Adaptive Scale-Invariant Online Algorithms for Learning Linear Models

Michał Kempka [1] **Wojciech Kotłowski** [1] **Manfred K. Warmuth** [2]

Abstract

We consider online learning with linear models, where the algorithm predicts on sequentially revealed instances (feature vectors), and is compared against the best linear function (comparator) in hindsight. Popular algorithms in this framework, such as Online Gradient Descent (OGD), have parameters (learning rates), which ideally should be tuned based on the scales of the features and the optimal comparator, but these quantities only become available at the end of the learning process. In this paper, we resolve the tuning problem by proposing online algorithms making predictions which are invariant under arbitrary rescaling of the features. The algorithms have no parameters to tune, do not require any prior knowledge on the scale of the instances or the comparator, and achieve regret bounds matching (up to a logarithmic factor) that of OGD with optimally tuned separate learning rates per dimension, while retaining comparable runtime performance.

1. Introduction

We consider the problem of online learning with linear models, in which at each trial $t = 1, \dots, T$, the algorithm receives an input instance (feature vector) $\boldsymbol{x}_t \in \mathbb{R}^d$, upon which it predicts $\widehat{y}_t \in \mathbb{R}$. Then, the true label y_t is revealed and the algorithm suffers loss $\ell(y_t, \widehat{y}_t)$, convex in $\widehat{y}_t$. The goal of the algorithm is have its cumulative loss not much larger to that of any linear predictor of the form $\boldsymbol{x} \mapsto \boldsymbol{x}^\top \boldsymbol{u}$ for $\boldsymbol{u} \in \mathbb{R}^d$, i.e. to have small *regret* against any comparator $\boldsymbol{u}$. This problem encompasses linear regression and classification (with convex surrogate losses) and has been extensively studied in numerous past works (Littlestone et al., 1991; Cesa-Bianchi et al., 1996; Shalev-Shwartz, 2011; Hazan, 2015).

One of the most popular algorithms in this framework is Online Gradient Descent (OGD) (Cesa-Bianchi et al., 1996). Its predictions are given by $\widehat{y}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_t$ for a weight vector $\boldsymbol{w}_t \in \mathbb{R}^d$ updated using a simple rule

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla_t, \tag{1}$$

where ∇_t is a (sub)gradient of the loss $\ell(y_t, \widehat{y}_t)$ at $\boldsymbol{w}_t$, and η is a parameter of the algorithm, called the learning rate. With the optimal "oracle" tuning of η (which involves the norm of the comparator $\boldsymbol{u}$ and of the observed gradients, unknown to the algorithm in advance), OGD would achieve a bound on the regret against $\boldsymbol{u}$ of order $\|\boldsymbol{u}\| \sqrt{\sum_t \|\nabla_t\|^2}$ (Zinkevich, 2003). Unfortunately, this bound might be very poor if the features have distinct scales. To see that first note that ∇_t is proportional to $\boldsymbol{x}_t$ due to linear dependence of $\widehat{y}_t$ on $\boldsymbol{x}_t$. Now, let $\boldsymbol{u}$ be the comparator which minimizes the total loss (assume such exists). If we scale the first coordinate of each $\boldsymbol{x}_t$ by a factor of c, the first coordinate of the optimal comparator $\boldsymbol{u}$ will scale down by a factor of c^{-1}, so its prediction and (optimal) loss remain the same, and the bound above will in general become worse by a factor of $\max\{c, c^{-1}\}$ (Ross et al., 2013). This is a well known issue with gradient descent, and is usually solved by prior normalization of the features. However, such pre-processing step cannot be done in an online setting.

The problem described above becomes apparent if we make an analogy from physics and imagine that all features have physical *units*. In particular, if we assigned a unit $[x_i]$ to feature i, and assumed for simplicity that the prediction and the label are unitless (as in, e.g., classification), the corresponding coordinate of the weight vector would need to have unit $1/[x_i]$. However, the units in the OGD update (1) are mismatched, because $\nabla_{t,i}$ has unit $[x_i]$ (as ∇_t is proportional to $\boldsymbol{x}_t$), while $w_{t,i}$ has unit $1/[x_i]$; even assigning a unit to η does not help as a single number cannot compensate different units. A reasonable solution to this "unit clash" problem is to use *one learning rate per dimension*, i.e. to modify (1) to:

$$w_{t+1,i} = w_{t,i} - \eta_i \nabla_{t,i}, \qquad i = 1, \dots, d. \tag{2}$$

If we choose the oracle tuning of the learning rates to minimize the regret against comparator $\boldsymbol{u}$, it follows that

[1]Poznan University of Technology, Poznan, Poland [2]Google Inc. Zürich & UC Santa Cruz. Correspondence to: Michał Kempka <mkempka@cs.put.poznan.pl>, Wojciech Kotłowski <wkotlowski@cs.put.poznan.pl>, Manfred K. Warmuth <manfred@ucsc.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

$\eta_i^* = |u_i|/\sqrt{\sum_t \nabla_{t,i}^2}$ which results in the regret bound of order $\sum_i |u_i|\sqrt{\sum_t \nabla_{t,i}^2}$, better than the bound obtained with a single learning rate. Interestingly, the unit of η_i^* becomes $1/[x_i]^2$, which fixes the "unit clash" in (2) and makes the scaling issues go away (as now scaling the i-the feature by any factor c will be compensated by scaling down u_i by c^{-1}). Unfortunately, it is infeasible in practice to separately tune a single learning rate per dimension (oracle tuning requires the knowledge the comparator and all future gradients).

Our contribution. In this paper we provide adaptive online algorithms which for any comparator $\boldsymbol{u}$ achieve regret bounds matching, up to logarithmic factors, that of OGD with optimally tuned separate learning rates per dimension. Note that as we want to capture arbitrary feature scales and comparators, our bounds come *without any prior assumptions on the magnitude of instances* $\boldsymbol{x}_t$*, comparator* $\boldsymbol{u}$*, or even predictions* $\boldsymbol{x}_t^\top \boldsymbol{u}$, as has been commonly assumed in the past work (we do, however, assume the Lipschitzness of the loss with respect to the prediction, which is satisfied for various popular loss functions, such as logistic, hinge or absolute losses[1]). Our algorithms achieve their bounds without the need to tune any hyperparameters, and have runtime performance of $O(d)$ per iteration, which is the same as that of OGD. As a by-product of being adaptive to the scales of the instances and the comparator, the proposed algorithms are *scale-invariant*: their predictions are invariant under arbitrary rescaling of individual features (Ross et al., 2013). More precisely, after multiplying the i-th coordinate of all input instances by a fix scaling factor a_i, $x_{t,i} \mapsto a_i x_{t,i}$ for all t, the predictions of the algorithms remain the same: they are independent on the units in which the instance vectors are expressed (in particular, they do do not require any prior normalization of the data). To achieve our goals, the design of our algorithms heavily rely on techniques recently developed in adaptive online learning (Streeter & McMahan, 2012; Orabona & Pál, 2016; Cutkosky & Boahen, 2017; Cutkosky & Orabona, 2018).

The first algorithm achieves a regret bound which depends on instances only relative to the scale of the comparator, through products of the form $|u|_i\sqrt{\max_t |x_{t,i}|^2 + \sum_t \nabla_{t,i}^2}$ for $i = 1, \dots, d$, similarly as in the bound of OGD with per-dimension learning rates (with additional maximum over feature values, which is usually much smaller than the sum over squared gradients). As the algorithm can be sometimes a bit conservative in its predictions, we also introduce a second algorithm which is more aggressive in decreasing its cumulative loss; the price to pay is a regret bound which mildly (logarithmically) depends on ratios between the largest and the first non-zero input value for each coordinate. While these quantities can be made arbitrarily large in the worst case, it is unlikely to happen in practice. We test both algorithms in a computational study on several real-life data sets and show that without any need to tune parameters, they are competitive to popular online learning methods, which are allowed to tune their learning rates to optimize the test set performance.

[1]Lipschitzness *does not* imply any bound on the gradients ∇_t which are proportional to feature vectors: $\nabla_t = g_t \boldsymbol{x}_t$ for some $g_t \in \mathbb{R}$; it only implies a bound on the proportionality constant g_t.

Related work. Our work is rooted from a long line of research on regret minimizing online algorithms (Cesa-Bianchi et al., 1996; Kivinen & Warmuth, 1997; Cesa-Bianchi & Lugosi, 2006). Most of the proposed methods have "range factors" present both in the algorithm and in the bound: it is typically assumed that some prior knowledge on the range of the comparator and the gradients is given, which allows the algorithm to tune its parameters appropriately. For instance, assuming $\|\boldsymbol{u}\| \le U$ and $\|\nabla_t\| \le G$ for all t, OGD (1) with learning rate $\eta = U/(G\sqrt{T})$ achieves $O(UG\sqrt{T})$ regret bound.

More recent work on adaptive algorithms aims to get rid of these range factors. In particular, with a prior bound on the comparator norm, it is possible to adapt to the unknown range of the gradients (Duchi et al., 2011; Orabona & Pál, 2015), whereas having a prior bound on all future gradients, one can adapt to the unknown norm of the comparator (Streeter & McMahan, 2012; McMahan & Abernethy, 2013; Orabona, 2014; Orabona & Pál, 2016; Orabona & Tommasi, 2017; Cutkosky & Orabona, 2018). In particular, using reduction methods proposed by Cutkosky & Orabona (2018), one can get a bound matching OGD with separate learning rate per dimension, but this requires to know $\max_{t,i} |\nabla_{t,i}|$ in advance. Interestingly, Cutkosky & Boahen (2017) have shown that in online convex optimization it is not possible to adapt to both unknown gradient range and unknown comparator norm at the same time. Here, we circumvent this negative result by exploiting the fact that the input instance $\boldsymbol{x}_t$ is available *ahead* of prediction and therefore can be used to construct $\widehat{y}_t$ (this idea was first discovered in the context of linear regression (Vovk, 2001; Azoury & Warmuth, 2001)).

Scale-invariant algorithm has been been studied by Ross et al. (2013); Orabona et al. (2015) in a setup very similar to ours. Their algorithms, however, require a prior knowledge on the largest per-coordinate comparator's prediction, $\max_{t,i} |u_i x_{t,i}|$, whereas their bounds scale with relative ratios between the largest and the first non-zero input value for each coordinate (the bound of our second algorithm also depends on these quantities but only in a logarithmic way). Luo et al. (2016); Koren & Livni (2017) considered even a more general setup of invariance under linear transformation of features (of which our invariance is a special case

if the transformation is diagonal), but a prior knowledge of $\max_t |\boldsymbol{x}_t^\top \boldsymbol{u}|$ must be available, and the resulting algorithms are second-order methods. The closest to our work are the results by Kotłowski (2017), which concern the same setup, general invariance under linear transformations, and, similarly to us, make no prior range assumptions. Their bounds, however, do not scale with gradients $\nabla_{t,i}^2$ (as in the optimal OGD bound), but with the *size of the features* $x_{t,i}^2$ (multiplied by the Lipschitz constant of the loss), which upper-bounds $\nabla_{t,i}^2$ and can become much larger. For instance, in the "noise-free" case, when some comparator $\boldsymbol{u}$ has zero loss, the algorithm playing sufficiently close to $\boldsymbol{u}$ can inflict arbitrarily small gradients, while the sum of squared feature values will still grow linearly in t.

The goal of scale-invariance seems to go hand in hand with a requirement for the updates to avoid unit clashes and this connection was the motivating idea for our work. In the most basic case, assume you want to design online algorithms for linear regression

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)\boldsymbol{x}_t$$

that are to be robust to scaling the input vectors $\boldsymbol{x}_t$ by a single positive constant factor. In this case $[\eta]$ should be $1/[\|\boldsymbol{x}_t\|^2]$. Interestingly enough, good tunings of the learning rates η often "fix the units": the properly tuned learning rates for the linear regression updates employed in (Cesa-Bianchi et al., 1996; Kivinen & Warmuth, 1997) have units $1/[\|\boldsymbol{x}_t\|^2]$. In this paper we focus on robustness to independently scaling the individual components $x_{t,i}$ of the input vectors by positive factors. This requires privatized learning rates η_i with the property that $[\eta_i] = 1/[x_{t,i}]^2$. Our paper focuses on this case because of efficiency concerns. However there is a third case (more expensive) where we want robustness to independent scaling and rotation of the input vectors $\boldsymbol{x}_t$. Now $\boldsymbol{\eta}$ must be a matrix parameter (playing a similar role to a Hessian) and if the instances are pre-multiplied by a fixed invertible $\boldsymbol{A}$, then the tuned learning rate matrix of the new instances must become $\boldsymbol{\eta}\boldsymbol{A}^{-1}$, thus correcting for the pre-multiplication with $\boldsymbol{A}$. The updates of Luo et al. (2016); Koren & Livni (2017); Kotłowski (2017), as well as the Newton algorithm, have this form, but they are all second order algorithms with runtime of at least $O(d^2)$ per trial.

2. Problem Setting

Our online learning protocol is defined as follows. In each trial $t = 1, \ldots, T$, the algorithm receives an input instance $\boldsymbol{x}_t \in \mathbb{R}^d$, on which it predicts $\widehat{y}_t \in \mathbb{R}$; we will always assume linear predictions $\widehat{y}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_t$, where $\boldsymbol{w}_t \in \mathbb{R}^d$ is allowed to depend on $\boldsymbol{x}_t$. Then, the output label $y_t \in \mathcal{Y}$ is revealed, and the algorithm suffers loss $\ell(y_t, \widehat{y}_t)$. As we make no assumptions about the label set $\mathcal{Y}$, in what follows we incorporate y_t into the loss function and use $\ell_t(\widehat{y})$ to denote $\ell(y_t, \widehat{y})$. The performance of the algorithm is measured by means of the *regret*:

$$R_T(\boldsymbol{u}) = \sum_{t=1}^T \ell_t(\boldsymbol{x}_t^\top \boldsymbol{w}_t) - \sum_{t=1}^T \ell_t(\boldsymbol{x}_t^\top \boldsymbol{u}),$$

which is the difference between the cumulative loss of the algorithm and that of a fixed, arbitrarily chosen, comparator weight vector $\boldsymbol{u} \in \mathbb{R}^d$ (for instance, $\boldsymbol{u}$ can be the minimizer of the cumulative loss on the whole data sequence, if such exists).

Loss function	$\ell(y, \widehat{y})$	$\partial_{\widehat{y}} \ell(y, \widehat{y})$	L
logistic	$\ln\left(1 + e^{-y\widehat{y}}\right)$	$\frac{-y}{1+e^{y\widehat{y}}}$	1
hinge	$\max\{0, 1 - y\widehat{y}\}$	$-y\mathbf{1}[y\widehat{y} \le 1]$	1
absolute	$\lvert\widehat{y} - y\rvert$	$\operatorname{sgn}(\widehat{y} - y)$	1

Table 1. L-Lipschitz loss functions for classification and regression. $\mathbf{1}[\cdot]$ denotes an indicator function.

We assume that for any t, $\ell_t(\widehat{y})$ is convex and L-Lipschitz; the latter implies that the (sub)derivative of the loss is bounded, $|\partial \ell_t(\widehat{y})| \le L$. Table 1 lists three popular losses with these properties. Throughout the paper, we assume $L = 1$ without loss of generality. Our setup can be considered as a variant of online convex optimization (Shalev-Shwartz, 2011; Hazan, 2015), with the main difference in $\boldsymbol{x}_t$ being observed before prediction.

We use a standard argument exploiting the convexity of the loss to bound $\ell_t(\widehat{y}') \ge \ell_t(\widehat{y}) + \partial \ell_t(\widehat{y})(\widehat{y}' - \widehat{y})$ for any $\widehat{y}, \widehat{y}' \in \mathbb{R}$. Substituting $\widehat{y} = \boldsymbol{w}_t^\top \boldsymbol{x}_t$ and $\widehat{y}' = \boldsymbol{u}^\top \boldsymbol{x}_t$, and denoting $g_t = \partial \ell_t(\widehat{y}_t) \in [-1, 1]$ for each t, the regret is upper-bounded by:

$$R_T(\boldsymbol{u}) \le \sum_{t=1}^T g_t \boldsymbol{x}_t^\top (\boldsymbol{w}_t - \boldsymbol{u}), \tag{3}$$

where $|g_t| \le 1$ follows from the Lipschitzness of the loss. Note that $g_t \boldsymbol{x}_t$ is equal to $\nabla_t = \nabla_{\boldsymbol{w}_t} \ell_t(\boldsymbol{x}_t^\top \boldsymbol{w}_t)$, the (sub)gradient of the loss with respect to the weight vector $\boldsymbol{w}_t$. Thus, we can bound the regret with respect to the original convex loss $\ell_t(\boldsymbol{x}_t^\top \boldsymbol{w})$ by upper-bounding its linearized version $g_t \boldsymbol{x}_t^\top \boldsymbol{w}$ on the right-hand side of (3).

Consider running Online Gradient Descent (OGD) algorithm on this problem, as defined in (2), i.e. we let the algorithm have a separate learning rate per dimension. When initialized at $\boldsymbol{w}_1 = \mathbf{0}$, OGD achieves the regret bound:

$$R_T(\boldsymbol{u}) \le \sum_{i=1}^d \left(\frac{u_i^2}{2\eta_i} + \frac{\eta_i}{2} S_{T,i}^2 \right),$$

where we introduced $S_{t,i}^2 = \sum_{j \le t} \nabla_{j,i}^2 = \sum_{j \le t} (g_j x_{j,i})^2$. This is a slight generalization of a standard textbook bound

(see, e.g., Hazan, 2015), proven in Appendix A for completeness. Tuning the learning rates to minimize the bound results in $\eta_i = \frac{|u_i|}{S_{t,i}}$, and the bound simply becomes:

$$R_T(\boldsymbol{u}) \leq \sum_{i=1}^{d} |u_i| S_{T,i}. \tag{4}$$

Such tuning is, however, not directly feasible as it would require knowing the comparator and the future gradients in hindsight. The goal of this work is to design adaptive online algorithms which for any comparator $\boldsymbol{u}$, and any data sequence $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^T$, without any prior knowledge on their magnitudes, achieve (4) up to logarithmic factors.

An interesting property of bound (4) is that it captures a natural symmetry of our linear framework. Given a data sequence, let $\boldsymbol{u}$ be the minimizer of the cumulative loss, $\boldsymbol{u} = \operatorname{argmin}_{\boldsymbol{w}} \sum_t \ell_t(\boldsymbol{x}_t^\top \boldsymbol{w})$ (assume such exists). If we apply a coordinate-wise transformation $x_{t,i} \mapsto a_i x_{t,i}$ simultaneously to all input instances ($t = 1, \ldots, T$) for any positive scaling factors $a_1, \ldots, a_d$, the minimizer of the loss will undergo the inverse transformation $u_i \mapsto a_i^{-1} u_i$ to keep its predictions $\boldsymbol{x}_t^\top \boldsymbol{u}$, and thus its cumulative loss, *invariant*. Indeed, $\boldsymbol{u}$ minimizes $\sum_t \ell_t(\boldsymbol{x}_t^\top \boldsymbol{w})$ if and only if $\boldsymbol{A}^{-1}\boldsymbol{u}$ minimizes $\sum_t \ell_t((\boldsymbol{A}\boldsymbol{x}_t)^\top \boldsymbol{w})$ for $\boldsymbol{A} = \operatorname{diag}(a_1, \ldots, a_d)$. Thus, when (4) is evaluated at the loss minimizer, it becomes invariant under any such scale transformation.

The invariance of predictions of the optimal comparator leads to the definition of scale-invariant algorithms. We call a learning algorithm *scale-invariant* if its behavior (sequence of predictions) is invariant under arbitrary rescaling of individual features (Ross et al., 2013; Kotłowski, 2017). More precisely, if we apply a transformation $x_{t,i} \mapsto a_i x_{t,i}$ simultaneously to all instances, the predictions of the algorithm $\widehat{y}_1, \ldots, \widehat{y}_T$ remain the same as on the original data sequence. Scale-invariant algorithm are thus independent on the "units" in which the instance vectors are expressed on each feature, and do not require any prior normalization of the data. Interestingly, OGD defined in (2) is not a scale invariant algorithm, but becomes one under the optimal tuning of its learning rates. The algorithms presented in the next section will turn out to be scale-invariant, essentially as a by-product of adaptiveness to arbitrary scale of the comparator and the instances, required to achieve (4).

Remark: As noted in the introduction, scale invariance can be generalized to arbitrary linear invertible transformations $\boldsymbol{x}_t \mapsto \boldsymbol{A}\boldsymbol{x}_t$. Unfortunately, this leads to second-order algorithms (Luo et al., 2016; Koren & Livni, 2017; Kotłowski, 2017), with the complexity at least $\Theta(d^2)$ per trial.

3. Scale-Invariant Algorithms

Motivation. We first briefly describe the motivating idea behind the construction of the algorithms. We start with rewriting the right hand side of (3) to get:

$$R_T(\boldsymbol{u}) \leq \sum_{i=1}^{d} \underbrace{\Big(\sum_{t=1}^{T} g_t x_{t,i} (w_{t,i} - u_i) \Big)}_{\stackrel{def}{=} \tilde{R}_{T,i}(u_i)},$$

so that it decouples coordinate-wise and it suffices to separately bound each term $\tilde{R}_{T,i}(u_i)$ $i = 1, \ldots, d$. As we aim to get close to (4), we want for each i a bound of the form $\tilde{R}_{T,i}(u_i) \leq B(u_i S_{T,i}) + c_T$, for some function $B(\cdot)$ plus a potential additional overhead c_T (to exactly get (4) we could set $B(x) = |x|$ and $c_T = 0$, but this turns out to be unachievable without any prior knowledge on the comparator). Using $G_{t,i} = -\sum_{j \leq t} g_j x_{j,i}$ to denote the cumulative negative gradient coordinate, such bound can be equivalently written as:

$$\sum_{t=1}^{T} g_t x_{t,i} w_{t,i} + G_{T,i} u_i - B(u_i S_{T,i}) \leq c_T.$$

Now, the key idea is to note that the bound must hold for any comparator u_i, therefore it must hold if we take a supremum over u_i on the left-hand side, $\sup_{u_i}\{G_{T,i} u_i - B(u_i S_{T,i})\}$. To evaluate this supremum, we note that under variable change $x = u_i S_{T,i}$ it becomes equivalent to $\sup_x\{G_{T,i}/S_{T,i}\, x - B(x)\}$. Recalling the definition of the *Fenchel conjugate* of a function $f(x)$, defined as $f^*(\theta) = \sup_x\{\theta x - f(x)\}$ (Boyd & Vandenberghe, 2004), we see that the supremum can be evaluated to $B^*(G_{T,i}/S_{T,i})$. Thus, the unknown comparator has been eliminated from the picture, and the algorithm can be designed to satisfy:

$$\sum_{t=1}^{T} g_t x_{t,i} w_{t,i} + B^*(G_{T,i}/S_{T,i}) \leq c_T,$$

for every data sequence. In fact, we construct our algorithms by proceeding in the reverse direction: starting with an appropriate function ψ playing the role of B^* (which we call a *potential*) and getting bound expressed by means of its conjugate ψ^*. What we just described is known as *regret-reward duality* and has been successfully used in adaptive online learning (Streeter & McMahan, 2012; McMahan & Orabona, 2014; Orabona & Pál, 2016).

As already briefly mentioned, achieving (4), which corresponds to a bound with $B(x) = |x|$, is actually not possible: a negative result by Streeter & McMahan (2012) implies that the best one can hope for is $B(x) = O(|x|\sqrt{\ln(|x|)})$. We will show that our algorithm achieve a bound of a slightly weaker form $B(x) = O(|x| \ln(|x|))$, but still giving only a logarithmic overhead comparing to (4).

Algorithms. We propose two scale-invariant algorithms presented as Algorithm 1 (ScInOL_1 from Scale-Invariant

Algorithm 1: ScInOL$_1(\epsilon = 1)$

Initialize: $S^2_{0,i}, G_{0,i}, M_{0,i} \leftarrow 0, \beta_{0,i} \leftarrow \epsilon; i = 1, \dots, d$
for $t = 1, \dots, T$ **do**
  Receive $\boldsymbol{x}_t \in \mathbb{R}^d$
  for $i = 1, \dots, d$ **do**
    $M_{t,i} \leftarrow \max\{M_{t-1,i}, |x_{t,i}|\}$
    $\beta_{t,i} \leftarrow \min\{\beta_{t-1,i}, \epsilon(S^2_{t-1,i} + M^2_{t,i})/(x^2_{t,i}t)\}$
    $w_{t,i} = \frac{\beta_{t,i}\mathrm{sgn}(\theta_{t,i})}{2\sqrt{S^2_{t-1,i}+M^2_{t,i}}}\left(e^{|\theta_{t,i}|/2} - 1\right)$
      where $\theta_{t,i} = \frac{G_{t-1,i}}{\sqrt{S^2_{t-1,i}+M^2_{t,i}}}$
  Predict with $\widehat{y}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_{t,i}$, receive loss $\ell_t(\widehat{y}_t)$ and compute $g_t = \partial_{\widehat{y}_t}\ell_t(\widehat{y}_t)$
  for $i = 1, \dots, d$ **do**
    $G_{t,i} \leftarrow G_{t-1,i} - g_t x_{t,i}$
    $S^2_{t,i} \leftarrow S^2_{t-1,i} + (g_t x_{t,i})^2$

Online Learning) and Algorithm 2 (ScInOL$_2$). They require $O(d)$ operations per trial and thus match OGD in the computational complexity. Both algorithms keep track of the negative cumulative gradients $G_{t,i} = -\sum_{j=1}^t g_j x_{j,i}$, sum of squared gradients $S^2_{t,i} = \sum_{j=1}^t (g_j x_{j,i})^2$, and the maximum encountered input values $M_{t,i} = \max_{j \le t} |x_{j,i}|$. The weight formula is written to highlight that the cumulative gradients are only accessed through a unitless quantity $\frac{G_{t-1,i}}{\sqrt{S^2_{t-1,i}+M^2_{t,i}}}$; an additional factor $\frac{1}{\sqrt{S^2_{t-1,i}+M^2_{t,i}}}$ in the weights is to compensate for $x_{t,i}$ in the prediction. To simplify the pseudocode we use the convention that $\frac{0}{0} = 0$ and $\frac{c}{0} = \infty$ for $c > 0$. Note that since in each trial t, the algorithms have access to the input feature vector $\boldsymbol{x}_t$ *before* the prediction, they are able to update $M_{t,i}$ prior to computing the weight vector $\boldsymbol{w}_t$. Both algorithms decompose into d one-dimensional copies, one per each feature, which are coupled only by the values of g_t. Both algorithm have a parameter ϵ, but it only affect the constants and is set to 1 in the experiments. Scale invariance of the algorithms is verified in Appendix B.

ScInOL$_1$. The algorithm is based on a potential $\psi_{t,i}(x) = \beta_{t,i}(e^{|x|/(2\hat{S}_{t,i})} - \frac{|x|}{2\hat{S}_{t,i}} - 1)$ with $\hat{S}_{t,i} = \sqrt{S^2_{t,i} + M^2_{t,i}}$. The weight $w_{t,i}$ is chosen in such a way that the loss of the algorithm at trial t is upper-bounded by the change in the potential, for any choice of $x_{t,i} \in \mathbb{R}$ and $g_t \in [-1, 1]$:

$$w_{t,i} g_t x_{t,i} \le \psi_{t-1,i}(G_{t-1,i}) - \psi_{t,i}(G_{t,i}) + \delta_{t,i}, \qquad (5)$$

where $\delta_{t,i}$ is a small additional overhead. The algorithm resembles FreeRex by (Cutkosky & Boahen, 2017), because it actually uses the same functional form of the potential. The choice of the weight looks *almost* like a derivative of a potential function $\psi_{t-1,i}(x)$ at $x = G_{t-1,i}$, but it differs slightly in using $M_{t,i}$ rather than $M_{t-1,i}$ in its definition. This prior update of $M_{t,i}$ let the algorithm account for potentially very large value of $x_{t,i}$ and avoid incurring too much loss. The coefficients $\beta_{t,i}$ multiplying the potential are chosen to be a nonincreasing sequence, which at the same time keeps the overhead δ_t upper-bounded by $\frac{\epsilon}{t}$, in order to to avoid terms in the regret bound depending on ratios between feature values and get $\sum_t \delta_{t,i} \le \epsilon(1 + \ln T)$. Summing (5) over trials and using using $\psi_{0,i}(G_0) = 0$ gives:

$$\sum_t w_{t,i} g_t x_{t,i} - \sum_t \delta_{t,i} \le -\psi_{T+1,i}(G_{T,i}).$$

Using the convexity of $\psi_{T+1,i}$ we can rewrite it by means of its Fenchel conjugate, $\psi_{T+1,i}(G_{T,i}) = \sup_u\{G_{T,i}u - \psi^*_{T+1,i}(u)\}$, which in turn can be bounded as:

$$\psi^*_{T,i}(u) \le 2|u|\hat{S}_{T,i} \ln\left(1 + 2|u|\beta^{-1}_{T,i}\hat{S}_{T,i}\right)$$

Summing over features $i = 1, \dots, d$, bounding $\beta_{T,i} \ge (\epsilon T)^{-1}$, and using (3) gives:

Theorem 3.1. *For any $\boldsymbol{u} \in \mathbb{R}^d$ the regret of* ScInOL$_1$ *is upper-bounded by:*

$$R_T(\boldsymbol{u}) \le \sum_{i=1}^d \left(2|u_i|\hat{S}_{T,i} \ln\left(1 + \frac{2|u_i|\hat{S}_{T,i}T}{\epsilon}\right) + \epsilon(1 + \ln T)\right)$$
$$= \sum_{i=1}^d \tilde{O}(|u_i|\hat{S}_{T,i}),$$

where $\hat{S}_{T,i} = \sqrt{S^2_{T,i} + M^2_{T,i}}$ and $\tilde{O}(\cdot)$ hides the constants and logarithmic factors.

The full proof of Theorem 3.1 is given in Appendix C. Note that the bound depends on the scales of the features only relative to the comparator weights $\boldsymbol{u}$ through quantities $|u_i|\hat{S}_{T,i}$, and is equivalent to the optimal OGD bound (4) up to logarithmic factors.

ScInOL$_2$. The algorithm described in the previous section is designed to achieve a regret bound which depends on instances only relative to the scale of the comparator, no matter how extreme are the ratios $|x_{t,i}|/M_{t-1,i}$ between the new inputs and previously observed maximum feature values. We have observed that this can make the behavior of the algorithm too conservative, due to guarding against the worst-case instances. Therefore we introduce a second algorithm, which is more aggressive in decreasing its cumulative loss; the price to pay is a regret bound which mildly depends on ratios between feature values. The algorithm has a multiplicative flavor and resembles a family of Coin Betting algorithms recently developed by Orabona & Pál (2016); Orabona & Tommasi (2017); Cutkosky & Orabona (2018).

Algorithm 2: ScInOL$_2(\epsilon = 1)$

Initialize: $S^2_{0,i}, G_{0,i}, M_{0,i} \leftarrow 0, \eta_{0,i} \leftarrow \epsilon; i = 1, \dots, d$
for $t = 1, \dots, T$ **do**
  Receive $\boldsymbol{x}_t \in \mathbb{R}^d$
  for $i = 1, \dots, d$ **do**
    $M_{t,i} \leftarrow \max\{M_{t-1,i}, |x_{t,i}|\}$
    $w_{t,i} = \frac{\operatorname{sgn}(\theta_{t,i}) \min\{|\theta_{t,i}|, 1\}}{2\sqrt{S^2_{t-1,i} + M^2_{t,i}}} \eta_{t-1,i}$
      where $\theta_{t,i} = \frac{G_{t-1,i}}{\sqrt{S^2_{t-1,i} + M^2_{t,i}}}$
  Predict with $\widehat{y}_t = \boldsymbol{x}_t^\top \boldsymbol{w}_{t,i}$, receive loss $\ell_t(\widehat{y}_t)$ and compute $g_t = \partial_{\widehat{y}_t} \ell_t(\widehat{y}_t)$
  for $i = 1, \dots, d$ **do**
    $G_{t,i} \leftarrow G_{t-1,i} - g_t x_{t,i}$
    $S^2_{t,i} \leftarrow S^2_{t-1,i} + (g_t x_{t,i})^2$
    $\eta_{t,i} \leftarrow \eta_{t-1,i} - g_t x_{t,i} w_{t,i}$

The algorithm is based on a potential function $\psi_{t,i}(x) = e^{\frac{1}{2} h(x/\hat{S}_{t,i})}$, where:

$$h(y) = \begin{cases} \frac{1}{2} y^2 & \text{for } |y| \le 1, \\ |y| - \frac{1}{2} & \text{for } |y| > 1. \end{cases}$$

Function $h(y)$ interpolates between between the quadratic (for $|y| \le 1$) and absolute value (otherwise). It is easy to check that $h(y) \ge |y| - \frac{1}{2}$ for all y.

By the definition, $\eta_{t,i} = \epsilon - \sum_{j \le t} g_t x_{t,i} w_{t,i}$ is (up to ϵ) the cumulative negative loss of the algorithm ("reward"). The weights are chosen in order to guarantee the relative increase in the reward lower-bounded by the relative increase in the potential:

$$\frac{\eta_{t,i}}{\eta_{t-1,i}} \ge \frac{\psi_{t,i}(G_{t,i})}{\psi_{t-1,i}(G_{t-1,i})} e^{-\delta_{t,i}},$$

where $\delta_{t,i}$ is an overhead which can be controlled. This inequality is a counterpart of (5) from the analysis of ScInOL$_1$. However, contrary to (5), it is not incorporated into the algorithm, but only used in the regret analysis. Taking the product over trials $t = 1, \dots, T$ and using $\psi_{0,i}(G_{0,i}) = 1$ gives $\eta_T \ge \epsilon \psi_{T,i}(G_{T,i}) e^{-\Delta_T}$, where $\Delta_{T,i} = \sum_t \delta_{t,i}$. Using the definition of $\eta_{T,i}$, this translates to:

$$\begin{aligned} \sum_t g_t x_{t,i} w_{t,i} \le& \epsilon - \epsilon \psi_{T,i}(G_{T,i}) e^{-\Delta_{T,i}} \\ \le& \epsilon - \epsilon e^{-\Delta_{T,i} - \frac{1}{4}} e^{|G_{T,i}|/(2\hat{S}_{T,i})}, \end{aligned}$$

where we used $h(y) \ge |y| - \frac{1}{2}$. Denote the function on the r.h.s. by $f(x) = \epsilon e^{-\Delta_{T,i} - \frac{1}{4}} e^{|x|/(2\hat{S}_{T,i})}$. Using convexity of $f(x)$, we can express it by means of its Fenchel conjugate, $f(G_{T,i}) = \sup_u \{G_{T,i} u - f^*(u)\}$, for which we have the following bound (Orabona, 2013):

$$f^*(u) \le 2|u| \hat{S}_{T,i} \left(\ln \left(2\epsilon^{-1} |u| \hat{S}_{T,i} e^{\frac{1}{4} + \Delta_{T,i}} \right) - 1 \right).$$

Unfortunately, it turns out that $\Delta_{T,i}$ can be $\Omega(T)$ in the worst case, which makes the bound linear in T. We can, however, bound $\Delta_{T,i}$ in a data-dependent way by:

$$\Delta_{T,i} \le \ln \left(\frac{\hat{S}^2_{T,i}}{x^2_{\tau_i,i}} \right)$$

where τ_i is the first trial in which $|x_{t,i}| \ne 0$. As $\hat{S}^2_{T,i} \le (T+1) \max_t x^2_{t,i}$, the bound involves the ratio between the largest and the first non-zero input value. While being vacuous in the worst case, this quantity is likely not to be excessively large for non-adversarial data encountered in practice, and it is moreover hidden under the logarithm in the bound (a similar quantity is analyzed by Ross et al. (2013), where its magnitude is bounded with high probability for data received in a random order). Following along the steps from the previous section, we end up with the following bound:

Theorem 3.2. *For any $\boldsymbol{u} \in \mathbb{R}^d$ the regret of* ScInOL$_2$ *is upper-bounded by:*

$$R_T(\boldsymbol{u}) \le d\epsilon + \sum_{i=1}^d 2|u_i| \hat{S}_{T,i} \left(\ln(3|u_i| \hat{S}^3_{T,i} \epsilon^{-1} / x^2_{\tau_i,i}) - 1 \right),$$

where $\hat{S}_{T,i} = \sqrt{S^2_{T,i} + M^2_{T,i}}$ and $\tau_i = \min\{t\colon |x_{t,i}| \ne 0\}$.

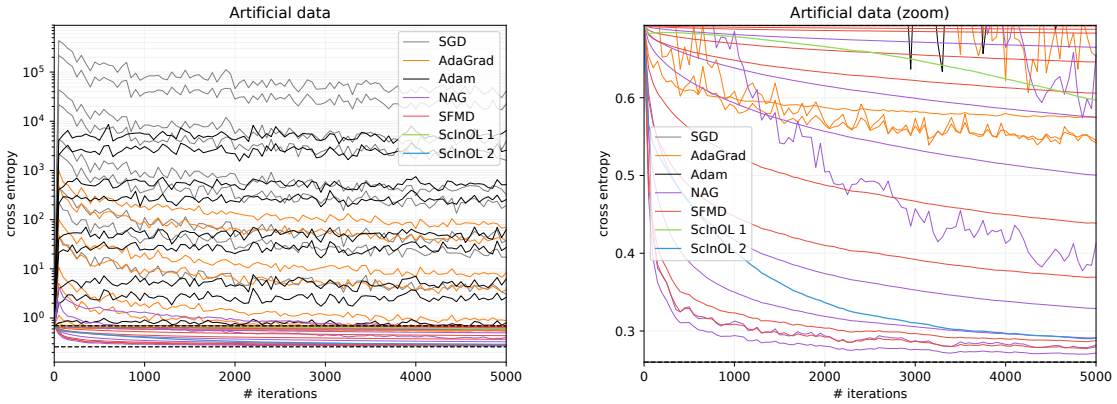
The proof of Theorem 3.2 is given in Appendix D.

4. Experiments

4.1. Illustrative Example

To empirically demonstrate a need for scale invariance we tested our algorithms against some popular adaptive variants of OGD. The tests were run on a simple artificial binary classification dataset that mildly exaggerates relative magnitudes of features (however still keeps them in reasonable ranges). The dataset contains 21 real features, values of which are drawn from normal distributions $N(0, \sigma_i)$, where $\sigma_i = 2^{i-11}$ $(i = 1, \dots, 21)$, so that the scales of features vary from 2^{-10} to 2^{10} (the ratio of the largest and the smallest scale is of order 10^6). Binary class labels where drawn from a Bernoulli distribution with $\Pr(y = 1|\boldsymbol{x}) = \operatorname{sigmoid}(\boldsymbol{x}^\top \boldsymbol{u})$ where $u_i = \pm \frac{1}{\sigma_i}$ with signs chosen uniformly at random. Note that $\boldsymbol{u}$ is set to compensate the scale of features and keep the predictions function $\boldsymbol{x}^\top \boldsymbol{u}$ to be of order of unity. We have drawn 5 000 training examples and 100 000 test examples. We repeated the experiment on 10 random training sets to decrease the variation of the results.

The algorithms were trained by minimizing the logistic loss (cross entropy loss) in an online fashion. Following similar experiments in the past papers concerning online methods



(a) Entire view of the results with logarithmic scale.

(b) Zoomed-in fragment showing best performers.

Figure 1. Average cross entropy from test set for the toy example.

(Kingma & Ba, 2014; Ross et al., 2013; Orabona & Tommasi, 2017), we report the average loss on the test set (after every 50 iterations) rather than the regret. We tested the following algorithms: OGD with learning rate decaying as $\eta_t = \eta/\sqrt{t}$ (called SGD here from *stochastic gradient descent*), AdaGrad (Duchi et al., 2011), Adam (Kingma & Ba, 2014), two scale-invariant algorithms from past work: NAG (Normalized Adaptive Gradient) (Ross et al., 2013) and Scale-free Mirror Descent by Orabona et al. (2015) (SFMD), and algorithms from this work. All algorithms except ours have a learning rate parameter, which in each case was set to values from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$ (results concerning all learning rates were reported). We implemented our algorithms in Tensorflow and used existing implementations whenever it was possible.

Figure 1 shows average cross entropy measured on test set as a function of the number of iterations. Lines of the same color show results for the same algorithm but with different learning rates. Figure 1a uses logarithmic scale for y axis: note the extreme values of the loss for most of the non-invariant algorithms. In fact, the two black dashed lines mark the loss achieved by the best possible model $\boldsymbol{u}$ (lower line) and a model with zero weight vector (upper line), so that every method above the upper dashed line does something worse than such a trivial baseline. Figure 1b shows only the fragment between dashed lines using linear scale for y axis.

The results clearly show that algorithms which are not invariant to feature scales (SGD, Adam, and AdaGrad) are unable to achieve any reasonable result for any choice of the learning rate (most of the time performing much worse than the zero vector). This is because a single learning rate is unable to compensate all feature scales at the same time. The scale invariant algorithms, NAG and SFMD, perform much better (achieving the best overall results), but their behavior still depends on the learning rate tuning. Among our algorithms, ScInOL$_1$ slowly decreases its loss moving away from the initial zero solution, but it is clearly too slow in this problem. On the other hand, ScInOL$_2$ was able to achieve descent results without any tuning at all.

4.2. Linear Classification

To further check empirical performance of our algorithms we tested them on some popular real-life benchmark datasets. We chose 5 datasets with varying levels of feature scale variance from UCI repository (Dheeru & Karra Taniskidou, 2017) (Covertype, Census, Shuttle, Bank, Madelon) and a popular benchmark dataset MNIST (LeCun & Cortes, 2010). For all datasets, categorical features were one-hot-encoded into multiple features and missing values were replaced by dedicated substitute features. Datasets that do not provide separate testing sets were split randomly into training/test sets ($2/1$ ratio). Short summary of datasets can be found in Appendix E.

Algorithms were trained by minimizing the cross entropy loss in an online fashion. Some of the data sets concern multiclass classification, which is beyond the framework considered here, as it would require *multivariate* prediction $\widehat{\boldsymbol{y}} = (\widehat{y}_1, \dots, \widehat{y}_K)$ for each of K classes, but it is straightforward to extend our setup to such multivariate case (details are given in Appendix G). To gain more insight into the long-term behavior of the algorithms, we trained all algorithms for multiple epochs. Each epoch consisted of running through the entire training set (shuffled) and testing average cross entropy and accuracy on the test set. Each algorithm was run 10 times for stability. We compared our algorithms with the following methods: SGD (with $\eta_t \sim 1/\sqrt{t}$), AdaGrad, Adam, NAG, CoCoB (Orabona & Tommasi, 2017)

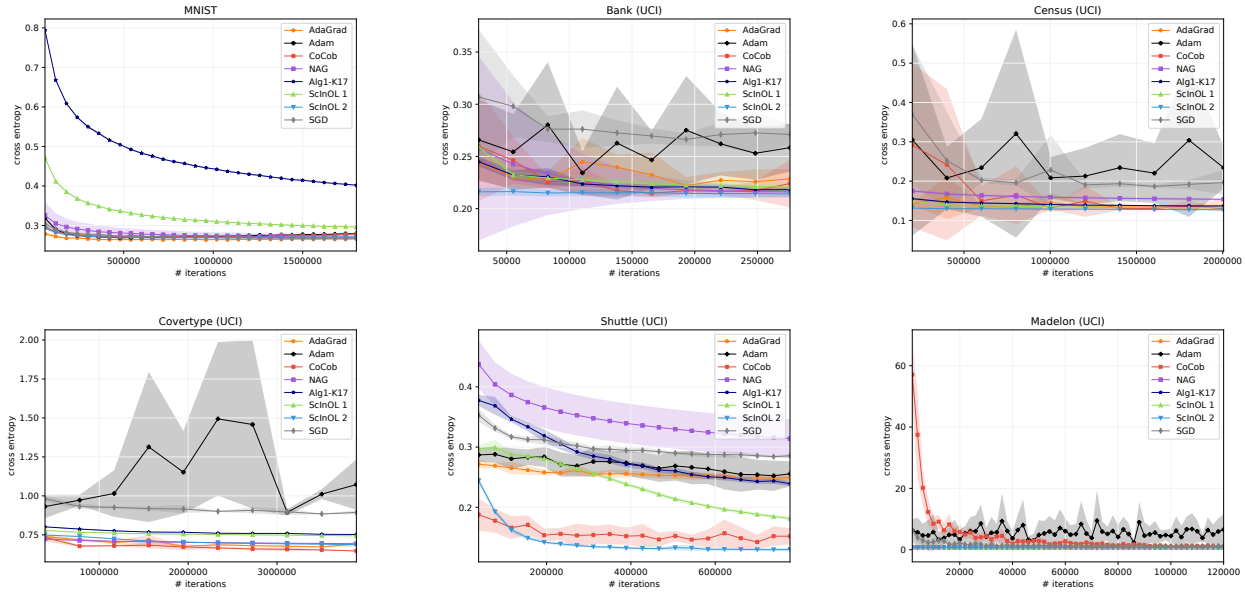


Figure 2. Mean test set cross entropy loss. Average values from 10 runs are depicted, shadows show $\pm$ standard deviation.

(an adaptive parameter-free algorithm; we used its Tensorflow implementation), and Algorithm 1 (coordinate-wise scale invariant method) by Kotłowski (2017) (which we call Alg1-K17). The algorithms using hand-picked learning rates (SGD, AdaGrad, Adam, NAG) were run with values from $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0\}$ (all other parameters were kept default), and only the best test set results were reported (note that this biases the results in favour of these algorithms).

Figure 2 shows mean (test set) cross entropy loss (classification accuracy, given in Appendix F, leads to essentially the same conclusions); shaded areas around each curve depicts $\pm$ one standard deviation (over different runs). All graphs start with the error measured after the first epoch for better readability. The most noticeable fact in the plots is comparatively high variance (between different runs) of SGD, AdaGrad, Adam and NAG, i.e. the approaches with tunable learning rate. They also often performed worse than the remaining algorithms. Among our methods, ScInOL$_2$ turned out to perform better than ScInOL$_1$ in every case, due to its more aggressive updates. Note, however, that both algorithms are surprisingly stable, exhibiting very small variance in their performance across different runs. The best performance was most of the time achieved by either ScInOL$_2$ or CoCoB. Alg1-K17 was often converging somewhat slower (which is most pronounced for MNIST data), which we believe is due to its very conservative update policy.

5. Conclusions and Future Work

We proposed two online algorithms which behavior is invariant under arbitrary rescaling of individual features. The algorithms do not require any prior knowledge on the scale of the instances or the comparator and achieve, without any parameter tuning, regret bounds which match (up to a logarithmic factor) the regret bound of Online Gradient Descent with optimally tuned separate learning rates per dimension. The algorithms run in $O(d)$ per trial, which is comparable to the runtime of vanilla OGD.

The framework considered in this paper concerns well-understood and relatively simple linear models with convex objectives. It would be interesting to evaluate the importance of scale-invariance for deep learning methods, comprised of multiple layers connected by non-linear activation functions. As scale-invariance leads to well-conditioned algorithms, we believe that it could not only avoid the need for prior normalization of the inputs to the network, but it would also make the algorithm be independent of the scale of the inputs fed forward to the next layers. A scale-invariant update for neural nets might be robust against the "internal covariance shift" phenomenon (Ioffe & Szegedy, 2015) and avoid the need for batch normalization.

Finally, the potential functions we use to analyze our updates seems closely related to the potential function of EGU$^{\pm}$ (Kivinen & Warmuth, 1997). It may be that our tuned online updates are simply approximation of (a version of) EGU$^{\pm}$ and this needs further investigation.

Acknowledgements

M. Kempka and W. Kotłowski were supported by the Polish National Science Centre under grant No. 2016/22/E/ST6/00299. Part of this work was done while M. K. Warmuth was at UC Santa Cruz, supported by NSF grant IIS-1619271.

References

Azoury, K. S. and Warmuth, M. K. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

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

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge University Press, 2006.

Cesa-Bianchi, N., Long, P., and Warmuth, M. K. Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent. *IEEE Transactions on Neural Networks*, 7(2):604–619, 1996.

Cutkosky, A. and Boahen, K. A. Online learning without prior information. In *Conference on Learning Theory (COLT)*, pp. 643–677, 2017.

Cutkosky, A. and Orabona, F. Black-box reductions for parameter-free online learning in banach spaces. In *Conference on Learning Theory (COLT)*, pp. 1493–1529, 2018.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Duchi, J. C., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Hazan, E. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3–4):157–325, 2015.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pp. 448–456, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Kivinen, J. and Warmuth, M. K. Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput.*, 132(1):1–63, 1997.

Koren, T. and Livni, R. Affine-invariant online optimization and the low-rank experts problem. In *Advances in Neural Information Processing Systems 30*, pp. 4747–4755. Curran Associates, Inc., 2017.

Kotłowski, W. Scale-invariant unconstrained online learning. In *Proceeding of the 28th International Conference on Algorithmic Learning Theory (ALT 2016)*, volume 76 of *Proceedings of Machine Learning Research*, pp. 412–433. PMLR, 2017.

LeCun, Y. and Cortes, C. MNIST handwritten digit database, 2010. URL http://yann.lecun.com/exdb/mnist/.

Littlestone, N., Long, P. M., and Warmuth, M. K. On-line learning of linear functions. *ACM Symposium on Theory of Computing (STOC)*, pp. 465–475, 1991.

Luo, H., Agarwal, A., Cesa-Bianchi, N., and Langford, J. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems (NIPS) 29*, 2016.

McMahan, H. B. and Abernethy, J. Minimax optimal algorithms for unconstrained linear optimization. In *Advances in Neural Information Processing Systems (NIPS) 26*, pp. 2724–2732, 2013.

McMahan, H. B. and Orabona, F. Unconstrained online linear learning in Hilbert spaces: Minimax algorithms and normal approximation. In *Proc. of the 27th Conference on Learning Theory (COLT)*, pp. 1020–1039, 2014.

Orabona, F. Dimension-free exponentiated gradient. In *Advances in Neural Information Processing Systems (NIPS) 26*, pp. 1806–1814, 2013.

Orabona, F. Simultaneous model selection and optimization through parameter-free stochastic learning. In *Advances in Neural Information Processing Systems (NIPS) 27*, pp. 1116–1124, 2014.

Orabona, F. and Pál, D. Scale-free algorithms for online linear optimization. In *Algorithmic Learning Theory (ALT)*, pp. 287–301, 2015.

Orabona, F. and Pál, D. Coin betting and parameter-free online learning. In *Neural Information Processing Systems (NIPS)*, 2016.

Orabona, F. and Tommasi, T. Training deep networks without learning rates through coin betting. In *Advances in Neural Information Processing Systems (NIPS) 30*, pp. 2157–2167, 2017.

Orabona, F., Crammer, K., and Cesa-Bianchi, N. A generalized online mirror descent with applications to classification and regression. *Machine Learning*, 99(3):411–435, 2015.

Ross, S., Mineiro, P., and Langford, J. Normalized online learning. In *Proc. of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 537–545, 2013.

Shalev-Shwartz, S. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

Streeter, M. and McMahan, H. B. No-regret algorithms for unconstrained online convex optimization. In *Advances in Neural Information Processing Systems (NIPS) 25*, pp. 2402–2410, 2012.

Vovk, V. Competitive on-line statistics. *International Statistical Review*, 69(213-248), 2001.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, pp. 928–936, 2003.