An online passive-aggressive algorithm for difference-of-squares classification

Anonymous Author(s)
Affiliation
Address
email

Abstract

We investigate a low-rank model of quadratic classification inspired by recent capsule-based architectures. The model is parameterized by a pair of affine transformations, and it classifies examples by comparing the magnitudes of vectors that these transformations produce. The model is also *over-parameterized* in the sense that different pairs of affine transformations can describe classifiers with the same decision boundary and confidence scores. We show that such pairs arise from discrete and continuous symmetries of the model's parameter space: in particular, the latter define Lie groups of rotations and Lorentz transformations, and we use these group structures to devise appropriately invariant procedures for model alignment and averaging. We also leverage the form of the model's decision boundary to derive simple margin-based updates for online learning. Here we explore a strategy of *passive-aggressive learning*: for each example, we compute the minimum change in parameters that is required to predict its correct label with high confidence. We derive these updates by solving a quadratically constrained quadratic program (QCQP); interestingly, this QCQP is *nonconvex but tractable*, and it can be solved efficiently by elementary methods. We highlight the conceptual and practical contributions of this approach. Conceptually, we show that it extends the paradigm of passive-aggressive learning to a larger family of nonlinear models for classification. Practically, we show that these models perform well on large-scale problems in online learning.

1 Introduction

As data sets grow in size and complexity, they create new opportunities—and challenges—for large-scale applications of online learning [1]. We begin with a hopeful observation: these challenges have been extensively studied and, for the most part, elegantly resolved for the simplest linear models of classification [2]. For such models, one particularly elegant approach is that of *passive-aggressive learning* [3]. In this framework, a model is only updated when it fails to classify an example correctly with high confidence. When an update is triggered, however, the model is changed by whatever minimum amount is required to achieve this goal. This approach neatly dispenses with the need to choose or adapt learning rates.

Given the appeal of such updates, it is of natural interest to extend this approach to a larger family of nonlinear models. Proceeding from the linear model, this can be done most straightforwardly by the use of kernel methods [4–6]. But kernel methods do not scale effortlessly to the regime of online learning that we envision in this paper—where the examples are arriving in a streaming fashion from an essentially unlimited source [7, 8]. It is possible to adopt a budgeted approach [9–14] for kernel-based passive-aggressive updates [15–17], conserving memory at the expense of model capacity, but this approach necessarily entails further complexities.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Without the kernel trick, we must face the crux of the problem. Passive-aggressive updates hinge on the ability to perform a fundamental calculation: when an update is triggered, we must compute the minimum change in model parameters that is required to fix the classifier's decision boundary. This is a relatively simple calculation for linear models (with hyperplane decision boundaries), but an enormously complex one for (say) decision trees, ensemble methods, and neural nets with threshold or ReLU activation functions. The question is whether this is true for all nonlinear models.

This paper answers that question. We investigate a family of low-rank models for quadratic classification inspired by so-called *capsule* networks for visual object recognition [18, 19]. For models in this family, we show how to derive passive-aggressive updates by solving a quadratically constrained quadratic program (QCQP). The QCQP is nonconvex but tractable: it reduces to an elementary minimization over a bounded interval. In this way, we are able to extend the framework of passive-aggressive learning to a larger family of nonlinear models. This is the main goal of the paper.

These models should also be of broad interest for other reasons. For practitioners, they provide a natural bridge between purely linear and fully quadratic models of classification. For theoreticians, they suggest a proving ground for larger questions of convergence and generalization in over-parameterized models [20–22]. We return to these ideas at the end of the paper.

The organization of this paper is as follows. In section 2, we review the main lines of work that motivated this study. In section 3, we formulate our model, elucidate its symmetries, and derive the updates for passive-aggressive learning. In section 4, we provide experimental results on a data set with 100M training examples. Finally, in section 5, we sketch several directions for future work.

2 Background and related work

Passive-aggressive updates [3] emerged from a long line of work on incremental learning of linear classifiers [23–32]. These updates are based on a simple intuition. We suppose that an algorithm has access to a stream of labeled examples $\{(\mathbf{x}_t, y_t)\}_{t\geq 1}$, where $\mathbf{x}_t \in \mathbb{R}^n$ and $y_t \in \{-1, +1\}$. The goal of the algorithm is to learn a linear classifier parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^n$. At the outset, the weight vector is initialized to the origin. Then, after the algorithm sees each example, the weight vector is updated by solving the following constrained optimization:

$$\mathbf{w}_{t+1} \;=\; \underset{\mathbf{w}\in\mathbb{R}^n}{\operatorname{argmin}}\; \|\mathbf{w} - \mathbf{w}_t\|^2 \quad \text{such that} \quad y_t \mathbf{w}^\top \mathbf{x}_t \;\geq\; 1. \tag{1}$$

The optimization is trivial if the algorithm already classifies $\mathbf{x}_t$ correctly with at least unit margin: in this case, the algorithm responds *passively*, making no change at all to the weight vector. When this is not the case, however, the algorithm responds *aggressively*, changing $\mathbf{w}$ by the minimum amount required to achieve this goal. The optimization in eq. (1) has the closed-form solution:

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t \;+\; \alpha_t y_t \mathbf{x}_t \quad \text{where} \quad \alpha_t \;=\; \max(0, 1 - y_t \mathbf{w}^\top \mathbf{x}_t)/\|\mathbf{x}_t\|^2. \tag{2}$$

This update is appealing in its simplicity and easily adapted to handle noisy labels or to learn a separating hyperplane that does not pass through the origin [3]. Similar updates have been studied for problems in online regression [3], portfolio selection [33], nonnegative matrix factorization [34], and active learning [35].

The above framework can also be generalized via the "kernel trick" [4–6] to learn nonlinear classifiers. This is done simply by replacing the dot products in eqs (1–2), such as $\mathbf{w}^\top \mathbf{x}$, by kernel function evaluations, such as $k(\mathbf{w}, \mathbf{x})$. In this case, however, the kernelized algorithm will typically limit the number of examples that are used to construct the model's decision boundary [9–17].

In this paper we also develop passive-aggressive updates for a nonlinear model of classification. Our model, however, is not rooted in the use of kernel methods; instead it is inspired by so-called capsule networks for visual object recognition [18, 19]. Like traditional neural nets, capsule-based architectures compute vectors of hidden unit activities; unlike traditional neural nets, they use the *magnitudes* of these vectors to encode the probability that an entity is present in an image. This idea has mainly been developed for deep architectures with multiple layers of interlinked capsules [19, 36–48], although recently it was explored for a single-capsule latent variable model of multiway classification [49]. Here we pursue this simplification even further for problems of binary classification. Whereas the latent variable model in [49] is well suited for maximum likelihood estimation in a batch setting, the model in this paper is much better suited to passive-aggressive learning in an online setting. We develop these ideas further in the next section.

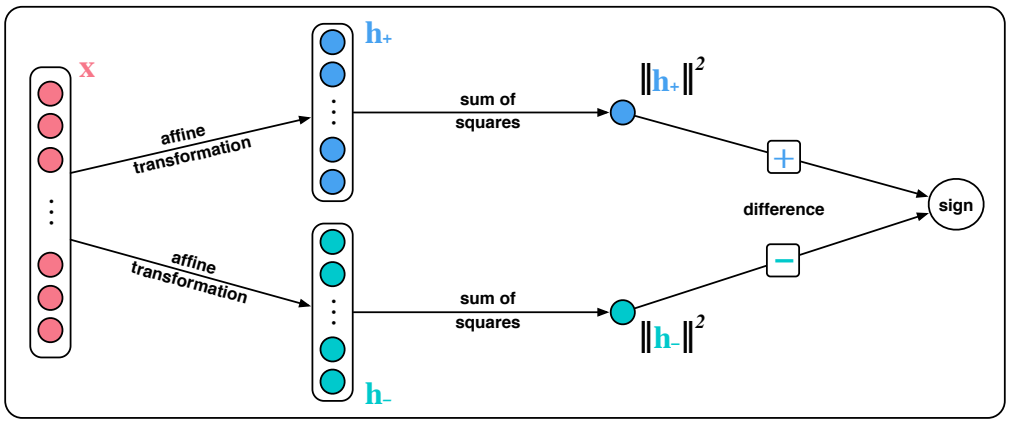

Figure 1: Difference-of-squares (DoS) classifier visualized as a neural network with one hidden layer. The network uses a pair of affine transformations to map each input $\mathbf{x}$ into vectors $\mathbf{h}_+$ and $\mathbf{h}_-$ of hidden unit activities. Then it labels the input by comparing the squared magnitudes of these vectors.

3 Model

In this section we introduce the model at the heart of this paper. We begin by formulating the model as a neural network (3.1) and elucidating the symmetries of its parameter space (3.2). Next we derive the passive-aggressive updates for online learning (3.3) and show how to average different models across time (3.4).

3.1 Difference-of-squares classification

The model we study is most easily visualized as the neural network in Fig. 1. The network maps the input into a hidden layer by a pair of affine transformations: $\mathbf{x} \mapsto \mathbf{A}_\pm \mathbf{x} + \mathbf{b}_\pm$. Then it predicts a binary label $y \in \{-1, +1\}$ by comparing the magnitudes of the vectors produced in this way. More concretely, let $\mathbf{h}_\pm \in \mathbb{R}^d$ denote the vectors of hidden unit activities. The label is predicted as follows:

$$\mathbf{h}_+ = \mathbf{A}_+\mathbf{x} + \mathbf{b}_+, \tag{3}$$

$$\mathbf{h}_- = \mathbf{A}_-\mathbf{x} + \mathbf{b}_-, \tag{4}$$

$$y = \text{sign}\left(\|\mathbf{h}_+\|^2 - \|\mathbf{h}_-\|^2\right). \tag{5}$$

Based on the form of eq. (5), we refer to this model as a difference-of-squares (DoS) classifier. The dimensionalities of $\mathbf{h}_+$ and $\mathbf{h}_-$ are hyperparameters of the model, which for convenience we set to be equal. Networks of this form are not universal function approximators because their decision boundary is necessarily quadratic. This is in contrast to neural networks with non-polynomial activation functions, which can approximate any decision boundary in the limit of infinitely many hidden units [50–53].

The DoS model may not be a universal approximator, but with sufficiently many hidden units, it can mimic any fully quadratic classifier. For example, consider a decision rule of the form:

$$y = \text{sign}\left(\mathbf{x}^\top \mathbf{P}\mathbf{x} + \mathbf{q}^\top \mathbf{x} + r\right). \tag{6}$$

To prove the claim, we simply express $\mathbf{P}$ as the difference of positive semidefinite matrices $\mathbf{P}_+$ and $\mathbf{P}_-$. In particular, if $\mathbf{P} = \mathbf{P}_+ - \mathbf{P}_-$, then the decision rule in eq. (6) can be recast in the form of eqs. (3–5) by setting

$$\mathbf{A}_\pm = \begin{bmatrix} \mathbf{q}^\top \\ \mathbf{P}_\pm^{\frac{1}{2}} \end{bmatrix}, \quad \mathbf{b}_\pm = \begin{bmatrix} r \pm \frac{1}{4} \\ \mathbf{0} \end{bmatrix}, \tag{7}$$

where the exponent of $\mathbf{P}_\pm$ on the left side denotes a matrix square root. From the mapping in eq. (7), we also see the potential of this parameterization. Suppose that the input $\mathbf{x}$ is high dimensional but that the matrix $\mathbf{P}$ is of very low rank. Then the matrices $\mathbf{P}_\pm$ can be short and wide as opposed to square, and the matrices $\mathbf{A}_\pm$ in eq. (7) will have many fewer rows than columns. Such a DoS model (with so many fewer parameters) may be easier to estimate than a fully quadratic classifier.

Before proceeding, we adopt a more unified notation for the model parameters $\mathbf{A}_\pm$ and $\mathbf{b}_\pm$. We do so by working in an augmented input space; specifically we define

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \quad \mathbf{U} = [\mathbf{A}_+ \ \mathbf{b}_+], \quad \mathbf{V} = [\mathbf{A}_- \ \mathbf{b}_-], \tag{8}$$

so that the vector $\mathbf{z}$ has exactly one more row than the input $\mathbf{x}$, and the matrices $\mathbf{U}$ and $\mathbf{V}$ have exactly one more column than the matrices $\mathbf{A}_+$ and $\mathbf{A}_-$. In this notation, the model's decision boundary simplifies to

$$y \ = \ \operatorname{sign}\left(\|\mathbf{U}\mathbf{z}\|^2 - \|\mathbf{V}\mathbf{z}\|^2\right) \ = \ \operatorname{sign}\Big(\mathbf{z}^\top\Big[\mathbf{U}^\top\mathbf{U} - \mathbf{V}^\top\mathbf{V}\Big]\mathbf{z}\Big). \tag{9}$$

The first expression in eq. (9) provides the most efficient way to compute the label $y \in \{-1, +1\}$, but the second makes plain the model's symmetries. We explore these symmetries in the next section.

3.2 Symmetries of the model

Eq. (9) shows that the model's decision boundary and confidence scores only depend on its parameters through the difference $\mathbf{U}^\top\mathbf{U} - \mathbf{V}^\top\mathbf{V}$. From this observation we deduce two important symmetries. First, the model's predictions are invariant to *orthogonal* transformations of the form

$$\mathbf{U} \ \mapsto \ \boldsymbol{\Omega}\,\mathbf{U}, \tag{10}$$

$$\mathbf{V} \ \mapsto \ \boldsymbol{\Lambda}\,\mathbf{V}, \tag{11}$$

where $\boldsymbol{\Omega}$ and $\boldsymbol{\Lambda}$ are (independently chosen) $d \times d$ orthogonal matrices, satisfying $\boldsymbol{\Omega}^\top\boldsymbol{\Omega} = \boldsymbol{\Lambda}^\top\boldsymbol{\Lambda} = \mathbf{I}$. Second, the model's predictions are invariant to *Lorentz* transformations (or " boosts") of the form

$$\mathbf{U} \ \mapsto \ \mathbf{U}\cosh\varphi - \mathbf{V}\sinh\varphi, \tag{12}$$

$$\mathbf{V} \ \mapsto \ \mathbf{V}\cosh\varphi - \mathbf{U}\sinh\varphi, \tag{13}$$

where $\varphi \in \mathbb{R}$. (Lorentz transformations are a symmetry of spacetime in the theory of special relativity [54].) Both symmetries reveal that the model is *over-parameterized*; by this we mean that models with different parameters may describe classifiers with the same decision boundary and confidence scores. It is common to observe this property in neural networks with hidden units [20], but in general the modes of overparameterization may not be so easily (or elegantly) characterized by well-known Lie groups of continuous symmetries. We return to this idea later.

The orthogonal and Lorentz transformations in eqs. (10–13) reflect fundamentally different symmetries of the `DoS` parameter space. Most importantly, note that the former do not change the Frobenius norms of the matrices $\mathbf{U}$ and $\mathbf{V}$, while the latter do. From eqs. (12–13) it is possible, in fact, to compute the Lorentz transformation that yields the model parameters of minimum norm:

$$\varphi^* \ = \ \operatorname*{argmin}_{\varphi} \left\{ \left\|\mathbf{U}\cosh\varphi - \mathbf{V}\sinh\varphi\right\|_F^2 + \left\|\mathbf{V}\cosh\varphi - \mathbf{U}\sinh\varphi\right\|_F^2 \right\} \tag{14}$$

As shorthand notation, we define the matrix inner product $\langle \mathbf{U}, \mathbf{V} \rangle = \sum_{ij} U_{ij} V_{ij}$. Then the solution to eq. (14) takes the simple form:

$$\varphi^* \ = \ \frac{1}{2}\tanh^{-1}\left(\frac{2\langle \mathbf{U}, \mathbf{V}\rangle}{\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2}\right). \tag{15}$$

We omit the derivation of this result, which is obtained by equating the derivative of the bracketed expression in eq. (14) to zero. The result will be needed in section 3.4, when we discuss how to average different models across time.

3.3 Passive-aggressive updates for online learning

We consider learning in the online setting where labeled examples arrive one at a time and are subsequently discarded. Let $(\mathbf{U}_t, \mathbf{V}_t)$ denote the model parameters before the example $(\mathbf{x}_t, y_t)$ arrives at time t. We explore a passive-aggressive approach, in which at each time step, the algorithm updates its model by solving the following optimization:

$$\min_{\mathbf{U},\mathbf{V}} \left\{ \|\mathbf{U} - \mathbf{U}_t\|_F^2 + \|\mathbf{V} - \mathbf{V}_t\|_F^2 \right\} \quad \text{such that} \quad y_t\Big(\|\mathbf{U}\mathbf{z}_t\|^2 - \|\mathbf{V}\mathbf{z}_t\|^2\Big) \geq 1. \tag{16}$$

Suppose that the example $(\mathbf{x}_t, y_t)$ is already classified correctly and with high confidence. Then the algorithm does not change the model: i.e., $\mathbf{U}_{t+1} = \mathbf{U}_t$ and $\mathbf{V}_{t+1} = \mathbf{V}_t$. On the other hand, when this is not the case, the algorithm performs the minimal update that is needed to correct this failure.

The optimization in eq. (16) is a quadratically constrained quadratic program (QCQP). This QCQP is not convex due to the difference of squared terms that appears in its constraint. It is, however, tractable [55–57]: nonconvex QCQPs with a single constraint can be efficiently solved by a so-called *S-procedure* from nonlinear control theory [58, 59]. This general procedure is beyond the scope of this paper, but for the problem of interest in eq. (16), it is still possible to obtain (if not fully justify) the solution by more elementary methods.

We start by presenting the solution as a *fait accompli*. The optimization over matrices $(\mathbf{U}, \mathbf{V})$ in eq. (16) reduces, in the end, to determining a step size α_t analagous to the one that appears in eq. (2). For DoS models, the step size is found by solving the following one-dimensional optimization:

$$\alpha_t \;=\; \operatorname*{argmin}_{\nu \in (0,1)} \left[\left(\frac{1}{1 - y_t \nu} \right) \left\| \mathbf{U}_t \mathbf{z}_t \right\|^2 \;+\; \left(\frac{1}{1 + y_t \nu} \right) \left\| \mathbf{V}_t \mathbf{z}_t \right\|^2 \;-\; \nu \right]. \tag{17}$$

This minimization is convex, and in practice it is easily solved by standard methods (e.g., golden section search). Finally, given the step size α_t, the updates[1] for $\mathbf{U}$ and $\mathbf{V}$ take the simple form:

$$\mathbf{U}_{t+1} \;=\; \mathbf{U}_t \;+\; \left(\frac{\alpha_t}{y_t - \alpha_t} \right) \cdot \frac{(\mathbf{U}_t \mathbf{z}_t)\mathbf{z}_t^\top}{\|\mathbf{z}_t\|^2}\;, \tag{18}$$

$$\mathbf{V}_{t+1} \;=\; \mathbf{V}_t \;-\; \left(\frac{\alpha_t}{y_t + \alpha_t} \right) \cdot \frac{(\mathbf{V}_t \mathbf{z}_t)\mathbf{z}_t^\top}{\|\mathbf{z}_t\|^2}. \tag{19}$$

We emphasize that these updates are very nearly as simple to implement as the one for linear classifiers in eq. (2). At the same time, however, they provide access to a much richer family of models.

Next we briefly justify the form of these updates. The calculation that follows is a special case of a more general treatment [60] for nonconvex but tractable QCQPs, to which the reader is referred for details; treatments of even greater generality are also available [59, 61, 62]. Note that if an update is required for example $(\mathbf{x}_t, y_t)$, it follows (by continuity) that the solution to eq. (16) will satisfy the margin constraint with equality. We therefore start by forming the Lagrangian

$$\mathcal{L}(\mathbf{U}, \mathbf{V}, \lambda) \;=\; \|\mathbf{U} - \mathbf{U}_t\|_F^2 + \|\mathbf{V} - \mathbf{V}_t\|_F^2 \;+\; \lambda \left(y_t - \|\mathbf{U}\mathbf{z}_t\|^2 + \|\mathbf{V}\mathbf{z}_t\|^2 \right). \tag{20}$$

The next step is to eliminate the matrices $\mathbf{U}$ and $\mathbf{V}$. By setting their partial derivatives to zero, we find that these matrices satisfy

$$\mathbf{U} \;= \mathbf{U}_t \Big(\mathbf{I} - \lambda \mathbf{z}_t \mathbf{z}_t^\top \Big)^{-1} \;=\; \mathbf{U}_t \left(\mathbf{I} + \frac{\lambda \mathbf{z}_t \mathbf{z}_t^\top}{1 - \lambda \|\mathbf{z}_t\|^2} \right), \tag{21}$$

$$\mathbf{V} \;= \mathbf{V}_t \Big(\mathbf{I} + \lambda \mathbf{z}_t \mathbf{z}_t^\top \Big)^{-1} \;=\; \mathbf{V}_t \left(\mathbf{I} - \frac{\lambda \mathbf{z}_t \mathbf{z}_t^\top}{1 + \lambda \|\mathbf{z}_t\|^2} \right). \tag{22}$$

To obtain the rightmost expressions in eq. (21–22), we have used the Sherman-Morrison formula to invert the matrices $(\mathbf{I} \pm \lambda \mathbf{z}_t \mathbf{z}_t)$ assuming that $|\lambda| \neq \|\mathbf{z}\|^{-2}$. (We will verify this assumption in the course of finding a solution.) Given these expressions for $\mathbf{U}$ and $\mathbf{V}$, it remains only to determine the Lagrange multiplier λ. To do so, we simply enforce the constraint that $y = \|\mathbf{U}\mathbf{z}_t\|^2 - \|\mathbf{V}\mathbf{z}_t\|^2$. Substituting eqs. (21–22) into this constraint, we obtain the nonlinear equation:

$$y_t \;=\; \left(\frac{1}{1 - \lambda \|\mathbf{z}_t\|^2} \right)^2 \|\mathbf{U}_t \mathbf{z}_t\|^2 \;-\; \left(\frac{1}{1 + \lambda \|\mathbf{z}_t\|^2} \right)^2 \|\mathbf{V}_t \mathbf{z}_t\|^2. \tag{23}$$

To simplify what follows, we make the elementary change of variables $\nu = y_t \lambda \|\mathbf{z}_t\|^2$. With this change, we can rewrite eq. (23) as

$$0 \;=\; \frac{d}{d\nu} \left[\left(\frac{1}{1 - y_t \nu} \right) \|\mathbf{U}_t \mathbf{z}_t\|^2 \;+\; \left(\frac{1}{1 + y_t \nu} \right) \|\mathbf{V}_t \mathbf{z}_t\|^2 \;-\; \nu \right]. \tag{24}$$

Finally we observe that the derivative in eq. (24) is negative at $\nu = 0$ and vanishes for some $\nu \in (0, 1)$ when $y_t \left(\|\mathbf{U}_t \mathbf{z}_t\|^2 - \|\mathbf{V}_t \mathbf{z}_t\|^2 \right) < 1$; this inequality is, of course, precisely what triggers the update. In this way, we recover the minimization prescribed by eq. (17). Likewise, we obtain the passive-aggressive updates in eqs. (18–19) by substituting this solution for the step size into eqs. (21–22).

[1] These updates assume that neither $\|\mathbf{U}_t \mathbf{z}\|$ nor $\|\mathbf{V}_t \mathbf{z}\|$ is equal to zero. The assumption holds in practice, as such precise cancellations do not occur when the model parameters are initialized with random values.

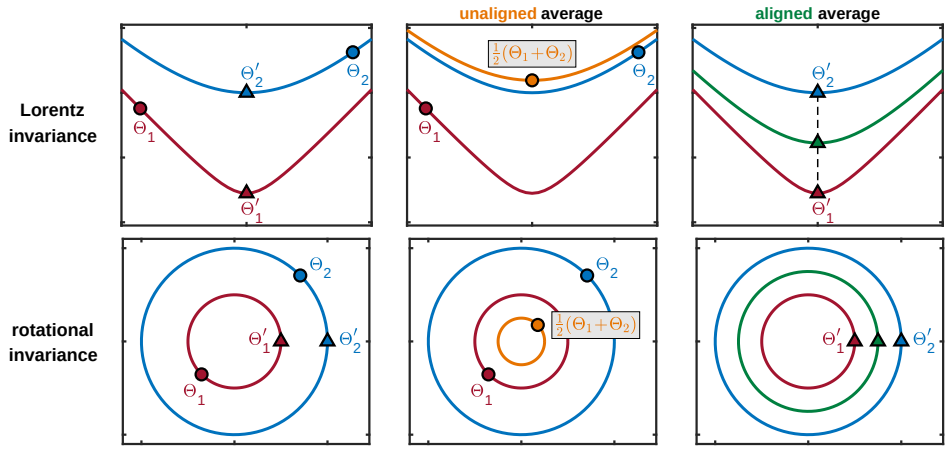

Figure 2: The DoS model is over-parameterized in the sense that different parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}'$ can generate classifiers with the same decision boundary and confidence scores. This is most evidently true when $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}'$ are related by a continuous symmetry of the model's parameter space, such as a Lorentz transformation (*top*) or a rotation (*bottom*). Let $\boldsymbol{\Theta}_1, \boldsymbol{\Theta}'_1$ be one such pair of parameters, and let $\boldsymbol{\Theta}_2, \boldsymbol{\Theta}'_2$ be another (*left*). Suppose we desire a model that represents, in some meaningful sense, the *average* of the models defined by $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$. Due to the model's symmetries, the midpoint $\frac{1}{2}(\boldsymbol{\Theta}_1+\boldsymbol{\Theta}_2)$ may yield a nonsensical result for this average (*middle*). However, the midpoint $\frac{1}{2}(\boldsymbol{\Theta}'_1+\boldsymbol{\Theta}'_2)$ will define a sensible average if $\boldsymbol{\Theta}'_1$ and $\boldsymbol{\Theta}'_2$ are appropriately aligned (*right*).

3.4 Model averaging

Passive-aggressive updates will not converge to a stable solution if there is no model that perfectly classifies the training examples, and they may converge very slowly even when there is. This may not matter if the model is to be indefinitely deployed in an online setting (e.g., when the examples are drawn from a nonstationary or adversarial distribution [63]). But it does matter if the ultimate goal is to train and deploy a single fixed model.

For linear classifiers, it is known that a stable result can be obtained by averaging models across time [27, 28, 64]. Let $\boldsymbol{\Theta}_m$ denote the model parameters obtained after m intervals of training (where each interval corresponds to some period of natural interest—a single update, an epoch, a second, a minute, etc). In the online setting, an averaged model $\hat{\boldsymbol{\Theta}}_m = \frac{1}{m}\sum_{k=1}^{m} \boldsymbol{\Theta}_k$ can be computed at the end of each interval. This type of averaging has a simple intuition for linear models where the parameters specify a separating hyperplane. For example, consider two models whose separating hyperplanes are parallel. Averaging their parameters, we obtain another model whose separating hyperplane is sandwiched in the middle. For linear classifiers, this is a sensible result.

The situation is more complicated for *over-parameterized* models, and a simple example shows why. Consider two DoS models with parameters $\boldsymbol{\Theta}_+ = (\mathbf{U}, \mathbf{V})$ and $\boldsymbol{\Theta}_- = (-\mathbf{U}, -\mathbf{V})$; also consider the model with the averaged parameter $\hat{\boldsymbol{\Theta}} = \frac{1}{2}(\boldsymbol{\Theta}_+ + \boldsymbol{\Theta}_-)$. Note that the models $\boldsymbol{\Theta}_+$ and $\boldsymbol{\Theta}_-$ specify exactly the *same* classifier, whereas the model $\hat{\boldsymbol{\Theta}} = (\mathbf{0}, \mathbf{0})$ does not specify a decision boundary at all. In this case, the naive average has clearly not yielded a sensible result. What has gone wrong?

The problem arises in large part due to the symmetries of the model's parameter space. The continuous symmetries, in particular, define low-dimensional manifolds of functionally equivalent classifiers, and these manifolds are not convex. This lack of convexity can lead to nonsensical results when intermediate models are constructed by linear interpolation. The problem is illustrated in Fig. 2 for the particular symmetries of DoS models. The figure also depicts a natural solution to this problem: it is to *align* the parameters of different models before performing their average. Intuitively, by first aligning these parameters, we compensate for the invariances of the model's parameter space.

More concretely, we can perform this aligned average in the following way. Suppose that we wish to compute a DoS model $\hat{\boldsymbol{\Theta}}$ that is intermediate, in a sensible way, between two other DoS models $\boldsymbol{\Theta}_1 = (\mathbf{U}_1, \mathbf{V}_1)$ and $\boldsymbol{\Theta}_2 = (\mathbf{U}_2, \mathbf{V}_2)$. We do this in three steps:

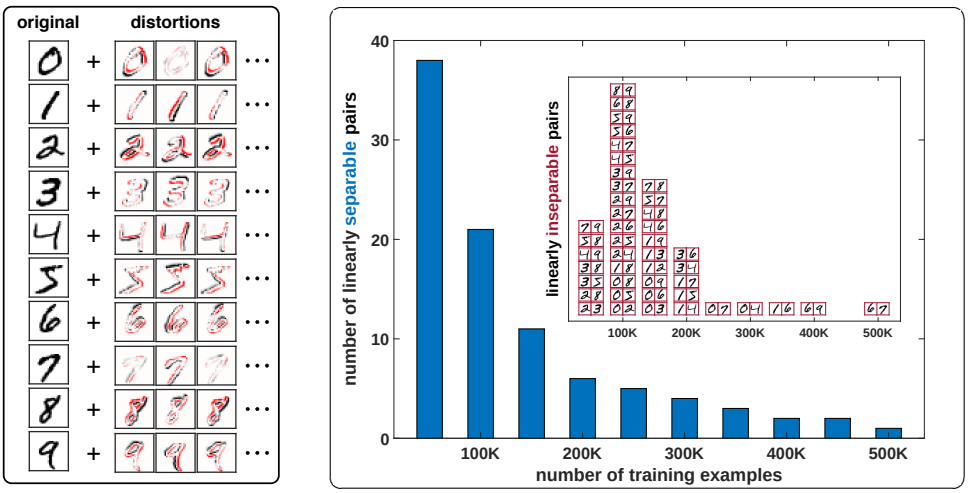

Figure 3: *Left*: The INFIMNIST data set [7] generates new digit images by distorting training examples from the original MNIST data set [67]. *Right:* Fewer pairs of digits remain linearly separable as more examples are added to the training set. The inset shows when (and which) pairs of digits become linearly inseparable as a function of the number of training examples.

1. Compute the boost φ_1 in eq. (14) that yields the minimum-norm model $\boldsymbol{\Theta}'_1 = (\mathbf{U}'_1, \mathbf{V}'_1)$ equivalent to $\boldsymbol{\Theta}_1 = (\mathbf{U}_1, \mathbf{V}_1)$. Also compute the boost φ_2 that yields the minimum-norm model $\boldsymbol{\Theta}'_2 = (\mathbf{U}'_2, \mathbf{V}'_2)$ equivalent to $\boldsymbol{\Theta}_2 = (\mathbf{U}_2, \mathbf{V}_2)$. This step accounts for the model's invariance under Lorentz transformations.
2. Compute the matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Lambda}$ that solve the orthogonal Procrustes problems [65]:

$$\boldsymbol{\Omega} = \operatorname*{argmin}_{\mathbf{Q}} \|\mathbf{Q}\mathbf{U}'_2 - \mathbf{U}'_1\|_F^2 \quad \text{such that} \quad \mathbf{Q}^\top \mathbf{Q} = \mathbf{I}, \tag{25}$$

$$\boldsymbol{\Lambda} = \operatorname*{argmin}_{\mathbf{Q}} \|\mathbf{Q}\mathbf{V}'_2 - \mathbf{V}'_1\|_F^2 \quad \text{such that} \quad \mathbf{Q}^\top \mathbf{Q} = \mathbf{I}. \tag{26}$$

   Intuitively, $\boldsymbol{\Omega}$ and $\boldsymbol{\Lambda}$ are the orthogonal transformations that best align $\mathbf{U}'_2$ with $\mathbf{U}'_1$ and $\mathbf{V}'_2$ with $\mathbf{V}'_1$. These matrices have closed-form solutions that are easily obtained from the singular value decompositions [66] of the matrices $\mathbf{U}'_1$, $\mathbf{U}'_2$, $\mathbf{V}'_1$, and $\mathbf{V}'_2$. Set $\boldsymbol{\Theta}''_2 = (\boldsymbol{\Omega}\mathbf{U}'_2, \boldsymbol{\Lambda}\mathbf{V}'_2)$, and note that by construction the models $\boldsymbol{\Theta}''_2$ and $\boldsymbol{\Theta}_2$ define the same classifier. This step accounts for the model's invariance under orthogonal transformations.
3. Compute $\hat{\boldsymbol{\Theta}}$ by averaging the newly aligned parameters $\boldsymbol{\Theta}'_1$ and $\boldsymbol{\Theta}''_2$, as opposed to the original (unaligned) parameters: i.e., for an unweighted average, set $\hat{\boldsymbol{\Theta}} = \frac{1}{2}(\boldsymbol{\Theta}'_1 + \boldsymbol{\Theta}''_2)$.

We can also extend this procedure to average multiple DoS models across time. To do so, we note that the *unaligned* average $\hat{\boldsymbol{\Theta}}_m = \frac{1}{m}\sum_{k=1}^{m} \boldsymbol{\Theta}_k$ may be computed incrementally by setting

$$\hat{\boldsymbol{\Theta}}_m = \left(1 - \tfrac{1}{m}\right) \hat{\boldsymbol{\Theta}}_{m-1} + \tfrac{1}{m}\boldsymbol{\Theta}_m. \tag{27}$$

For the *aligned* average, we simply replace the terms $\hat{\boldsymbol{\Theta}}_{m-1}$ and $\boldsymbol{\Theta}_m$ in eq. (27) by their equivalent but aligned counterparts $\hat{\boldsymbol{\Theta}}'_{m-1}$ and $\boldsymbol{\Theta}''_m$ from the above procedure.

4 Experimental results

We experimented on the INFIMNIST data set of handwritten digit images [7], a purposefully constructed superset of the original MNIST data set [67]. The first 10K images of this superset are those of the MNIST test set, and the next 60K images are those of the MNIST training set. The remaining images—about one trillion of them—are programmatically generated by distorting the 60K images of the MNIST training set; see Fig. 3 (*left*). Our experiments set aside the first 10K images as a test set and used the next 100M images (without reshuffling) as training examples for online learning.

The MNIST data has been extensively benchmarked. For the original 60K training examples, it is well known that most pairs of digit classes are linearly separable. This situation changes, however,

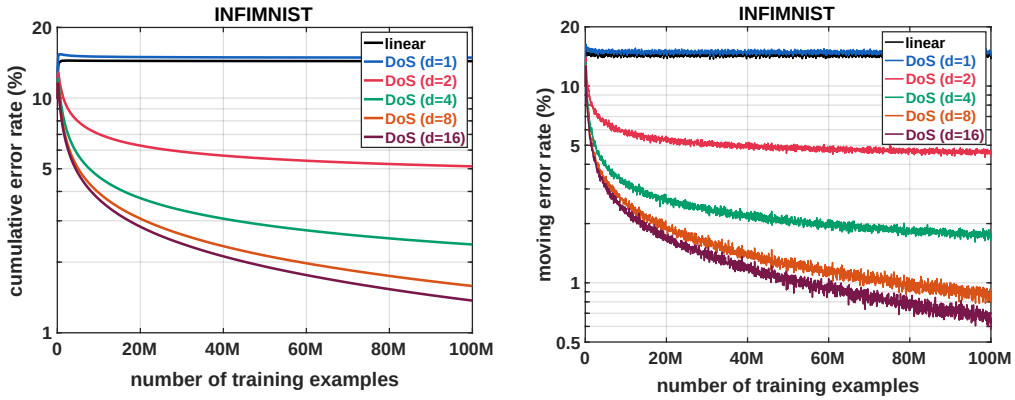

Figure 4: Cumulative and moving error rates (%) on training examples for different models of `INFIMNIST` digits. The latter were computed on non-overlapping sliding windows of 50K examples.

when the training data is augmented with more examples as described above. Fig. 3 (*right*) shows the number of linearly separable pairs of digits as a function of the number of `INFIMNIST` training examples. Note that only only one pair of digits (`0` versus `1`) remains linearly separable when the training set contains as few as 500K examples. This suggests a role for models of higher capacity.

Our experimental setup[2] was straightforward. We trained linear and `DoS` models using the updates, respectively, from eqs. (2) and (18–19). We initialized the parameters of the linear model with zero values and those of the `DoS` models with small random values. Specifically, we sampled the elements of $\mathbf{U}$ and $\mathbf{V}$ from a zero-mean normal distribution whose variance was inversely proportional to the number of elements in these matrices. (Note that the `DoS` models require a nonzero initialization to break the symmetry between different hidden units.) We computed ten-way error rates from the majority votes of $45 = \frac{1}{2}(10 \cdot 9)$ pairwise classifiers. There were no other sources of randomness in the experiments other than the seed for the random number generator. Though not shown here, we verified on the first 10M examples that the results varied little across five different values of this seed.

Fig. 4 compares the cumulative and moving error rates on training examples from different models. The moving error rates were computed on non-overlapping sliding windows of 50K examples. The results are shown for the linear model as well as `DoS` models with hidden vectors $\mathbf{h}_{\pm}$ of dimensionality $d \in \{1, 2, 4, 8, 16\}$. It is clear that the `DoS` models with higher capacity learn more accurate classifiers. In addition, as one might expect, the error rates stabilize more quickly for smaller models. For the largest model, the error rates are still decreasing after 100M examples, showing that even at this stage they are still improving.

Fig. 5 compares the *test* error rates from models sampled at intervals of 50K training examples. The error rates of non-averaged models (*left*) are extremely variable, especially for the smaller models. However, the results stabilize quickly when the models are averaged across time (*right*). In addition, it is clear that the averaged models (*right*) generalize much better than the non-averaged ones (*left*).

Overall the results demonstrate that `DoS` models of low capacity can learn much more accurate classifiers than linear models. In addition, these improvements are obtained from online updates that are nearly as simple to implement; compare eqs. (18–19) to eq. (2). It is curious that the linear model slightly outperforms the `DoS` model with $d = 1$, even though the latter contains the former as a special case. We do not understand this effect, but we did observe it consistently in the course of our work.

5 Conclusion

In this paper we have investigated a family of low-rank models for quadratic classification. These models provide a natural bridge between purely linear and fully quadratic classifiers, and they are especially straightforward to learn in an online setting. The models should enjoy the same broad

[2] In total we purchased several hundred CPU-hours on an externally managed cluster of Intel Xeon Gold 6132 processors. Much of this was expended on prototyping, debugging, and other intermediate or non-results.

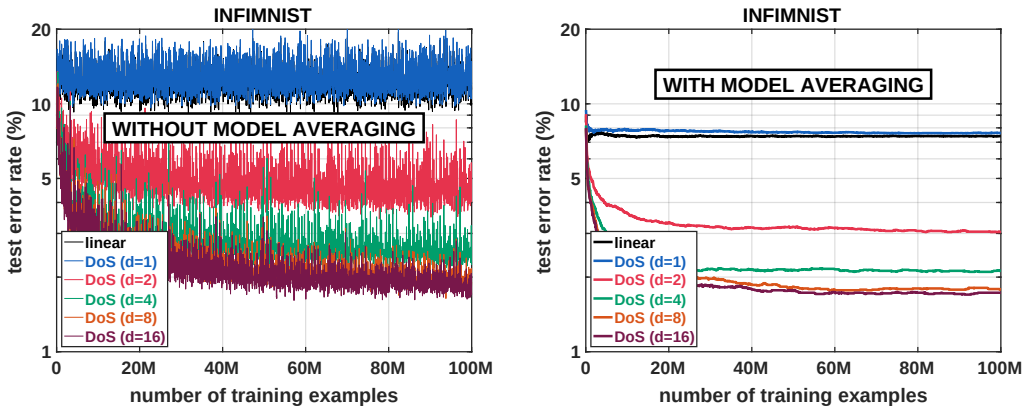

Figure 5: Test error rates (%) for different models of `INFIMNIST` digits as a function of the number of training examples. The models on the right were averaged across time; those on the left were not.

applicability as other low-rank models of high-dimensional quadratic forms. There is a large literature, for example, on latent variable models of factor analysis [68, 69], which are used to estimate very large but non-diagonal covariance matrices. Our models can play an analogous role for problems in high-dimensional classification, thus providing a useful addition to the practitioner's toolbox.

The models in this paper also raise several questions deserving of further theoretical study. For instance, many formal results are available for passive-aggressive learning of linear classifiers, including regret bounds on the cumulative number of mistakes [3]. It remains to show whether these bounds can be transferred to `DoS` classifiers. It would also be interesting to derive some sort of (*Lorentztron?*) convergence theorem for the realizable setting.

Another direction is to explore the issue of over-parameterization [20]. As noted earlier, the parameter space of `DoS` models is characterized by continuous symmetries of rotations and Lorentz transformations, and these symmetries give rise to low-dimensional manifolds of global minima for any margin-based loss function. It has been argued that such manifolds—and the non-convex landscapes they imply—are an essential (though not yet fully understood) feature[3] of deep learning [22]. Perhaps `DoS` models can serve as a vehicle to analyze larger questions of convergence and generalization in these landscapes. An interesting step in this direction has been taken by recent theoretical work on shallow (but wide) neural networks with quadratic activations [21].

In pursuing this work, we realize that any technological advance carries the risk of negative societal impacts. There may be settings (e.g., battlefields, markets, networks) where better online algorithms could be exploited by adversaries to overcome static or slowly evolving defenses. More generally, any classifier of sensitive data can be misapplied, deployed with bias, or deliberately put to misuse [70–72]. By raising awareness of these negative impacts, however, we can hope to forestall them.

We conclude by observing that the fields of machine learning and optimization are inextricably linked. The simplest linear models [73–75] are powered by least-squares methods and singular value decompositions [66]. With extra machinery, these models can be made more robust [76–78] or interpretable [79] or even extended to nonlinear settings [80]; these generalizations tend to be formulated as convex quadratic or semidefinite programs [81]. In machine learning as a whole, optimizations that once seemed exotic are now commonplace. In the study of passive-aggressive learning algorithms, this paper has found a role for nonconvex but tractable QCQPs. It seems likely that other useful models, yet to be formulated, can benefit from these methods.

Acknowledgements

We thank the reviewers in advance for their time and consideration, and we welcome all comments and suggestions for improvement.

[3] As in neural networks with ReLU activation functions [20], these manifolds emerge in `DoS` models independently of the size of the data set: i.e. they arise even when there are fewer parameters than examples.

References

[1] L. Bottou and Y. LeCun. Large scale online learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 217–224. MIT Press, 2004.

[2] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[3] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[4] M. A. Aizerman, É. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[5] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[6] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2001.

[7] G. Loosli, S. Cani, 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, 2007.

[8] J. Lu, S. C. H. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu. Large scale online kernel learning. *Journal of Machine Learning Research*, 17(47):1–43, 2016.

[9] K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 225–232. MIT Press, 2004.

[10] J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. In *Proceedings of the 8th International Conference on AI and Statistics (AISTATS-05)*, pages 413–420, 2005.

[11] G. Cavallanti, N.Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2):143–167, 2007.

[12] O. Dekel, S. S.-Shwartz, and Y. Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

[13] F. Orabona, J. Keshet, and B. Caputo. Bounded kernel-based online learning. *Journal of Machine Learning Research*, 10(92):2643–2666, 2009.

[14] Z. Wang, K. Crammer, and S. Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training. *Journal of Machine Learning Research*, 13(100):3103–3131, 2012.

[15] Z. Wang and S. Vucetic. Online passive-aggressive algorithms on a budget. In *Proceedings of the 13th International Conference on AI and Statistics (AISTATS-10)*, pages 908–915, 2010.

[16] J. Lu, P. Zhao, and S. C. H. Hoi. Online sparse passive aggressive learning with kernels. In *Proceedings of the 2016 SIAM International Conference on Data Mining (SDM-16)*, pages 675–683, 2016.

[17] C.-H. Wu, H. H.-S. Lu, and H.-M. Hang. Budgeted passive-aggressive learning for online multiclass classification. *IEEE Access*, 8:227420–227437, 2020.

[18] G. E. Hinton, A. Krizhevsky, and S. D Wang. Transforming auto-encoders. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-11)*, pages 44–51. Springer, 2011.

[19] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3856–3866. Curran Associates, Inc., 2017.

[20] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*, pages 1019–1028, 2017.

[21] M. Soltanolkotabi, A. Javanmard, and J. D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[22] C. Liu, L. Zhu, and M. Belkin. Towards a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. arXiv preprint arXiv:2003.00307.

[23] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

[24] A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume XII, pages 615–622, 1962.

[25] N. Littlestone. *Mistake bounds and logarithmic linear-threshold learning algorithms*. PhD thesis, UC Santa Cruz, 1989.

[26] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, 1997.

[27] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296, 1999.

[28] C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

[29] M. Herbster. Learning additive models online with fast evaluating kernels. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 444–460, 2001.

[30] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2002.

[31] Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387, 2002.

[32] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

[33] B. Li, P. Zhao, S. C. H. Hoi, and V. Gopalkrishnan. PAMR: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 87(2):221–258, 2012.

[34] M. Blondel, Y. Kubo, and N. Ueda. Online passive-aggressive algorithms for nonnegative matrix factorization and completion. In *Proceedings of the 17th International Conference on AI and Statistics (AISTATS-14)*, pages 96–104, 2014.

[35] J. Lu, P. Zhao, and S. C. H. Hoi. Online passive-aggressive active learning. *Machine Learning*, 103:141–183, 2016.

[36] M. T. Bahadori. Spectral capsule networks, 2018. Workshop at the International Conference on Learning Representations (ICLR-18).

[37] K. Duarte, Y. Rawat, and M. Shah. Videocapsulenet: A simplified network for action detection. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7610–7619. Curran Associates, Inc., 2018.

[38] G. E. Hinton, S. Sabour, and N. Frosst. Matrix capsules with EM routing. In *Proceedings of the International Conference on Learning Representations (ICLR-18)*, 2018.

[39] Dilin Wang and Qiang Liu. An optimization view on dynamic routing between capsules, 2018. Workshop at the International Conference on Learning Representations (ICLR-18).

[40] L. Zhang, M. Edraki, and G.-J. Qi. Cappronet: Deep feature learning via orthogonal projections onto capsule subspaces. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5814–5823. Curran Associates, Inc., 2018.

[41] K. Ahmed and L. Torresani. STAR-Caps: Capsule networks with straight-through attentive routing. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9101–9110. Curran Associates, Inc., 2019.

[42] T. Hahn, M. Pyeon, and G. Kim. Self-routing capsule networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7658–7667. Curran Associates, Inc., 2019.

[43] T. Jeong, Y. Lee, and H. Kim. Ladder capsule network. In *Proceedings of the 36th International Conference on Machine Learning (ICML-19)*, pages 3071–3079, 2019.

[44] A. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton. Stacked capsule autoencoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15512–15522. Curran Associates, Inc., 2019.

[45] Zhang Xinyi and Lihui Chen. Capsule graph neural network. In *Proceedings of the International Conference on Learning Representations (ICLR-19)*, 2019.

[46] Y. Qin, N. Frosst, S. Sabour, C. Raffel, G. Cottrell, and G. E. Hinton. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. In *Proceedings of the International Conference on Learning Representations (ICLR-20)*, 2020.

[47] Y.-H. H. Tsai, N. Srivastava, H. Goh, and R. Salakhutdinov. Capsules with inverted dot-product attention routing. In *Proceedings of the International Conference on Learning Representations (ICLR-20)*, 2020.

[48] S. R. Venkataraman, S. Balasubramanian, and R. R. Sarma. Building deep equivariant capsule networks. In *Proceedings of the International Conference on Learning Representations (ICLR-20)*, 2020.

[49] L. K. Saul. An EM algorithm for capsule regression. *Neural Computation*, 33(1):194–226, 2021.

[50] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[51] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[52] M. Leshno, V. Ya, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[53] A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[54] E. F. Taylor and J. A. Wheeler. *Spacetime Physics*. W. H. Freeman, 1992.

[55] E. Feron. Nonconvex quadratic programming, semidefinite relaxations and randomization algorithms in information and decision systems. In T. E. Djaferis and I. C. Schick, editors, *System Theory*, pages 255–274. Springer, 2000.

[56] Y. Ye and S. Zhang. New results on quadratic minimization. *SIAM Journal on Optimization*, 14(1):245–267, 2003.

[57] Z.-Q. Luo, W.-K. Ma, A. M.-C So, Y. Ye, and S. Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, 2010.

[58] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, 1994.

[59] I. Pólik and T. Terlaky. A survey of the s-lemma. *SIAM Review*, 49(3):371–418, 2007.

[60] J. Park and S. Boyd. General heuristics for nonconvex quadratically constrained quadratic programming, 2017. arXiv preprint arXiv:1703.07870.

[61] J. J. Moré. Generalizations of the trust region problem. *Optimization Methods and Software*, 2(3-4):189–209, 1993.

[62] J.-M. Feng, G.-X. Lin, R.-L. Sheu, and Y. Xia. Duality and solutions for quadratic programming over single non-homogeneous quadratic constraint. *Journal of Global Optimization*, 54(2):275–293, 2012.

[63] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious URLs: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML-09)*, pages 681–688, 2009.

[64] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 71–79, 2013.

[65] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31:1–10, 1966.

[66] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

[67] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[68] D. B. Rubin and D. T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.

[69] D. Bartholomew, M. Knott, and I. Moustaki. *Latent Variable Models and Factor Analysis: A Unified Approach*. Wiley, 2011.

[70] J. Buolamwini and T. Gebru. Gender shades: intersectional accuracy disparities in commercial gender classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency (FACCT-18)*, pages 77–91, 2018.

[71] I. D. Raji, T. Gebru, M. Mitchell, J. Buolamwini, J. Lee, and E. Denton. Saving face: investigating the ethical concerns of facial recognition auditing. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES-20)*, pages 145–151, 2020.

[72] R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth. Fairness in criminal justice risk assessments: the state of the art. *Sociological Methods & Research*, 50(1):3–44, 2021.

[73] S. Weisberg. *Applied Linear Regression*. Wiley, 2013.

[74] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[75] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[76] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[77] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[78] E. J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):1–37, 2009.

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

[80] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90, 2006.

[81] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   In our view both the abstract and introduction accurately reflect the paper as a whole.

   (b) Did you describe the limitations of your work? [Yes]

   One limitation is that the proposed models are not universal function approximators; see section 3.1. Another limitation is that the linear model seems to outperform the lowest-rank quadratic model, which we observe but cannot explain; see the last paragraph of section 4. A third limitation is that models need to be averaged across time to obtain a single, deployable model: see Figure 5. A final limitation is that we do not yet have regret bounds for the passive-aggressive updates in these models; see the second paragraph of section 5.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

   See the penultimate paragraph of section 5.

   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

   The paper conforms to all of the published guidelines. The proposed method is evaluated on a data set that was made publicly available for research and that has not been associated with any personally identifiable or sensitive information.

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

   The paper contains no proofs, but several calculations. Every attempt was made to state the full set of assumptions behind each calculation (e.g., the footnote on page 5).

   (b) Did you include complete proofs of all theoretical results? [Yes]

   The calculations should provide enough details to be followed and reproduced by a typical reader. In general where further details might help, the reader is referred to existing papers that provide them; see section 3 before eq. (16). One exception is that we omitted the derivation of eq. (15), which is a tedious (but not difficult) exercise in basic calculus and linear algebra. The paper is intended to be complete without a supplementary appendix.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

   We have submitted code and instructions to reproduce the main experimental results in the supplemental material.

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

   We explained carefully how all models were initialized. The passive-aggressive updates in this paper do not have hyperparameters (e.g., learning rates, momentum parameters, mini-batch sizes). There were no data splits: the models were evaluated in an online manner over an extremely large number (100M) of distinct examples.

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

   We verified that the results on 10M examples were not materially different for five

different values of the random seed; this is reported in section 4. But we were not able to experiment with multiple seeds on the full data set of 100M examples.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

See the footnote in section 4.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

The paper cites the researchers [7, 67] that introduced the `INFIMNIST` and `MNIST` data sets into the public domain.

(b) Did you mention the license of the assets? [N/A]

These data sets appear to be distributed without a license.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

We include sample code in the supplemental material. The code is for the benefit of the reviewers and the program committee, but like the paper, it is not meant to be publicly distributed while under review.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

The data sets that we use have been in the public domain for a long while.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

The data does not contain any personally identifiable information or offensive content.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

This work did not use crowdsourcing or conduct research with human subjects.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

This work did not use crowdsourcing or conduct research with human subjects.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

This work did not use crowdsourcing or conduct research with human subjects.