Solving Prediction Games with Parallel Batch Gradient Descent

Michael Großhans[1⋆] and Tobias Scheffer[2]

[1] freiheit.com technologies gmbh, Hamburg, Germany
michael.grosshans@freiheit.com
[2] University of Potsdam, Department of Computer Science, Potsdam, Germany
scheffer@cs.uni-potsdam.de

Abstract. Learning problems in which an adversary can perturb instances at application time can be modeled as games with data-dependent cost functions. In an equilibrium point, the learner's model parameters are the optimal reaction to the data generator's perturbation, and vice versa. Finding an equilibrium point requires the solution of a difficult optimization problem for which both, the learner's model parameters and the possible perturbations are free parameters. We study a perturbation model and derive optimization procedures that use a single iteration of batch-parallel gradient descent and a subsequent aggregation step, thus allowing for parallelization with minimal synchronization overhead.

1 Introduction

In many security-related applications, the assumption that training data and data at application time are identically distributed is routinely violated. For instance, new malware is designed to bypass detection methods which their designers believe virus and malware scanners to employ, and email spamming tools allow their users to develop templates of randomized messages that produce a low spam score with current filters. In these examples, the party that creates the predictive model and the data-generating party factor the possible actions of their opponent into their decisions. This interaction can be modeled as a *prediction game* in which one player controls the predictive model whereas another exercises some control over the process of data generation.

Robust learning methods have been derived under the *zero-sum assumption* that the loss of one player is the gain of the other, for several types of adversarial feature transformations [17,11,23,12,13,7,8,24]. Settings in which both players have individual cost functions—a fraudster's profit is not the negative of an email service provider's goal of achieving a high spam recognition rate at close-to-zero false positives—cannot adequately be modeled as zero-sum games.

When the learner has to act first and model parameters are disclosed to the data generator, this non-zero-sum interaction can be modeled as a Stackelberg

⋆ This work was done while the author was at the University of Potsdam, Germany

competition [4,15]. A Stackelberg competition always has an optimal solution, but generally a difficult bi-level optimization problem has to be solved to find it [4]. For simultaneously acting players, one may resort to the concept of a Nash equilibrium. An equilibrium is a pair of a predictive model and a transformation of the input distribution. In an equilibrium point, the learner's model parameters are the optimal reaction to the data generator's perturbation function, and vice versa. If a game has a unique Nash equilibrium and is played by rational players that aim at minimizing their costs, it may be reasonable for each player to assume that the opponent will play according to the Nash equilibrium strategy as well. If, however, multiple equilibria exist and the players choose their strategy according to distinct ones, then the resulting combination may be arbitrarily disadvantageous for either player. For certain cost functions, the prediction game has been shown to have a unique Nash equilibrium [3].

Finding the equilibrium point of a prediction game requires the solution of a difficult optimization problem: in each iteration of an outer gradient-descent, nested optimization problems have to be solved. This process is two orders of magnitude more expensive than *iid* learning [3]—even more so, if the learner is uncertain about the adversary's cost function [14].

Gradient descent algorithms can be parallelized by distributing the data in batches across multiple worker nodes. Casting gradient descent into the MapReduce programming model [6] offers an almost unlimited potential speed-up, because synchronization is limited to a final *reduce* step, and, unlike multicore or GPU parallelism, MapReduce is not constrained by the limited number of cores that can be fitted into a single unit of computing hardware. In order to conduct gradient descent within the MapReduce model, parallel nodes have to perform gradient descent on subsets of the data. Only in the final step, the local parameter vectors are aggregated [18,26]. This procedure has known convergence bounds [26].

Work on HaLoop [5] and ScalOps [25] aims at allowing for more flexible algorithm design that may include aggregation steps during the parallel optimization process [19]; this, however comes at the cost at higher communication costs which limit the gain of increased parallelization. This paper therefore focuses on rephrasing the search for an equilibrium point of a prediction game within the MapReduce model.

The known analysis and algorithm for finding the equilibrial prediction model [3] are based on a model of the adversarial data transformation that allows the perturbation of each instance to potentially depend on other instances. It is therefore unsuitable for parallelization: When the perturbation of an instance may depend on different instances, a node that does not have access to all interdependent instances cannot anticipate the outcome of the adversary's action. Therefore, we derive a model of adversarial manipulations of the input distribution that is based on the manipulation of individual feature vectors.

The rest of this paper is organized as follows. Section 2 lays out the problem setting and introduces an adversarial data generation model. In Section 3 we study conditions under which a unique equilibrium point exists under this data

generation model. We derive a method for finding the unique equilibrium point in a way that can be parallelized in Section 4. Section 5 presents empirical result and Section 6 concludes.

2 Problem Setting and Data Transformation Model

We study static prediction games between two players: The *learner* and its adversary, the *data generator*. For example, in email spam filtering, the learner may be an email service provider whereas the data generator is an amalgamated model of all legitimate and abusive email senders.

At *training time*, the data generator produces a matrix $\mathbf{X}$ of training instances $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and a corresponding vector $\mathbf{y}$ of class labels $y_i \in \mathcal{Y}$. These object-class pairs are drawn according to a training distribution with density function $p(\mathbf{x}, y)$.

The task of the learner is to select the parameters $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^m$ of a linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$. Simultaneously, the data generator can choose a parameterized transformation $g_{\mathbf{A}} : \mathbb{R}^m \to \mathbb{R}^m$, with $\mathbf{A} \in \mathcal{A}$ that perturbs instances; regularizer $\rho_g \Omega_g(\mathbf{A})$ quantifies *transformation costs* which the data generator incurs. For instance, a spammer may obfuscate text messages and remove conspicuous URLs at the cost of reducing the response rate. At test time, instances are drawn according to $p(\mathbf{x}, y)$ and perturbed by $g_{\mathbf{A}}$; this defines the test distribution.

The learner's theoretical *costs* at application time are given by Equation 1; the data generator's theoretical costs by Equation 2.

$$\theta_f(\mathbf{w}, \mathbf{A}) = \sum_{y \in \mathcal{Y}} \int \ell_f(f_{\mathbf{w}}(g_{\mathbf{A}}(\mathbf{x})), y) p(\mathbf{x}, y) \mathrm{d}\mathbf{x} \tag{1}$$

$$\theta_g(\mathbf{w}, \mathbf{A}) = \sum_{y \in \mathcal{Y}} \int \ell_g(f_{\mathbf{w}}(g_{\mathbf{A}}(\mathbf{x})), y) p(\mathbf{x}, y) \mathrm{d}\mathbf{x} + \rho_g \Omega_g(\mathbf{A}) \tag{2}$$

The theoretical costs of both players depend on the unknown test distribution; we will therefore resort to regularized, empirical costs based on the training sample. The empirical costs incurred by the predictive model $f_{\mathbf{w}}$ and transformation $g_{\mathbf{A}}$ are

$$\hat{\theta}_f(\mathbf{w}, \mathbf{A}) = \frac{1}{n} \sum_{i=1}^{n} \ell_f(\mathbf{w}^\mathsf{T} g_{\mathbf{A}}(\mathbf{x}), y_i) + \rho_f \Omega_f(\mathbf{w})$$

$$\hat{\theta}_g(\mathbf{w}, \mathbf{A}) = \frac{1}{n} \sum_{i=1}^{n} \ell_g(\mathbf{w}^\mathsf{T} g_{\mathbf{A}}(\mathbf{x}), y_i) + \rho_g \Omega_g(\mathbf{A}) .$$

We employ a linear, parameterized data transformation model of the form $g_{\mathbf{A}}(\mathbf{x}) = \mathbf{x} + \mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{m \times m}$ is the transformation matrix chosen by the adversary. Under this model, the perturbation vector that is added to each instance $\mathbf{x}$ is a linear function of $\mathbf{x}$. For $\mathbf{A} = 0$, instances remain unperturbed.

This model subsumes many relevant data-manipulation operations. For instance, features are scaled by nonzero values at the diagonal elements of $\mathbf{A}$; features i are deleted by $a_{ii} = -1$. Feature i is replaced by feature j (*e.g.,* `Viagra` $\rightarrow$ `V1agra`) by a matrix that has entries $a_{ii} = -1$ and $a_{ji} = 1$, and is 0 everywhere else.

We will write the transformation matrix as vector of m-dimensional row vectors $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]^\mathsf{T}$ or as an m^2-dimensional vector $\mathbf{a} = \left[\mathbf{a}_1^\mathsf{T}, \ldots, \mathbf{a}_m^\mathsf{T}\right]^\mathsf{T}$ whenever this will simplify the notation.

Under this data transformation model, standard ℓ_2 regularization for the data generator [3] would amount to

$$\|(\mathbf{x}_i + \mathbf{A}\mathbf{x}_i) - \mathbf{x}_i\|^2 = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{A}\mathbf{x}_i\|^2$$

which is not strongly convex in $\mathbf{A}$ for every data matrix $\mathbf{X}$. Hence, this regularizer can have multiple optima, which should be avoided. Therefore, we use the Frobenius norm of $\mathbf{A}$ as regularizer for the data generator; we use standard ℓ_2 regularization for the learner:

$$\Omega_f(\mathbf{w}) = \|\mathbf{w}\|^2, \tag{3}$$

$$\Omega_g(\mathbf{A}) = \frac{1}{2}\|\mathbf{A}\|_F^2 = \frac{1}{2}\|\mathbf{a}\|^2. \tag{4}$$

3 Analysis of Equilibrium Points

Note that both $\hat{\theta}_f$ and $\hat{\theta}_g$ depend on both players' actions. Neither player can minimize their costs without considering their adversary's options. This motivates the concept of an equilibrium point. Assume that the learner considers using model parameters $\mathbf{w}_1$. The learner can now determine a possible reaction $\mathbf{A}_1$ of the data generator that would minimize $\hat{\theta}_g$ for the given $\mathbf{w}_1$. In turn, the learner can determine model parameters $\mathbf{w}_2$ that would minimize $\hat{\theta}_f$ for this transformation $\mathbf{A}_1$, continue to determine reaction $\mathbf{A}_2$, and so on. It his sequence of reactions reaches a fixed point—a point $(\mathbf{w}^*, \mathbf{A}^*)$ that is the best possible reaction to itself—then this point is a Nash equilibrium and satisfies

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \hat{\theta}_f(\mathbf{w}, \mathbf{A}^*), \tag{5}$$

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \hat{\theta}_g(\mathbf{w}^*, \mathbf{A}). \tag{6}$$

In this section, we analyze the prediction game between learner and data generator that we have introduced in the previous section. We will derive conditions under which equilibrium points exist, and conditions under which an equilibrium point is unique.

Known results on the existence of equilibrium points for prediction games [3] do not apply to the data transformation model derived in Section 2: Equation 4 regularizes $\mathbf{A}$ because regularization of $||\mathbf{X} - g_\mathbf{A}(\mathbf{X})||$ would not be convex in $\mathbf{A}$, and therefore Assumption 3 of [3] is not met.

3.1 Existence of Equilibrium Points

We will now study under which conditions the prediction game between learner and data generator with the data transformation introduced above has at least one equilibrium point. We start by formulating conditions on action spaces and loss functions in the following assumption.

Assumption 1 *The players' action sets $\mathcal{W}$ and $\mathcal{A}$ and loss functions ℓ_f and ℓ_g satisfy the following properties.*

1. *Action spaces $\mathcal{W} \subseteq \mathbb{R}^m$ and $\mathcal{A} \subseteq \mathbb{R}^m \times \ldots \times \mathbb{R}^m$ are non-empty, compact and convex,*
2. *The loss functions $\ell_f(z, y)$ and $\ell_g(z, y)$ are convex and continuous in z for every $y \in \mathcal{Y}$*

Theorem 1. *Under Assumption 1 the game has at least one equilibrium point.*

Proof. By Assumption 1 the loss functions $\ell_f(z_i, y_i)$ and $\ell_g(z_i, y_i)$ are continuous and convex in z_i for any $y_i \in \mathcal{Y}$. Note that $z_i = \mathbf{w}^\mathsf{T}\mathbf{x}_i + \mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{x}_i$ is linear in $\mathbf{w} \in \mathbb{R}^m$ and linear in $\mathbf{A} \in \mathbb{R}^{m^2}$ for any $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$. Hence, for both $\nu \in \{f, g\}$, the sum of loss terms $\sum_{i=1}^n \ell_\nu(z_i, y_i)$ is jointly continuous in $(\mathbf{w}, \mathbf{A}) \in \mathbb{R}^{m\times(m+1)}$ and convex in both $\mathbf{w} \in \mathbb{R}^m$ and $\mathbf{A} \in \mathbb{R}^{m\times m}$. Both regularizers Ω_f and Ω_g are jointly continuous in $(\mathbf{w}, \mathbf{A}) \in \mathbb{R}^{m\times(m+1)}$. Additionally Ω_f is strictly convex in $\mathbf{w} \in \mathbb{R}^m$ and Ω_g is strictly convex in $\mathbf{A} \in \mathbb{R}^{m\times m}$.

Hence, both empirical cost functions $\hat{\theta}_f$ and $\hat{\theta}_g$ are jointly continuous in $(\mathbf{w}, \mathbf{A}) \in \mathbb{R}^{m\times(m+1)}$. Additionally $\hat{\theta}_f$ is strictly convex in $\mathbf{w} \in \mathbb{R}^m$ and $\hat{\theta}_g$ is strictly convex in $\mathbf{A} \in \mathbb{R}^{m\times m}$. Therefore by Theorem 4.3. in [2]—together with the fact that both action spaces are non-empty, compact and convex—at least one equilibrium point exists.

3.2 Uniqueness of Equilibrium Points

In this section, we will derive conditions for uniqueness of equilibrium points. The significance of this result is that an action that is part of an equilibrium point minimizes the costs for either player only if the opponent chooses the same equilibrium point. Otherwise, either player's costs may be arbitrarily high. If multiple equilibria exist, the players cannot determine which action even a perfectly rational opponent will take. We will make use of a theorem of Rosen [22] which states that a unique equilibrium point exists if the Jacobian of the combined loss

$$r_\mathbf{w}\theta_f(\mathbf{w}, \mathbf{A}) + r_\mathbf{A}\theta_g(\mathbf{w}, \mathbf{A})$$

is positive definite for any fixed $r_\mathbf{w} > 0, r_\mathbf{A} > 0$. To prove this condition, we formulate two lemmas. Lemma 1 and Lemma 2 derive two different forms of matrices that are always positive (semi-)definite. In the following, the symbol $\otimes$ denotes the Kronecker-product.

Lemma 1. *For any* $\mathbf{A} \in \mathbb{R}^{m \times m}$ *and* $\mathbf{w} \in \mathbb{R}^m$ *and any positive semi-definite matrix* $\mathbf{X} \in \mathbb{R}^{m \times m}$*, the matrix*

$$\mathbf{M}_1 := \begin{bmatrix} \mathbf{A}\mathbf{X}\mathbf{A}^\mathsf{T} & \mathbf{w}^\mathsf{T} \otimes (\mathbf{A}\mathbf{X}) \\ \mathbf{w} \otimes (\mathbf{X}\mathbf{A}^\mathsf{T}) & (\mathbf{w}\mathbf{w}^\mathsf{T}) \otimes \mathbf{X} \end{bmatrix} \in \mathbb{R}^{(m^2+m)\times(m^2+m)}$$

is positive semi-definite.

Proof. Note that we can rewrite this matrix as a product of three matrices:

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{w} \otimes \mathbf{I}_m \end{bmatrix} \mathbf{X} \begin{bmatrix} \mathbf{A}^\mathsf{T} \, \mathbf{w}^\mathsf{T} \otimes \mathbf{I}_m \end{bmatrix}^\mathsf{T}$$

where $\mathbf{I}_m$ denotes the $m \times m$ unit matrix. By Assumption 1 the matrix $\mathbf{X}$ is positive semi-definite and therefore the product $\mathbf{v}_1^\mathsf{T}\mathbf{X}\mathbf{v}_1 \geq 0$ is non-negative for all vectors $\mathbf{v}_1 \in \mathbb{R}^m$. By using the substitution $\mathbf{v}_1 = \left[\mathbf{A}^\mathsf{T} \, \mathbf{w}^\mathsf{T} \otimes \mathbf{I}_m\right] \mathbf{v}_2$, the product

$$\mathbf{v}_2^\mathsf{T} \begin{bmatrix} \mathbf{A} \\ \mathbf{w} \otimes \mathbf{I}_m \end{bmatrix} \mathbf{X} \begin{bmatrix} \mathbf{A}^\mathsf{T} \, \mathbf{w}^\mathsf{T} \otimes \mathbf{I}_m \end{bmatrix}^\mathsf{T} \mathbf{v}_2 = \mathbf{v}_1^\mathsf{T}\mathbf{X}\mathbf{v}_1 \geq 0$$

is non-negative. Hence, the matrix $\mathbf{M}_1$ is positive semi-definite, which completes the proof.

Lemma 2. *For any* $\mathbf{x} \in \mathbb{R}^m$ *and any* $a, b \in \mathbb{R}^+$ *the matrix*

$$\mathbf{M}_2 := \begin{bmatrix} a\mathbf{I}_m & \mathbf{I}_m \otimes \mathbf{x}^\mathsf{T} \\ \mathbf{I}_m \otimes \mathbf{x} & b\mathbf{I}_{m^2} \end{bmatrix} \in \mathbb{R}^{(m^2+m)\times(m^2+m)}$$

is positive definite, if and only if $a \cdot b > \mathbf{x}^\mathsf{T}\mathbf{x}$.

Proof. The matrix is a symmetric square matrix. Hence it is positive definite if and only if all eigenvalues $\lambda_i > 0$ for all $i \in \{0, \ldots, m^2 + m\}$. Let $\left(\mathbf{w}^\mathsf{T}, \mathbf{v}_1^\mathsf{T}, \ldots, \mathbf{v}_m^\mathsf{T}\right)^\mathsf{T}$ be an arbitrary eigenvector with eigenvalue λ and let us define

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1^\mathsf{T} \\ \vdots \\ \mathbf{v}_m^\mathsf{T} \end{bmatrix}.$$

Then—by using the definition of eigenvectors—the following equations hold:

$$(\lambda - a)\mathbf{w} = \mathbf{V}\mathbf{x} \tag{7}$$

$$(\lambda - b)\mathbf{V} = \mathbf{w}\mathbf{x}^\mathsf{T}. \tag{8}$$

By combining Equation 7 and Equation 8 the following equation

$$(\lambda - a)(\lambda - b)\mathbf{w} = \mathbf{x}^\mathsf{T}\mathbf{x}\mathbf{w} \tag{9}$$

holds for every eigenvector $\left(\mathbf{w}^\mathsf{T}, \mathbf{v}_1^\mathsf{T}, \ldots, \mathbf{v}_m^\mathsf{T}\right)^\mathsf{T}$ with corresponding eigenvalue λ.

Firstly, assume that $\mathbf{w} = \mathbf{0}$ holds. Due to the definition of an eigenvector, the matrix $\mathbf{V} \neq \mathbf{0}$ is non-zero. By Equation 8, the corresponding eigenvalue would be $\lambda = b$, and hence the corresponding eigenvalue would be positive.

Now assume that $\mathbf{w} \neq \mathbf{0}$ holds. Then, by using Equation 9 it turns out that $(\lambda - a)(\lambda - b) = \mathbf{x}^\mathsf{T}\mathbf{x}$. Solving this Equation for λ results in the following two solutions:

$$\lambda_{1,2} = \frac{a+b}{2} \pm \sqrt{\frac{(a-b)^2}{4} + \mathbf{x}^\mathsf{T}\mathbf{x}}.$$

Therefore, matrix $\mathbf{M}_2$ is positive semi-definite if and only if

$$\frac{a+b}{2} \geq \sqrt{\frac{(a-b)^2}{4} + \mathbf{x}^\mathsf{T}\mathbf{x}} \tag{10}$$

holds, which is equivalent to the inequality

$$\frac{(a+b)^2}{4} \geq \frac{(a-b)^2}{4} + \mathbf{x}^\mathsf{T}\mathbf{x}.$$

Hence, the smallest eigenvalue is non-negative if and only if $a \cdot b > \mathbf{x}^\mathsf{T}\mathbf{x}$ which completes the proof.

We can now formulate Assumptions under which a unique equilibrium point exists.

Assumption 2 *For a given data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ and labels $\mathbf{y} \in \mathcal{Y}^n$, the players' action sets $\mathcal{W}$ and $\mathcal{A}$, loss functions ℓ_f and ℓ_g, and regularization parameters ρ_f, ρ_g satisfy the following properties.*

1. *the second derivatives of the loss functions are equal for all $y \in \mathcal{Y}$ and $z \in \mathbb{R}$*

$$\ell_f''(z, y) = \ell_g''(z, y).$$

2a. *The regularization parameters satisfy*

$$\rho_f \rho_g > \sup_{(\mathbf{w},\mathbf{A}) \in \mathcal{W} \times \mathcal{A}} \bar{\mathbf{x}}_{(\mathbf{w},\mathbf{A},\mathbf{X},\mathbf{y})}^\mathsf{T} \bar{\mathbf{x}}_{(\mathbf{w},\mathbf{A},\mathbf{X},\mathbf{y})},$$

where $\bar{\mathbf{x}}$ is the (derivate-) weighted average over all instances

$$\bar{\mathbf{x}}_{(\mathbf{w},\mathbf{A},\mathbf{X},\mathbf{y})} = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{1}{2} \left(\ell_f'(\mathbf{w}^\mathsf{T} \phi_\mathbf{A}(\mathbf{x}_i), y_i) + \ell_g'(\mathbf{w}^\mathsf{T} \phi_\mathbf{A}(\mathbf{x}_i), y_i) \right) \mathbf{x}_i \right].$$

2b. *(Sufficient condition for 2a) the regularization parameters satisfy*

$$\rho_f \rho_g > \sup_{(\mathbf{w},\mathbf{A}) \in \mathcal{W} \times \mathcal{A}} \max_{i \in \{1,\ldots,n\}} \tau_i^2(\mathbf{w}, \mathbf{A}) \cdot \mathbf{x}_i^\mathsf{T} \mathbf{x}_i,$$

where $\tau_i(\mathbf{w}, \mathbf{A})$ is specified by

$$\tau_i(\mathbf{w}, \mathbf{A}) = \frac{1}{2} \left(\ell_{pL}'(\mathbf{w}^\mathsf{T} \phi_\mathbf{A}(\mathbf{x}_i), y_i) + \ell_g'(\mathbf{w}^\mathsf{T} \phi_\mathbf{A}(\mathbf{x}_i), y_i) \right).$$

Theorem 2. *Under Assumptions 1 and 2, the prediction game between learner and data generator has exactly one equilibrium point.*

The conditions of Assumption 1 impose technical, rather common requirements on the cost functions that can be met in practice. The first condition of Assumption 2 requires the loss function of learner and data generator to have identical curvatures. This can be met, for instance, if both player use a logistic loss function [3]. The second condition imposes a joint bound on the regularization coefficients. Intuitively, if the data generator is allowed to perturb instances strongly, then a unique equilibrium exists only if the learner's cost function has a sufficiently large regularization term.

Proof. By Assumption 1 the game has at least one equilibrium point. We now turn towards the uniqueness of the equilibrium point. Therefore—by following the theorems in [10,22]—we show that the pseudo-Jacobian

$$\mathbf{J}_{r_{\mathbf{w}},r_{\mathbf{A}}}(\mathbf{w},\mathbf{A}) = \begin{bmatrix} r_{\mathbf{w}}\mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & r_{\mathbf{A}}\mathbf{I}_{m^2} \end{bmatrix} \begin{bmatrix} \nabla^2_{\mathbf{w},\mathbf{w}}\hat{\theta}_f & \nabla^2_{\mathbf{w},\mathbf{a}_1}\hat{\theta}_g & \cdots & \nabla^2_{\mathbf{w},\mathbf{a}_m}\hat{\theta}_f \\ \nabla^2_{\mathbf{a}_1,\mathbf{w}}\hat{\theta}_g & \nabla^2_{\mathbf{a}_1,\mathbf{a}_1}\hat{\theta}_g & \cdots & \nabla^2_{\mathbf{a}_1,\mathbf{a}_m}\hat{\theta}_g \\ \vdots & \vdots & \ddots & \vdots \\ \nabla^2_{\mathbf{a}_m,\mathbf{w}}\hat{\theta}_g & \nabla^2_{\mathbf{a}_m,\mathbf{a}_1}\hat{\theta}_g & \cdots & \nabla^2_{\mathbf{a}_m,\mathbf{a}_m}\hat{\theta}_g \end{bmatrix}$$

is positive definite at every point $(\mathbf{w},\mathbf{A}) \in \mathcal{W}\times\mathcal{A}$ for some fixed $r_{\mathbf{w}}, r_{\mathbf{A}} > 0$. We set $r_{\mathbf{w}} = r_{\mathbf{A}} = 1$. Therefore the pseudo-Jacobian (first and second derivations can be found in the Appendix) is given as

$$\begin{aligned} \mathbf{J}_r(\mathbf{w},\mathbf{A}) = &\begin{bmatrix} (\mathbf{I}_m + \mathbf{A})\,\mathbf{X}\boldsymbol{\Gamma}_f\mathbf{X}^\mathsf{T}\,(\mathbf{I}_m + \mathbf{A})^\mathsf{T} & \mathbf{w}^\mathsf{T} \otimes \left[(\mathbf{I}_m + \mathbf{A})\,\mathbf{X}\boldsymbol{\Gamma}_f\mathbf{X}^\mathsf{T}\right] \\ \mathbf{w} \otimes \left[\mathbf{X}\boldsymbol{\Gamma}_g\mathbf{X}^\mathsf{T}\,(\mathbf{I}_m + \mathbf{A})^\mathsf{T}\right] & \left[\mathbf{w}\mathbf{w}^\mathsf{T}\right] \otimes \left[\mathbf{X}\boldsymbol{\Gamma}_g\mathbf{X}^\mathsf{T}\right] \end{bmatrix} \\ &+ \begin{bmatrix} \rho_f\mathbf{I}_m & \mathbf{I}_m \otimes [\mathbf{X}\boldsymbol{\gamma}_f]^\mathsf{T} \\ \mathbf{I}_m \otimes [\mathbf{X}\boldsymbol{\gamma}_g] & \rho_g\mathbf{I}_{m^2} \end{bmatrix}. \end{aligned} \tag{11}$$

Following Assumption 2(1) the matrices $\boldsymbol{\Gamma}_f = \boldsymbol{\Gamma}_g$ are equal. Additionally, according to Assumption 1(2) the loss functions are convex and, therefore, the matrices $\boldsymbol{\Gamma}_f$ and $\boldsymbol{\Gamma}_g$ are positive semi-definite. Hence, the matrices $\mathbf{X}\boldsymbol{\Gamma}_f\mathbf{X}^\mathsf{T}$ and $\mathbf{X}\boldsymbol{\Gamma}_g\mathbf{X}^\mathsf{T}$ are equal and positive semi-definite. Following Lemma 1 the first summand in Equation 11 is positive semi-definite.

The second summand is positive definite if and only if the square matrix

$$\begin{bmatrix} \rho_f\mathbf{I}_m & \mathbf{I}_m \otimes \left[\frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_f + \frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_g\right]^\mathsf{T} \\ \mathbf{I}_m \otimes \left[\frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_f + \frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_g\right] & \rho_g\mathbf{I}_{m^2} \end{bmatrix}$$

is positive definite. According to Lemma 2 this square matrix is positive definite if and only if

$$\rho_f\rho_g > \left[\frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_f + \frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_g\right]^\mathsf{T}\left[\frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_f + \frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_g\right].$$

Note that the relation

$$\frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_f + \frac{1}{2}\mathbf{X}\boldsymbol{\gamma}_g = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\left(\ell'_f(\mathbf{w}^\mathsf{T}\mathbf{x}_i + \mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{x}_i, y_i) + \ell'_g(\mathbf{w}^\mathsf{T}\mathbf{x}_i + \mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{x}_i, y_i)\right)\mathbf{x}_i\right] \quad (12)$$

holds. Hence, according to Assumption 2(2a) the second summand in Equation 11 and therefore the pseudo-Jacobian $\mathbf{J}_{r_\mathbf{w}, r_\mathbf{A}}(\mathbf{w}, \mathbf{A})$ is positive definite at every point $(\mathbf{w}, \mathbf{A}) \in \mathcal{W} \times \mathcal{A}$, which completes the proof.

4 Finding the Unique Equilibrium Point Efficiently

In the previous section, we derived conditions for the existence of unique equilibrium points. In this section, we will discuss algorithms that find this unique solution and can be phrased as a single iteration of a parallel map step and a reduce step that aggregates the results.

4.1 Inexact Line Search

Equilibrium points can be located by inexact line search [16,3]. In each iteration, the procedure computes the response $\bar{\mathbf{w}}$ of the learner that minimizes $\hat{\theta}_f$ given the previous transformation $\mathbf{A}$, and response $\bar{\mathbf{A}}$ of the data generator that minimizes $\hat{\theta}_g$ given the previous prediction model $\mathbf{w}$ in nested optimization problems using L-BFGS [3]. The descent directions are then given by:

$$\mathbf{d}_f = \bar{\mathbf{w}} - \mathbf{w},$$
$$\mathbf{d}_g = \bar{\mathbf{A}} - \mathbf{A}.$$

Inexact line search tries increasingly large values of the step size t and perform an update by adding $t\mathbf{d}_f$ to the learner's prediction model $\mathbf{w}$ and by adding $t\mathbf{d}_g$ to the data generator's transformation matrix $\mathbf{A}$. This procedure converges to the unique Nash equilibrium—von Heusinger and Kanzow discuss its convergence properties [16].

4.2 Arrow-Hurwicz-Uzawa Method

Inexact line search is computationally expensive because it solves nested optimization problems in each iteration. In this section, we derive an alternative approach without nested optimization problems; it is based on the Arrow-Hurwicz-Uzawa saddle-point method. Equations 5 and 6 define equilibrium points. We start our derivation introducing the Nikaido-Isoda function [21]:

$$\hat{\theta}([\mathbf{w}_1, \mathbf{A}_1], [\mathbf{w}_2, \mathbf{A}_2]) = \left[\hat{\theta}_f(\mathbf{w}_1, \mathbf{A}_1) - \hat{\theta}_f(\mathbf{w}_2, \mathbf{A}_1)\right] + \left[\hat{\theta}_g(\mathbf{w}_1, \mathbf{A}_1) - \hat{\theta}_g(\mathbf{w}_1, \mathbf{A}_2)\right]. \quad (13)$$

This function quantifies the cost savings that the learner could achieve by unilaterally changing the model from $\mathbf{w}_1$ to $\mathbf{w}_2$ plus the cost savings that the data generator could achieve by unilaterally changing the transformation from $\mathbf{A}_1$ to $\mathbf{A}_2$. Nikaido-Isoda function $\hat{\theta}$ is concave in $(\mathbf{w}_2, \mathbf{A}_2)$ because $\hat{\theta}_f$ and $\hat{\theta}_g$ are convex, and the cost functions for $(\mathbf{w}_2, \mathbf{A}_2)$ enter the function as negatives.

For convex-concave Nikaido-Isoda functions, parameters $[\mathbf{w}^*, \mathbf{A}^*]$ are an equilibrium point if and only if the Nikaido-Isoda function has a saddle point at $([\mathbf{w}^*, \mathbf{A}^*], [\mathbf{w}^*, \mathbf{A}^*])$ [9]. The intuition behind this result is the following. An equilibrium point $(\mathbf{w}^*, \mathbf{A}^*)$ satisfies Equations 5 and 6 by definition. By Equation 13, $\hat{\theta}([\mathbf{w}^*, \mathbf{A}^*], [\mathbf{w}^*, \mathbf{A}^*]) = 0$. Equations 5 and 6 imply that $\hat{\theta}([\mathbf{w}, \mathbf{A}], [\mathbf{w}^*, \mathbf{A}^*])$ is positive and $\hat{\theta}([\mathbf{w}^*, \mathbf{A}^*], [\mathbf{w}, \mathbf{A}])$ is negative for $[\mathbf{w}, \mathbf{A}] \neq [\mathbf{w}^*, \mathbf{A}^*]$. When $\hat{\theta}$ is convex in $[\mathbf{w}_1, \mathbf{A}_1]$ and concave in $[\mathbf{w}_2, \mathbf{A}_2]$, this means that $(\mathbf{w}^*, \mathbf{A}^*)$ is an equilibrium point if and only if $\hat{\theta}$ has a saddle point at position $[\mathbf{w}^*, \mathbf{A}^*], [\mathbf{w}^*, \mathbf{A}^*]$.

Saddle points of convex-concave functions can be located with the Arrow-Hurwicz-Uzawa method [1]. We implement the method as an iterative procedure with a constant stepsize t [20]. In each iteration j, the method computes the gradient of $\hat{\theta}$ with respect to $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{A}_1$ and $\mathbf{A}_2$, and performs a descent by updating previous estimates:

$$(\mathbf{w}_1, \mathbf{A}_1)^{(j+1)} = (\mathbf{w}_1, \mathbf{A}_1)^{(j)} - t\nabla_{(\mathbf{w}_1, \mathbf{A}_1)}\hat{\theta}([\mathbf{w}_1, \mathbf{A}_1]^{(j)}, [\mathbf{w}_2, \mathbf{A}_2]^{(j)})$$
$$(\mathbf{w}_2, \mathbf{A}_2)^{(j+1)} = (\mathbf{w}_2, \mathbf{A}_2)^{(j)} + t\nabla_{(\mathbf{w}_2, \mathbf{A}_2)}\hat{\theta}([\mathbf{w}_1, \mathbf{A}_1]^{(j)}, [\mathbf{w}_2, \mathbf{A}_2]^{(j)}).$$

The final estimator of the equilibrium point after T iterations is the average of all iterates: $(\hat{\mathbf{w}}^*, \hat{\mathbf{A}}^*) = \sum_{j=1}^{T}(\mathbf{w}_1, \mathbf{A}_1)^{(j)}$. For any convex-concave $\hat{\theta}$, this method converges towards a saddle-point.

4.3 Parallelized Methods

Both the inexact line search method sketched in Section 4.1 and the Arrow-Hurwicz-Uzawa method derived in Section 4.2 can be implemented in a batch-parallel manner. To this end, the data is randomly partitioned into k batches $(\mathbf{X}_i, \mathbf{y}_i)$, where $i = 1, \ldots, k$. In practice, rather than splitting the data into k disjoint partitions, it is advisable to split the data into a larger number of portions but have some overlap between the portions. In the *map* step, k parallel nodes perform gradient descent on their respective batch of training examples; in the final *reduce step*, the k parameter vectors are averaged [26]. The execution time of averaging k parameter vectors $\mathbf{w}^i \in \mathbb{R}^m$ is vanishingly small in comparison to the execution time of the parallel gradient descent.

When $\mathbf{w}_1, \ldots, \mathbf{w}_k$ are equilibrium points of the games given by the respective partitions of the sample, then the averaged vector $\mathbf{w} = \frac{1}{k}\sum_{j=1}^{k}\mathbf{w}_j$ still cannot be guaranteed to be an equilibrium point of the game given by the entire sample. In fact, in the experimental study we will find example cases where this is not the case.

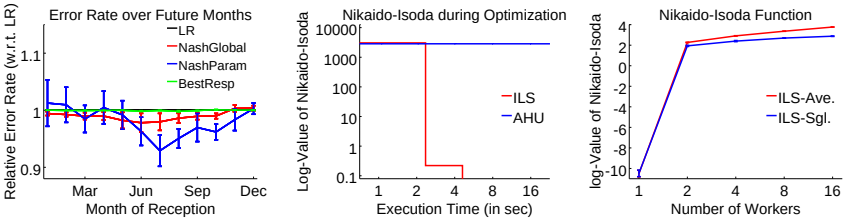

Fig. 1. Relative error (with respect to logistic regression, *LR*) of classification models evaluated into future (left). Value of Nikaido-Isoda function over time for three different optimization algorithms (center) and parallelized models (right). Error bars show standard errors.

5 Experimental Results

The goal of this section is to explore the robustness and scalability of sequential and parallelized methods that locate equilibrium points. We use a data set of 290,262 emails collected by an email service provider [3]. Each instance contains the term frequency of 226,342 terms. We compute a PCA of the emails and use the first 250 principal components as feature representation for most experiments. The data set is sorted chronologically. Emails that have been received over the final 12 months of the data collection period are held out for evaluation. Emails received in the month before that are used for tuning of the regularization parameters. Training emails are drawn from the remaining set of older emails.

5.1 Reference Methods

We use the logistic loss for all methods and for both learner and data generator. This makes logistic regression (*LR*) our natural *iid* baseline learning method. In the first experiment, we compare the transformation model derived in Section 2 (*NashParam*) that uses a parameterized function of individual instances to the global transformation model [3] that allows arbitrary dependencies between perturbations of multiple instances (*NashGlobal*). Additionally, we use the following simple game-theoretic reference method (*BestResp*): The data generator chooses the perturbation that is the best response to the standard logistic regression, and the learner chooses the model parameters that are the best response to this perturbation. That is, *BestResp* chooses parameters $\mathbf{w}^*$ according to:

1. $\mathbf{w}' = \arg\min_{\mathbf{w}} \hat{\theta}_f(\mathbf{w}, \mathbf{0}_{m\times m})$
2. $\mathbf{A}' = \arg\min_{\mathbf{A}} \hat{\theta}_g(\mathbf{w}', \mathbf{A})$
3. $\mathbf{w}^* = \arg\min_{\mathbf{w}} \hat{\theta}_f(\mathbf{w}, \mathbf{A})$.

5.2 Performance of the Parameterized Transformation Model

We compare a standard logistic regression approach (*LR*), game-theoretic heuristic *BestResp*, an equilibrium point with global transformation model (*Nash-*

Global), and the equilibrium point with the parameterized transformation model (*NashParam*). In each iteration, we sample 2500 training instances from the training portion of the data. We tune the free parameters—the regularization parameters of the learner and the data-generator—on the younger tuning portion of the data. All models are then evaluated over the final 12 evaluation months. We repeat this procedure 10 times and average the resulting error rates. For all following experiments, we keep all regularization parameters fixed, using the values that the parameters have been tuned to here.

Figure 1 (left) show the average relative error of all models with respect to logistic regression (*LR*); error bars show the standard error. Both *NashGlobal* and *NashParam* achieve significant improvements over *LR*. The parameterized transformation model *NashParam* reduces the error rate over *NashGlobal* by up to eight percent. The heuristic *BestResp* does not perform better than *LR*.

5.3 Optimization Algorithms

This section compares the convergence rates of the inexact line search (*ILS*) and Arrow-Hurwicz-Uzawa (*AHU*) approaches to finding equilibrium points, discussed in Sections 4.1 and 4.2, respectively.

In each repetition of the experiment, we sample 10,000 instances from the training portion of the data. Here, we use the 500 first principal components as feature representation. In each iteration of the optimization procedures, we measure he Nikaido-Isoda function of the current pair of parameters and the best possible reactions to these parameters—this function reaches zero at an equilibrium point. Figure 1 (center) shows that the *ILS* procedure converges very quickly. By contrast, *AHU* requires several orders of magnitude more time before the Nikaido-Isoda function drops noticably (not visible in the diagram); we have not been able to observe convergence. Increasing the regularization parameters by a factor of 100—which should make the optimization criterion more convex—did not change these findings. We therefore excluded *AHU* from further investigation.

5.4 Parallelized Models

In this section, we study parallel batch gradient descent, as discussed in Section 4.3, based on *ILS*. In each repetition, we sample 3200 instances from the training portion; we average 10 trials. The baseline model *LR-1-Sgl* is trained on all training data. Each of 8 nodes then processes a batch of data and returns a model *LR-8-Sgl*; these parameter vectors are averaged into *LR-8-Avg*. Likewise, *ILS-1-Sgl* is trained on all training data. Each node returns a model *ILS-8-Sgl*; these models are averaged into *ILS-8-Avg*.

For logistic regression, Figures 2 (left diagram, each node processes $\frac{1}{8}$ of the data), 3 (left, each node processes $\frac{1}{4}$ of the data), and 4 (left, each node processes $\frac{1}{\sqrt{8}}$ of the data), the averaged models *LR-8-Avg* consistently outperform the individual models *LR-8-Sgl*. Model *LR-1-Sgl* that has been trained sequentially

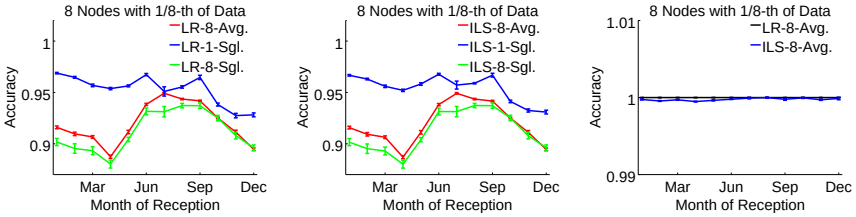

Fig. 2. Accuracy of logistic regression (left) and equilibrium points (center). Error rate of the aggregated equilibrium point relative to the error rate of aggregated logistic regression models (right). Each of 8 nodes processes $\frac{1}{8}$-th of the data. Error bars show standard errors.

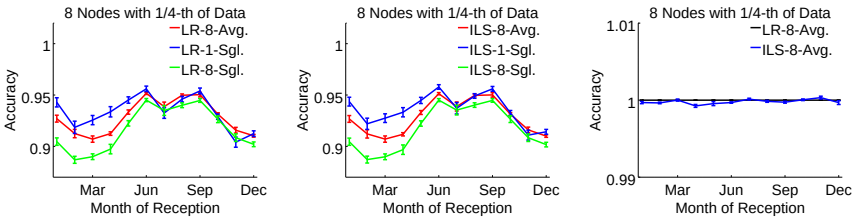

Fig. 3. Accuracy of logistic regression (left) and equilibrium points (center). Error rate of the aggregated equilibrium point relative to the error rate of aggregated logistic regression models (right). Each of 8 nodes processes $\frac{1}{4}$-th of the data. Error bars show standard errors.

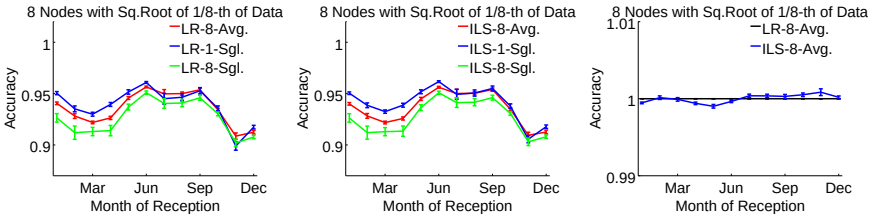

Fig. 4. Accuracy of logistic regression (left) and equilibrium points (center). Error rate of the aggregated equilibrium point relative to the error rate of aggregated logistic regression models (right). Each of 8 nodes processes $\frac{1}{\sqrt{8}}$-th of the data. Error bars show standard errors.

on all available data outperforms the averaged models—this is consistent with earlier results on parallel stochastic gradient descent [18,26]. The same is true for the equilibrium models found by *ILS*: Figures 2 (center, $\frac{1}{8}$ of the data per node), 3 (center, $\frac{1}{8}$ of the data per node), and 4 (center, $\frac{1}{\sqrt{8}}$ of the data per node) show that the averaged models *ILS-8-Avg* outperform the parallel models *ILS-8-Sgl*. The sequentially trained model *ILS-1-Sgl* outperforms the averaged models.

Figures 2 (right, $\frac{1}{4}$ of the data), and 3 (right, $\frac{1}{8}$ of the data), and 4 (right, $\frac{1}{\sqrt{8}}$ of the data) show the error rate of *ILS-8-Avg* relative to the error rate of *LR-8-Avg*, in analogy to Figure 1 (left). While in Figure Figure 1 (left) the equilibrium points have outperformed *LR*, the averaged model *ILS-8-Avg* tends to have a similar error rate as *LR-8-Avg*. The averaged equilibrium parameters—while still outperforming the equilibrium parameters trained on parallel batches—are no longer more accurate than the averaged logistic regression models.

We investigate further why this is the case. Figure 1 (right) shows the value of the Nikaido-Isoda function at the end of the batch optimization process for a single model trained on $\frac{1}{k}$ of the data (*ILS-Sgl*), and the corresponding Nikaido-Isoda function value for the average of k models trained on $\frac{1}{k}$ of the data each (*ILS-Avg*). Surprisingly, the averaged parameter vectors have a higher function value which means that they are further away from being equilibrium points than the individual models.

We can conclude that for this application (a) equilibrium points tend to be more accurate than standard logistic regression models; (b) averaging parameter vectors that have been trained on different batches of the data always leads to more accurate models; but (c) averaging equilibrium points tends to lead to model parameters that are no longer equilibrium points, and are therefore not generally more accurate than standard logistic regression models.

6 Conclusion

We have derived a model of adversarial learning in which the data generator gets to choose a parametric perturbation function $g_{\mathbf{A}}(\mathbf{x}) = \mathbf{x} + \mathbf{A}\mathbf{x}$ which is used to transform observations at application time. We have shown that the game between learner and data generator has at least one equilibrium point for convex and continuous loss functions. We have shown that the equilibrium point is unique if the loss function of learner and data generator have identical curvatures (as can be achieved with logistic loss functions) and the relationship between the regularization coefficients of learner and data generator are balanced as required by Assumption 2. Empirically, we observe that for the application of email spam filtering, equilibrium points under the derived data generation model maintain a higher accuracy over an evaluation period of 12 months after training than *iid* learning and reference methods.

The MapReduce programming model offers an unrivaled speed-up potential because it requires all synchronization to be limited to a final aggregation step. We derived batch-parallel stochastic gradient methods that identify a unique

equilibrium point and can be implemented using the MapReduce model. Prior work on parallel stochastic gradient descent has established that the aggregate of models that have been trained in parallel on subsets of the data are more accurate than the individual, aggregated models, and that the aggregate converges toward to performance of a single model that has been trained sequentially on all the data [18,26]. We observe that this is also true for the aggregates of equilibrium points that have been located in parallel on batches of the data. However, it turns out that aggregates of equilibrium points are not equilibrium points themselves; the Nikaido-Isoda function increases its value during the aggregation step. Therefore, aggregated logistic regression models are about as accurate as aggregated equilibrium points. From a practical point of view, this implies that searching for equilibrium points is advisable for adversarial applications as long as training data, and not computation time, is the limiting factor. As the sample size increases, the computation time needed to locate equilibrium points on a single node becomes the limiting factor. For intermediate sample sizes, it may still be possible (and advisable) to train a model on a single node using *iid* learning. For even larger sample sizes, this becomes impossible. At this point, aggregated batch-parallel gradient descent outperforms sequential optimization using a subset of the data. At this point, however, aggregated equilibrium points offer no advantage over aggregated models trained under the *iid* assumption.

Acknowledgment

This work was supported by the German Science Foundation DFG under grant SCHE540/12-2.

References

1. Arrow, K., Hurwicz, L., Uzawa, H.: Studies in Linear and Non-Linear Programming. Stanford University Press, Stanford (1958)
2. Basar, T., Olsder, G.J.: Dynamic Noncooperative Game Theory. Academic Press, Londen New York (1995)
3. Brückner, M., Kanzow, C., Scheffer, T.: Static prediction games for adversarial learning problems. Journal of Machine Learning Research 12, 2617–2654 (2012)
4. Brückner, M., Scheffer, T.: Stackelberg games for adversarial learning problems. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2011)
5. Bu, Y., Howe, B., Balazinska, M., Ernst, M.: Haloop: Efficient iterative data processing on large clusters. In: Proceedings of the VLDB Endowment. vol. 3 (2010)
6. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of the 6th Symposium on Operating System Design and Implementation (2004)
7. Dekel, O., Shamir, O.: Learning to classify with missing and corrupted features. In: Proceedings of the International Conference on Machine Learning. ACM Press (2008)
8. Dekel, O., Shamir, O., Xiao, L.: Learning to classify with missing and corrupted features. Machine Learning 81(2), 149–178 (2010)

9. Flam, S., Ruszczynski, A.: Computing normalized equilibria in convex-concave games. Working Papers Working Papers 2006:9, Lund University, Department of Economics (2006)
10. Geiger, C., Kanzow, C.: Theorie und Numerik restringierter Optimierungsaufgaben. Springer-Verlag, Berlin Heidelberg New York (2002)
11. Ghaoui, L.E., Lanckriet, G.R.G., Natsoulis, G.: Robust classification with interval data. Tech. Rep. UCB/CSD-03-1279, EECS Department, University of California, Berkeley (2003)
12. Globerson, A., Roweis, S.T.: Nightmare at test time: Robust learning by feature deletion. In: Proceedings of the International Conference on Machine Learning. ACM Press (2006)
13. Globerson, A., Teo, C.H., Smola, A.J., Roweis, S.T.: Dataset Shift in Machine Learning, chap. An adversarial view of covariate shift and a minimax approach, pp. 179–198. MIT Press (2009)
14. Großhans, M., Sawade, C., Brückner, M., Scheffer, T.: Bayesian games for adversarial regression problems. In: Proceedings of the International Conference on Machine Learning (2013)
15. Hardt, M., Megiddo, N., Papadimitriou, C., Wooters, M.: Strategic classification. Unpublished manuscript
16. von Heusinger, A., Kanzow, C.: Relaxation methods for generalized nash equilibrium problems with inexact line search. Journal of Optimization Theory and Applications 143(1), 159–183 (2009)
17. Lanckriet, G.R.G., Ghaoui, L.E., Bhattacharyya, C., Jordan, M.I.: A robust minimax approach to classification. Journal of Machine Learning Research 3, 555–582 (2002)
18. Mann, G., McDonald, R., Mohri, M., Silberman, N., Walker, D.: Efficient large-scale distributed training of conditional maximum entropy models. Advances in Neural Information Processing 22 (2009)
19. Nedic, A., Bertsekas, D., Borkar, V.: Distributed asynchronous incremental subgradient methods. Studies in Computational Mathematics 8, 381–407 (2001)
20. Nedic, A., Ozdaglar, A.: Subgradient methods for saddle-point problems. Journal of Optimization Theory and Applications 142(1), 205–228 (2009)
21. Nikaido, H., Isoda, K.: Note on noncooperative convex games. Pacific Journal of Mathematics 5, 807–815 (1955)
22. Rosen, J.B.: Existence and uniqueness of equilibrium points for concave n-person games. Econometrica 33(3), 520–534 (1965)
23. Teo, C.H., Globerson, A., Roweis, S.T., Smola, A.J.: Convex learning with invariances. In: Advances in Neural Information Processing Systems. MIT Press (2007)
24. Torkamani, M., Lowd, D.: Convex adversarial collective classification. In: Proceedings of the International Conference on Machine Learning (2013)
25. Weimer, M., Condie, T., Ramakrishnan, R.: Machine learning in scalops, a higher order cloud computing language. In: NIPS 2011 Workshop on parallel and large-scale machine learning (BigLearn) (2011)
26. Zinkevich, M., Weimer, M., Smola, A., Li, L.: Parallelized stochastic gradient descent. Advances in Neural Information Processing 23 (2010)