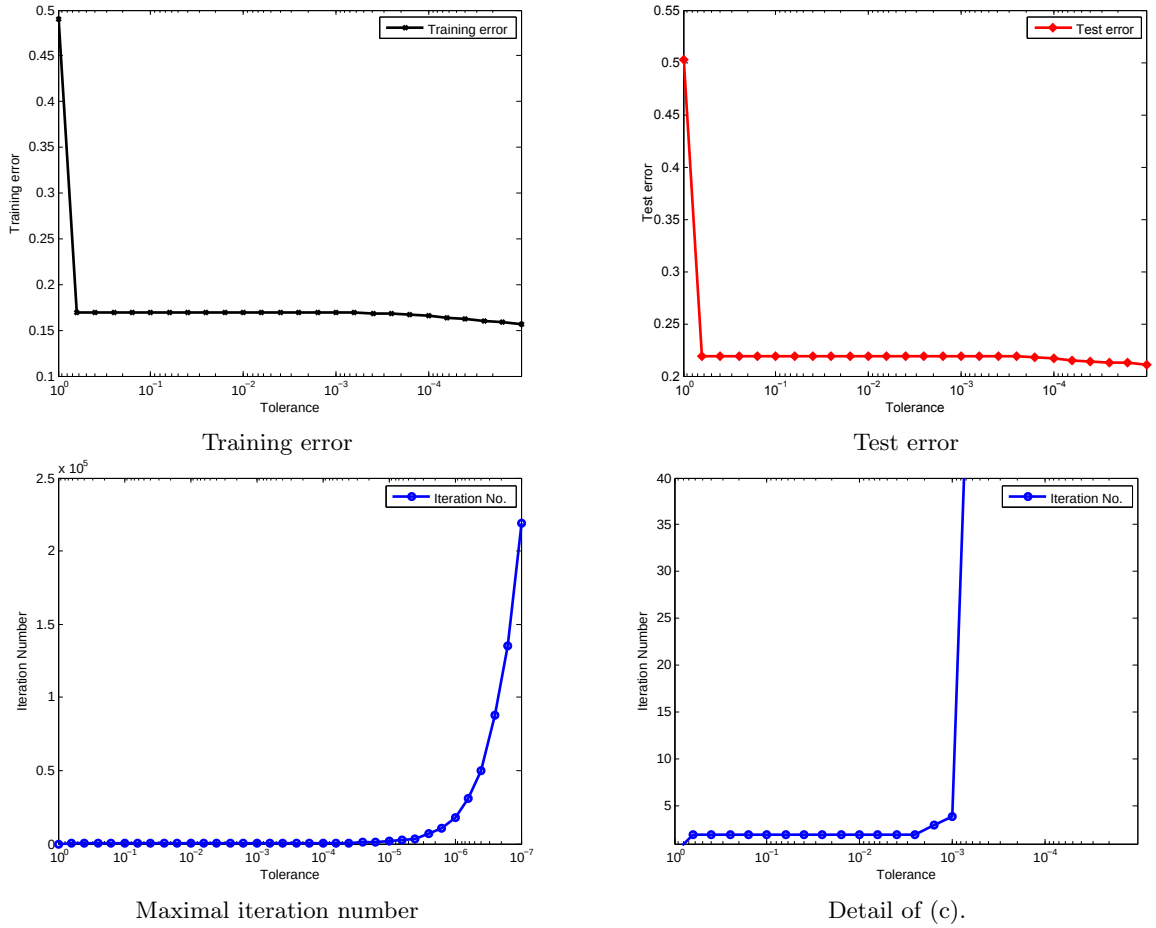



**Figure D3** The effect of tolerance of ADMM on *Sample B*.

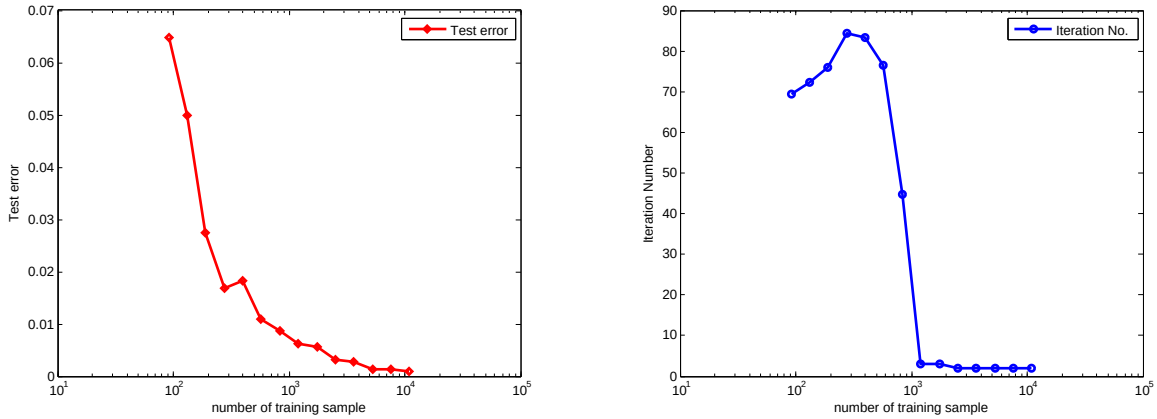



**Figure D4** The effect of different number of training sample on *Sample A*.

## Appendix E.1.4 *Experimental results*

The detailed training time and test accuracies are summarized in Tables E3 and E4. The results show that the classification accuracies of our model are similar to those of the neural network, and the training times are much shorter than those of the neural network which proves the validity of the proposed equivalent model. In addition, compared with the neural network, our model can converge in a shorter time because it does not need to solve a non-convex optimization problem. Furthermore, the test accuracies of our algorithm are higher than those of any other mentioned methods expect seismic, musk2 and HTRU2. For the seismic dataset, the test accuracy is better than that of any other method expect neural network. For musk2, the test accuracy is better than the SVM methods, the RF and the baseline provided by the UCI website. For HTRU2, the test accuracies of all methods are similar to that of the baseline. From Table E4, the most training time is significantly less than the traditional kernel algorithms. In a word, considering generalization ability and classification accuracy, most of these methods show better performance than the baseline. However, our method is superior to other methods in terms of the training efficiency. Therefore, the experimental results prove the

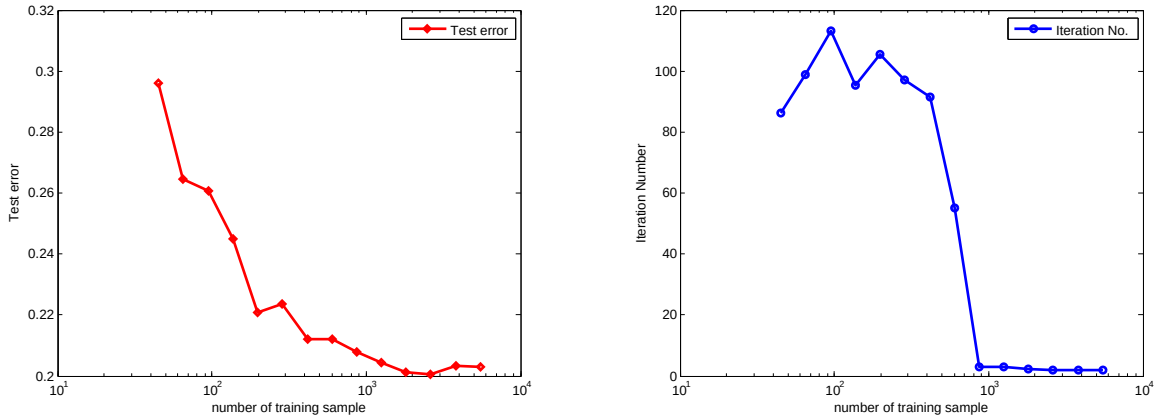



**Figure D5** The effect of different number of training sample on *Sample B*.

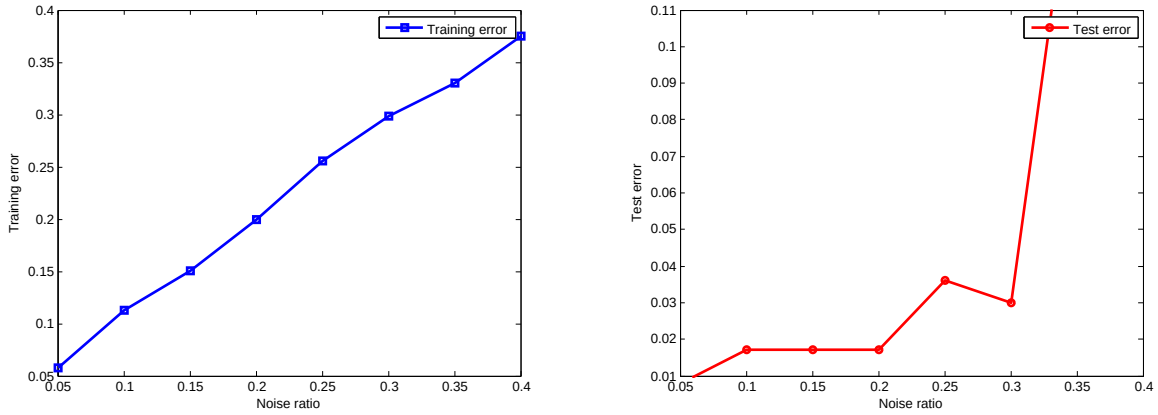



**Figure D6** The errors with different noise ratio on *Sample A*.

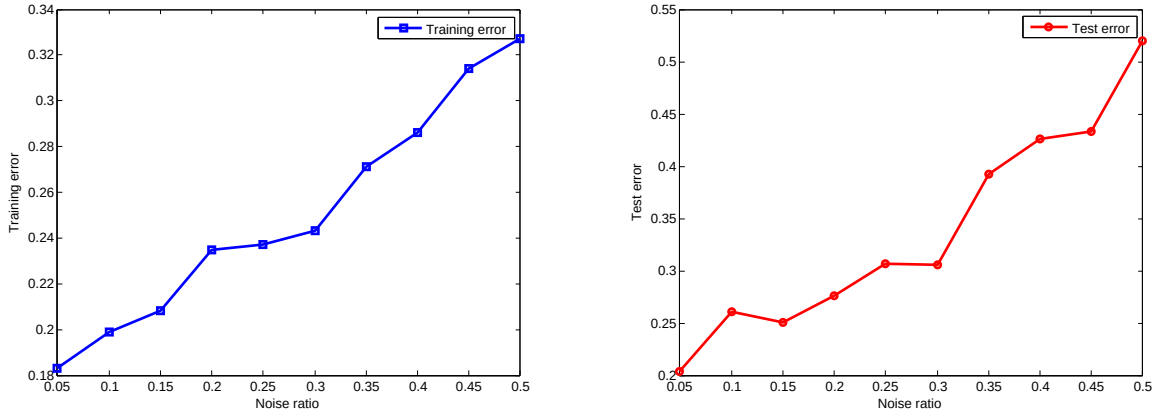



**Figure D7** The errors with different noise ratio on *Sample B*.

effectiveness of our model to tackle real world data. These results also show the potential of applying of our method in massive classification tasks.

## Appendix E.2 Massive data experiment

In the following, to verify the effectiveness of our algorithm in the actual application of massive data classification, we conduct an experiment on the super symmetry particles (SUSY) dataset. SUSY is a classification dataset that is used to distinguish signals from backgrounds. The number of samples in the dataset is 5 million. For each sample, the first 8 attribute characteristics are some kinematic properties of particles and the last 10 attribute characteristics are functions of the first 8. In general, the last 10 attribute characteristics are the high-level features produced by physicists to distinguish these two classes.

In this experiment, we divide the samples of the SUSY dataset into three parts. The numbers of samples in the training set, verification set and test set are $4,000,000$, $500,000$ and $500,000$, respectively. We compare it with 11 popular classification methods

**Table E3** The classification accuracies on real world massive dataset. The best and second best results are highlighted in bold and underlined, respectively.

| Datasets | SVM-rbf | SVM-poly | RF | NN | FPC | Ours | Baseline |
|---|---|---|---|---|---|---|---|
| banknote | 98.07 (0.71) | 97.72 (0.99) | 98.99 (0.57) | 98.81(0.35) | 98.15 (0.73) | **99.89** (0.10) | 95.81 |
| seismic | 93.84 (0.41) | 93.59 (0.74) | 92.88 (0.78) | **95.51(0.26)** | 93.68 (0.84) | 94.20 (0.69) | 88.00 |
| musk2 | 91.11 (0.41) | 92.82 (0.32) | 96.56 (0.51) | 95.58(0.77) | **99.08** (0.32) | 93.07 (0.99) | 90.30 |
| HTRU2 | 97.53 (0.06) | 97.42 (0.08) | 97.88(0.18) | 97.76 (0.28) | 97.26(0.23) | 97.99 (0.34) | **99.00** |
| MAGIC | 85.69 (0.08) | 86.00 (0.11) | 86.90(0.53) | 95.33 (0.49) | 85.10 (0.53) | **98.78** (0.34) | 86.34 |
| default | 81.60 (1.05) | 82.10 (0.32) | 81.01(0.40) | 81.63 (0.43) | 80.51 (0.29) | **82.93** (0.69) | 82.00 |

**Table E4** Comparison of average training time (in second).

| Datasets | SVM-rbf | SVM-poly | RF | NN | FPC | Ours |
|---|---|---|---|---|---|---|
| banknote | 8.35 | 6.07 | 1.21 | 10.51 | 0.05 | 0.047 |
| seismic | 30.12 | 12.94 | 1.86 | 20.31 | 0.04 | 0.605 |
| musk2 | 1350.4 | 845.89 | 7.36 | 31.47 | 6.44 | 0.025 |
| HTRU2 | 167.29 | 78.55 | 8.11 | 10.63 | 0.51 | 0.032 |
| MAGIC | 1026.9 | 12011.5 | 18.18 | 20.43 | 0.91 | 0.0325 |
| default | 17998.9 | 35902.2 | 30.65 | 20.35 | 0.43 | 0.0815 |

in terms of the test accuracy and the training time. These comparisons include the k-nearest neighbors (kNN) [2], stochastic gradient descent (SGD), Hoeffding tree (HT) [3] neural network (NN) some extension methods proposed in [4]. For neural network and our method, the detailed settings are described as follows.

• Our method: We set $\alpha = \beta = 1$ and the initial parameters of ADMM are $p^0 = (0, y, 0)$. The tolerance candidates are $\{5 \times 10^{-5.5}, 5 \times 10^{-5}, 5 \times 10^{-4.5}\}$.

• Neural network: The number of hidden neurons is chosen from a group of candidates, which are set empirically. Specifically, the candidates numbers of hidden neurons are $\{300, 600, 900, 1200, 1500\}$. For choosing learning rate, we tried values of 0.0001, 0.0005 and 0.001 on various networks with different widths (number of hidden neurons). As a result, there are 15 networks for testing. The batch size used to train each network is set to 512.

**Table E5** The classification accuracies on real world massive dataset. The best and second best results are highlighted in bold and underlined, respectively.

| Methods | kNN | SGD | HT | LB-HT | HT-kNN | kNN-F | SGD-F | LB-SGD-F | HT-SGD-F | NN | FPC | Ours |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy(%) | 67.5 | 76.5 | 78.2 | 78.7 | 77.2 | 71.2 | 77.7 | 77.7 | 78.4 | 82.7 | 79.0 | **83.2** |
| Training time (seconds) | 1,464 | 25 | 45 | 530 | 1,428 | 4,714 | 118 | 1,022.4 | 154.3 | 1250.44 | 732.8 | 604.2 |

We choose the parameters with the best performance and record the experimental results. As shown in Table E5, our method achieves the highest classification accuracy compared to the other methods. From the perspective of training time, our method has a shorter training time than most other methods. It should be noted that the first 10 algorithms were implemented on GPU while FPC and our model were implemented on a single CPU. Considering the hardware factors, our algorithm is also superior to the first 10 algorithms and FPC algorithm in training time. In a nutshell, our algorithm can classify massive data efficiently and accurately.

**References**


1 M. Culp, K. Johnson, G. Michailidis, Ada: An r package for stochastic boosting. Journal of Statistical Software, 2012, 17(2): 1-27.

2 A. Shaker, E. Hüllermeier. Instance-based classification and regression on data streams. Learning in Non-Stationary Environments. Springer, New York, NY, 2012: 185-201.

3 P. Domingos, G. Hulten. Mining high-speed data streams. Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2000: 71-80.

4 D. Marrón, J. Read, A. Bifet, N. Navarro, Data stream classification using random feature functions and novel method combinations. Journal of Systems and Software, 2015, 127: 211-216.

5 J. Fang, S. Lin, Z. Xu. Learning through deterministic assignment of hidden parameters. IEEE Transactions on Cybernetics, 2020, 50(5): 2321-2334.

6 T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. The Annals of Stats, 2004, 32(1): 56-85.

7 J. Zeng, M. Wu, S. Lin, D. Zhou. Fast polynomial kernel classification for massive data. arXiv preprint arXiv:1911.10558, 2019.

8 Y. Nesterov. Smooth minimization of non-smooth functions. Mathematical Programming, 2005, 103(1): 127-152.