# Matrix-Free Preconditioning in Online Learning

**Ashok Cutkosky** [1] **Tamas Sarlos** [1]

## Abstract

We provide an online convex optimization algorithm with regret that interpolates between the regret of an algorithm using an optimal preconditioning matrix and one using a diagonal preconditioning matrix. Our regret bound is never worse than that obtained by diagonal preconditioning, and in certain setting even surpasses that of algorithms with full-matrix preconditioning. Importantly, our algorithm runs in the same time and space complexity as online gradient descent. Along the way we incorporate new techniques that mildly streamline and improve logarithmic factors in prior regret analyses. We conclude by benchmarking our algorithm on synthetic data and deep learning tasks.

## 1. Online Learning

This paper considers the online linear optimization (OLO) problem. An OLO algorithm chooses output vectors $w_t \in \mathbb{R}^d$ in response to linear losses $\ell_t(w) = g_t \cdot w$ for some $g_t \in \mathbb{R}^d$. Performance is measured by the *regret* (Shalev-Shwartz et al., 2012; Zinkevich, 2003):

$$R_T(\mathring{w}) = \sum_{t=1}^T \ell_t(w_t) - \ell_t(\mathring{w}) = \sum_{t=1}^T g_t \cdot (w_t - \mathring{w})$$

OLO algorithms are important because they also solve solve online *convex* optimization problems, in which the losses $\ell_t$ need be only convex by virtue of taking $g_t$ to be a gradient (or subgradient) of the $t$th convex loss. Even better, these algorithms also solve *stochastic* convex optimization problems by setting $\ell_t$ to be the $t$th minibatch loss and $\mathring{w}$ to be the global minimizer (Cesa-Bianchi et al., 2004). Due to both the simplicity of the linear setting and the power of the resulting algorithms, OLO has become a successful and popular framework for designing and analyzing many of the algorithms used to train machine learning models today.

[1]Google Research, California, USA. Correspondence to: Ashok Cutkosky <cutkosky@google.com>.

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

Our goal is to obtain *adaptive* regret bounds so that $R_T(\mathring{w})$ may be much smaller in easier problems while still maintaining optimal worst-case guarantees. One relevant prior result is the "diagonal preconditioner" approach of Adagrad-style algorithms (Duchi et al., 2011; McMahan & Streeter, 2010):

$$R_T(\mathring{w}) \le \sum_{i=1}^d \|\mathring{w}\|_\infty \sqrt{\sum_{t=1}^T g_{t,i}^2} \tag{1}$$

where $g_{t,i}$ indicates the $i$th coordinate of $g_t$. This bound can be achieved via gradient descent with learning rates properly tuned to the value of $\|\mathring{w}\|_\infty$, and algorithms of this flavor have found much use in practice. Similar regret bounds that do not require tuning to the value of $\|\mathring{w}\|_\infty$ can be obtained by making use of a Lipschitz assumption $\|g_t\| \le G$, leading to a bound of the form (Cutkosky & Orabona, 2018):

$$R_T(\mathring{w}) \le O\left(\epsilon + \sum_{i=1}^d |\mathring{w}_i| \sqrt{\log(\tfrac{d|\mathring{w}_i|T}{\epsilon}) \sum_{t=1}^T g_{t,i}^2}\right) \tag{2}$$

where $\epsilon$ is a free parameter representing the "regret at the origin". The extra logarithmic factor is an unavoidable penalty for this extra adaptivity (McMahan & Streeter, 2012). This bound has the advantage that by Cauchy-Schwarz it is at most a logarithmic factor away from $\|\mathring{w}\|_2 \sqrt{\sum_{t=1}^T \|g_t\|_2^2}$, while the diagonal Adagrad bound may be a factor of $\sqrt{d}$ worse due to the $\|\mathring{w}\|_\infty$ instead of $\|\mathring{w}\|_2$.

Another type of bound is the "full-matrix Adagrad" bound (Duchi et al., 2011)

$$R_T(\mathring{w}) \le \|\mathring{w}\|_2 \mathrm{Tr}\left(\sqrt{\sum_{t=1}^T g_t g_t^T}\right) \tag{3}$$

or the more recent improved full-matrix bounds of (Cutkosky & Orabona, 2018; Koren & Livni, 2017):

$$R_T(\mathring{w}) \le \sqrt{d \sum_{t=1}^T \langle g_t, \mathring{w}\rangle^2} \tag{4}$$

The above bound (4) may be much better than the diagonal bound, but unfortunately the algorithms involve manipulating a $d \times d$ matrix (often called a "preconditioning matrix") and so require $O(d^2)$ time per update. Our goal is to design an algorithm that maintains $O(d)$ time per update, but still manages to smoothly interpolate between the diagonal bound (2) and full-matrix bounds (4).

Efficiently approximating the performance of full-matrix algorithms is an active area of research. Prior approaches include approximations based on sketching, low-rank approximations, and exploiting some assumed structure in the gradients (Luo et al., 2016; Gupta et al., 2018; Agarwal et al., 2018; Gonen & Shalev-Shwartz, 2015; Martens & Grosse, 2015). The typical approach is to trade off computational complexity for approximation quality by maintaining some kind of lossy compressed representation of the preconditioning matrix. The properties of these tradeoffs vary: for some strategies one may obtain worse-regret than a non-full matrix algorithm (or even linear regret) if the data is particularly adversarial, while for others one may be unable to see nontrivial gains without significant complexity penalties. Our techniques are rather different, and so we make no complexity tradeoffs, never suffer worse regret than diagonal algorithms, and yet still obtain full-matrix bounds in favorable situations.

This paper is organized as follows: In Section 2 we give some background in online learning analysis techniques that we will be using. In Sections 3-5 we state and analyze our algorithm that can interpolate between full-matrix and diagonal regret bounds efficiently. In Section 6 we provide an empirical evaluation of our algorithm.

## 2. Betting Algorithms

A recent technique for designing online algorithms is via the wealth-regret duality approach (McMahan & Orabona, 2014) and betting algorithms (Orabona & Pál, 2016). In betting algorithms, one keeps track of the "wealth":

$$\text{Wealth}_T = \epsilon - \sum_{t=1}^{T} g_t \cdot w_t$$

where $\epsilon > 0$ is some user-defined hyperparameter. The goal is to make the wealth as big as possible, because

$$R_T(\mathring{w}) = \epsilon - \text{Wealth}_T - \sum_{t=1}^{T} g_t \cdot \mathring{w}$$

and in some sense the wealth is the only part of the above expression that the algorithm actually has control over.

Specifically, we want to obtain a statement like:

$$\text{Wealth}_T \geq f\left(-\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}\right)$$

for some function $f$, which exists only for analysis purposes here. Given this inequality, we can write:

$$\begin{aligned} R_T(\mathring{w}) &= \epsilon - \sum_{T=1}^{T} g_t \cdot \mathring{w} - \text{Wealth}_T \\ &\leq \epsilon - \sum_{T=1}^{T} g_t \cdot \mathring{w} - f\left(-\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}\right) \\ &\leq \epsilon + \sup_{X \in \mathbb{R}} X\|\mathring{w}\| - f(X) \\ &= \epsilon + f^\star(\|\mathring{w}\|) \end{aligned}$$

where $f^\star$ is the Fenchel conjugate, defined by $f^\star(u) = \sup_x xu - f(x)$. Formally, we have:

**Lemma 1.** *If Wealth*$_T \geq f\left(-\sum_{t=1}^{T} \frac{g_t \cdot \mathring{w}}{\|\mathring{w}\|}\right)$ *for some arbitrary norm* $\|\cdot\|$ *and function* $f$*, then* $R_T(\mathring{w}) \leq \epsilon + f^\star(\|\mathring{w}\|)$.

One way to increase the wealth is to view the vectors $w_t$ as some kind of "bet" and $g_t$ as some kind of outcome (e.g. imagine that $w_t$ is a portfolio and $-g_t$ is a vector of returns). Then the amount of "money" you win at time $t$ is $-g_t \cdot w_t$ and so Wealth$_T$ is the total amount of money you have at time $T$, assuming you started out with $\epsilon$ units of currency.

In order to leverage this metaphor, we make a critical assumption: $\|g_t\|_\star \leq 1$ for all $t$. Here $\|\cdot\|_\star$ is the dual norm, $\|g\|_\star = \sup_{\|w\|\leq 1} g \cdot w$ (e.g. when $\|\cdot\|$ is the 2-norm, $\|\cdot\|_\star$ is also the 2-norm, and when $\|\cdot\|$ is the infinity-norm, $\|\cdot\|_\star$ is the 1-norm). There is nothing special about 1 here; we may choose any constant, but use 1 for simplicity.

Under this assumption, guaranteeing $R_T(0) \leq \epsilon$ is equivalent to never going into debt (i.e. Wealth$_T < 0$). We assure this by never betting more than we have: $\|w_t\| \leq \text{Wealth}_{t-1}$. In fact, in order to simplify subsequent calculations, we will ask for a somewhat stronger restriction:

$$\|w_t\| < \frac{1}{2}\text{Wealth}_{t-1} \tag{5}$$

Given (5), we can also write

$$w_t = v_t \text{Wealth}_{t-1}$$

where $v_t$ is a vector with $\|v_t\| < 1/2$, which we call the "betting fraction". $v_t$ is a kind of "learning rate" analogue. However, in the $d$-dimensional setting $v_t$ is only a $d$-dimensional vector, while previous full-matrix algorithms use a learning rate analogue that is a $d \times d$ matrix.

### 2.1. Constant $v_t$

To understand the potential of this approach, consider the case of a fixed betting fraction $v_t = \mathring{v}$. Using the inequality $\log(1-x) \geq -x - x^2$ for all $x \leq 1/2$, we proceed:

$$\text{Wealth}_T = \epsilon \prod_{t=1}^{T}(1 - g_t \cdot \mathring{v})$$

$$\log(\text{Wealth}_T) = \log(\epsilon) + \sum_{t=1}^{T} \log(1 - g_t \cdot \mathring{v}) \tag{6}$$

$$\geq \log(\epsilon) + \sum_{t=1}^{T} -g_t \cdot \mathring{v} - (g_t \cdot \mathring{v})^2 \tag{7}$$

Now if we set

$$\mathring{v} = -\frac{\mathring{w}}{\|\mathring{w}\|} \frac{\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}}{2\left|\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}\right| + 2\sum_{t=1}^{T}(g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|})^2}$$

we obtain:

$$\text{Wealth}_T \geq \epsilon \exp\left[\frac{\left(\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}\right)^2}{4\left|\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}\right| + 4\sum_{t=1}^{T}(g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|})^2}\right]$$
$$= f\left(-\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}\right)$$

Where $f(x) = \epsilon \exp\left[\frac{x^2}{4|x| + 4\sum_{t=1}^{T}(g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|})^2}\right]$. Finally, bound $f^\star$ by Lemma 19 of (Cutkosky & Orabona, 2018):

$$R_T(\mathring{w}) \leq \epsilon + f^\star(\|\mathring{w}\|) \leq \tilde{O}\left[\epsilon + \sqrt{\sum_{t=1}^{T}(g_t \cdot \mathring{w})^2}\right]$$

This is actually a factor of up to $\sqrt{d}$ better than the full-matrix guarantee (4) and, more importantly, there are no matrices in this algorithm! Instead, the role of the preconditioner is played by the vector $\mathring{v}$, which corresponds to a kind of "optimal direction".

### 2.2. Varying $v_t$

Now that we know that there exists a good fixed betting fraction $v$ given oracle tuning, we turn to the problem of using varying $v_t$. To do this we use the reduction developed by Cutkosky & Orabona (2018) for recasting the problem of choosing $v_t$ as itself an online learning problem. The first step is to calculate the wealth with changing $v_t$:

$$\log(\text{Wealth}_T) = \log(\epsilon) + \sum_{t=1}^{T} \log(1 - g_t \cdot v_t) \tag{8}$$

Next, denote the wealth of the algorithm that uses a fixed fraction $\mathring{v}$ as $\text{Wealth}_T(\mathring{v})$ and then subtract (8) from (6):

$$\log(\text{Wealth}_T(\mathring{v})) - \log(\text{Wealth}_T)$$
$$= \sum_{t=1}^{T} \log(1 - g_t \cdot \mathring{v}) - \log(1 - g_t \cdot v_t)$$
$$= \sum_{t=1}^{T} \ell_t(v_t) - \ell_t(\mathring{v})$$

where we define $\ell_t(x) = -\log(1 - g_t \cdot x)$. Notice that $\ell_t$ is convex, so we can try to find $\mathring{v}$ by using an online convex optimization algorithm that outputs $v_t$ in response to the loss $\ell_t$. Let $R_T^v$ be the regret of this algorithm. Then by definition of regret, for any $\mathring{v}$:

$$\log(\text{Wealth}_T(\mathring{v})) - \log(\text{Wealth}_T) = R_T^v(\mathring{v})$$

Combining the above with inequality (7) we have

$$\log(\text{Wealth}_T) = \log(\text{Wealth}_T(\mathring{v})) - R_T^v(\mathring{v})$$
$$= \log(\epsilon) + \sum_{t=1}^{T} -g_t \cdot \mathring{v} - (g_t \cdot \mathring{v})^2 - R_T^v(\mathring{v}) \tag{9}$$

So now need to find a $\mathring{v}$ that maximizes this expression.

Our analysis diverges from prior work at this point. Previously, (Cutkosky & Orabona, 2018) observed that $\ell_t$ is exp-concave, and so by using the Online Newton Step (Hazan et al., 2007) algorithm one can make $R_T^v(\mathring{v}) = O(\log(T))$ and obtain regret

$$R_T(\mathring{w}) \leq O\left(\|\mathring{w}\|\sqrt{\log(\tfrac{\|\mathring{w}\|T^{4.5}}{\epsilon})\sum_{t=1}^{T}\|g_t\|^2}\right) \tag{10}$$

Instead, we take a different strategy by using *recursion*. The idea is simple: we can apply the exact same reduction we have just outlined to design an "inner" coin-betting strategy for choosing $v_t$ and minimizing $R_T^v$. The major subtlety that needs to be addressed is the restriction $\|v_t\| \leq 1/2$. Fortunately, (Cutkosky & Orabona, 2018) also provides a black-box reduction that converts any unconstrained optimization algorithm into a constrained algorithm without modifying the regret bound, and so we can essentially ignore the constraint on $v_t$ in our analysis.

## 3. Recursive Betting Algorithm

The key advantage of using a recursive strategy to choose $v_t$ is that the regret $R_T^v(\mathring{v})$ may depend strongly on $\|\mathring{v}\|$. Since in many cases $\|\mathring{v}\|$ is small, this results in better overall performance than if we were to directly apply the Online Newton Step algorithm. We formalize this strategy and intuition in Algorithm 1 and Theorem 1.

**Algorithm 1** Recursive Optimizer

```
procedure RECURSIVEOPTIMIZER(ε)
    Wealth_0 ← ε.
    Initialize INNEROPTIMIZER.
    for t = 1 ... T do
        Let v_t be the tth output of INNEROPTIMIZER.
        w_t ← Wealth_{t-1} v_t.
        Output w_t, receive g_t.
        Wealth_t ← Wealth_{t-1} − g_t · w_t.
        z_t ← g_t / (1 − g_t · v_t) = d/dv_t − log(1 − g_t · v_t).
        Send z_t as t^th gradient to INNEROPTIMIZER.
    end for
end procedure
```

**Theorem 1.** *Suppose $\|g_t\|_\star \le 1$ for some norm $\|\cdot\|$ for all $t$. Further suppose that* INNEROPTIMIZER *satisfies $\|v_t\| \le 1/2$ and guarantees regret nearly linear in $\|\mathring{v}\|$:*

$$R_T^v(\mathring{v}) = \sum_{t=1}^{T} z_t \cdot v_t - z_t \cdot \mathring{v} \le \epsilon + \|\mathring{v}\| G_T(\mathring{v}/\|\mathring{v}\|)$$

*for some function $G_T(\mathring{v}/\|\mathring{v}\|)$ for any $\mathring{v}$ with $\|\mathring{v}\| \le 1/2$. Then if $-\sum_{t=1}^T g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|} \ge 2G_T(\mathring{w}/\|\mathring{w}\|)$, Algorithm 1 obtains*

$$R_T(\mathring{w}) \le \tilde{O}\left(\epsilon + \sqrt{\sum_{t=1}^{T}(g_t \cdot \mathring{w})^2}\right)$$

*and otherwise*

$$R_T(\mathring{w}) \le \epsilon + 2\|\mathring{w}\| G_T(\mathring{w}/\|\mathring{w}\|)$$

Let us unpack the condition $-\sum_{t=1}^T g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|} \ge 2G_T(\mathring{w}/\|\mathring{w}\|)$. First we consider the LHS. Observe that $-\sum_{t=1}^T \frac{\mathring{w}}{\|\mathring{w}\|} \cdot g_t$ is the regret at $\mathring{w}/\|\mathring{w}\|$ of an algorithm that always predicts 0. In a classic adversarial problem we should expect this value to grow as $\Omega(T)$. Even in the case that each $g_t$ is an i.i.d. mean-zero random variable, we should expect growth of at least $\Omega(\sqrt{T})$. For the RHS, observe that so long as INNEROPTIMIZER obtains the optimal $T$-dependence in its regret bound, we should expect $G_T = \tilde{O}(\sqrt{T})$ - for example the algorithm of (Cutkosky & Orabona, 2018) obtains $G_T(\mathring{v}/\|\mathring{v}\|) = O\left(\sqrt{\sum_{t=1}^T \|g_t\|_\star^2 \log(T^{4.5}/\epsilon)}\right)$ for any $\|\mathring{v}\| \le 1/2$. Therefore the condition $-\sum_{t=1}^T g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|} \ge 2G_T(\mathring{w}/\|\mathring{w}\|)$ can be viewed as saying that the $g_t$ in some sense violate standard concentration inequalities and so are clearly not mean-zero random variables: intuitively, there is some amount of signal in the gradients.

As a simple concrete example, suppose the $g_t$ are i.i.d. random vectors with covariance $\Sigma$ and mean $-\sqrt{\epsilon}x$, where $x$ is the eigenvector of $\Sigma$ with smallest eigenvalue. Then $\sum_{t=1}^T g_t \cdot x$ will grow as $\Theta(T\sqrt{\epsilon})$, and so for sufficiently large $T$ we will obtain the full-matrix regret bound where $R_T(x)$ grows with $\sum_{t=1}^T (g_t \cdot x)^2$. This has expectation $x^T \Sigma x T + \epsilon T = (\lambda_d + \epsilon)T$, where $\lambda_d$ is the smallest eigenvalue of $\Sigma$. In contrast, a standard regret bound may depend on $\sum_{t=1}^T \|g_t\|_2^2$. This has expectation $\mathrm{Trace}(\Sigma)T + \epsilon T$, which is a factor of $d$ larger for small $\epsilon$, and even more if $\Sigma$ is poorly conditioned.

Next, let us consider the second case in which the regret bound is $O(\epsilon + \|\mathring{w}\| G_T)$. This bound is also actually a subtle improvement on prior guarantees. For example, if INNEROPTIMIZER guarantees regret $R_T^v(\mathring{v}) \le \epsilon + \|\mathring{v}\| \sqrt{\log(\|\mathring{v}\|T/\epsilon) \sum_{t=1}^T \|g_t\|_\star^2}$, we can use the fact that $\|\mathring{v}\| \le 1/2$ to bound $G_T$ by $\sqrt{\log(T/\epsilon)\sum_{t=1}^T \|g_t\|_\star^2}$. Thus the bound $\|\mathring{w}\| G_T$ is better than previous regret bounds like (10) due to removing the $\|\mathring{w}\|$ from inside the log.

In summary, we improve prior art in two important ways:

1. When the sum of the gradients is greater than $\tilde{\Omega}(\sqrt{T})$, we obtain the optimal full-matrix regret bound.
2. When the sum of the gradients is smaller, our regret bound grows only linearly with $\|\mathring{w}\|$, without any $\sqrt{\log(\|\mathring{w}\|)}$ factor.

Both of these improvements appear to contradict lower bounds. First, (Luo et al., 2016) suggests that the factor $\sqrt{d}$ is necessary in a full-matrix regret bound, which seems to rule out improvement 1. Second, (McMahan & Orabona, 2014; Orabona, 2013) state that a $\sqrt{\log(\|\mathring{w}\|)}$ factor is required when $\|\mathring{w}\|$ is unknown, appearing to rule out improvement 2. We are consistent with these results because of the condition $-\sum_{t=1}^T g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|} \ge 2G_T(\mathring{w}/\|\mathring{w}\|)$. Both lower bounds use $g_t$ whose coordinates are random $\pm 1$. However, the bound of (Luo et al., 2016) involves a "typical sequence", which concentrates appropriately about zero and does not satisfy the condition to have our improved full-matrix bound. In contrast, the bounds of (McMahan & Orabona, 2014; Orabona, 2013) are stated for 1-dimensional problems and rely on *anti-concentration*, so that the adversarial sequence is very atypical and does satisfy the condition, yielding our full-matrix bound that does include the log factor.

In Section 4 we propose a diagonal algorithm for use as INNEROPTIMIZER. This will enable Algorithm 1 to interpolate between a diagonal regret bound and the full-matrix guarantee. At first glance, this phenomenon is somewhat curious: how can an algorithm that keeps only per-coordinate state manage to adapt to the covariance between pairs of coordinates? The answer lies in the gradients supplied to the INNEROPTIMIZER: $\frac{g_t}{1-g_t\cdot v_t}$. The denominator of this expression actually contains information from all coordinates, and so even when INNEROPTIMIZER is a diagonal algorithm it still has access to interactions between coordinates.

Now we sketch a proof of Theorem 1. We will drop constants, logs and $\epsilon$ and leave full details to Appendix B.

*Proof Sketch of Theorem 1.* We start from (9) and use our assumption on the regret bound of INNEROPTIMIZER:

$$\log(\text{Wealth}_T) \geq \sum_{t=1}^{T} -g_t \cdot \mathring{v} - (g_t \cdot \mathring{v})^2 - \|\mathring{v}\| G_T(\mathring{v}/\|\mathring{v}\|)$$

for all $\mathring{v}$. So now we choose $\mathring{v}$ to optimizes the bound.

Let us suppose that $\mathring{v}$ is of the form $\mathring{v} = x\frac{\mathring{w}}{\|\mathring{w}\|}$ for some $x$ so that $\mathring{v}/\|\mathring{v}\| = \mathring{w}/\|\mathring{w}\|$. We consider two cases: either $-\sum_{t=1}^{T} g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|} \geq 2G_T(\mathring{w}/\|\mathring{w}\|)$ or not.

**Case 1** $-\sum_{t=1}^{T} g_t \cdot \mathring{w}/\|\mathring{w}\| \geq 2G_T(\mathring{w}/\|\mathring{w}\|)$**:**

In this case we have

$$-\sum_{t=1}^{T} g_t \cdot \mathring{v} - \|\mathring{v}\| G_T(\mathring{v}/\|\mathring{v}\|) \geq -\frac{1}{2}\sum_{t=1}^{T} g_t \cdot \mathring{v}$$

Therefore we have

$$\log(\text{Wealth}_T) \geq \sum_{t=1}^{T} -\frac{1}{2} g_t \cdot \mathring{v} - (g_t \cdot \mathring{v})^2$$

So now using essentially the same argument as in the fixed $v$ case, we end up with a full-matrix regret bound:

$$R_T(\mathring{w}) = \tilde{O}\left(\epsilon + \sqrt{\sum_{t=1}^{T} (g_t \cdot \mathring{w})^2}\right)$$

**Case 2** $-\sum_{t=1}^{T} g_t \cdot \mathring{w}/\|\mathring{w}\| < 2G_T(\mathring{w}/\|\mathring{w}\|)$

In this case, observe that since we guarantee $\text{Wealth}_T > 0$ no matter what strategy is used to pick $v_t$, we have

$$\begin{aligned} R_T(\mathring{w}) &= \epsilon - \text{Wealth}_T - \sum_{t=1}^{T} g_t \cdot \mathring{w} \\ &\leq \epsilon + 2\|\mathring{w}\| G_T(\mathring{w}/\|\mathring{w}\|) \end{aligned}$$

And so we are done. □

## 4. Diagonal INNEROPTIMIZER

As a specific example of an algorithm that can be used as INNEROPTIMIZER, we provide Algorithm 2. This algorithm will achieve a regret bound similar to (2). Here we use $\text{clip}(x, a, b)$ to indicate truncating $x$ to the interval $[a, b]$. Algorithm 2 works by simply applying a separate 1-dimensional optimizer on each coordinate of the problem. Each 1-dimensional optimizer is itself a coin-betting algorithm that uses Follow-the-Regularized leader (Hazan et al., 2016) to choose the betting fractions $v_t$. There are also two important modifications at lines 7 and 12-15 that implement the unconstrained-to-constrained reduction.

**Algorithm 2** Diagonal Betting Algorithm

1: **procedure** DIAGOPTIMIZER($\epsilon$, $\eta$)
2: $\quad$ $\text{Wealth}_{0,i} \leftarrow \epsilon$ for $i \in \{1, \dots, d\}$.
3: $\quad$ $A_{0,i} \leftarrow 5$ and $v_{1,i} \leftarrow 0$ for $i \in \{1, \dots, d\}$.
4: $\quad$ **for** $t = 1 \dots T$ **do**
5: $\quad\quad$ **for** $i = 1 \dots d$ **do**
6: $\quad\quad\quad$ $x_{t,i} \leftarrow v_{t,i}\text{Wealth}_{t-1,i}$.
7: $\quad\quad\quad$ Set $w_{t,i} = \text{clip}(x_{t,i}, -1/2, 1/2)$.
8: $\quad\quad$ **end for**
9: $\quad\quad$ Output $w_t = (w_{t,1}, \dots, w_{t,d})$.
10: $\quad\quad$ Receive $g_t$ with $g_{t,i} \in [-1, 1]$ for all $i$.
11: $\quad\quad$ **for** $i = 1 \dots d$ **do**
12: $\quad\quad\quad$ $\tilde{g}_{t,i} \leftarrow g_{t,i}$
13: $\quad\quad\quad$ **if** $g_{t,i}(x_{t,i} - w_{t,i}) < 0$ **then**
14: $\quad\quad\quad\quad$ $\tilde{g}_{t,i} \leftarrow 0$.
15: $\quad\quad\quad$ **end if**
16: $\quad\quad\quad$ $\text{Wealth}_{t,i} \leftarrow \text{Wealth}_{t-1,i} - x_{t,i}\tilde{g}_{t,i}$.
17: $\quad\quad\quad$ $z_{t,i} \leftarrow \frac{d}{dv_{t,i}} - \log(1 - \tilde{g}_{t,i}v_{t,i}) = \frac{\tilde{g}_{t,i}}{1 - \tilde{g}_{t,i}v_{t,i}}$.
18: $\quad\quad\quad$ $A_{t,i} \leftarrow A_{t-1,i} + z_{t,i}^2$.
19: $\quad\quad\quad$ $v_{t,i} \leftarrow \text{clip}\left(\frac{-2\eta\sum_{t'=1}^{t} z_{t',i}}{A_{t,i}}, -1/2, 1/2\right)$
20: $\quad\quad$ **end for**
21: $\quad$ **end for**
22: **end procedure**

Before analyzing DIAGOPTIMIZER, we perform a second analysis of RECURSIVEOPTIMIZER that makes no restrictions on INNEROPTIMIZER. We will eventually see that Algorithm 2 is essentially an instance of RECURSIVEOPTIMIZER and so this Lemma will be key in our analysis:

**Lemma 2.** *Suppose* $\|g_t\|_\star \leq 1$ *for all* $t$. *Suppose* INNEROPTIMIZER *satisfies* $\|v_t\| \leq 1/2$ *and has regret* $R_T^v(\mathring{v})$. *Then* RECURSIVEOPTIMIZER *obtains regret*

$$\begin{aligned} R_T(\mathring{w}) \leq \inf_{c \in [0,1/2]} \epsilon &+ \frac{\|\mathring{w}\|}{c}\left(\log\left(\frac{\|\mathring{w}\|}{c\epsilon}\right) - 1\right) \\ &+ \|\mathring{w}\| c Z + \frac{\|\mathring{w}\|}{c} R_T^v\left(c\frac{\mathring{w}}{\|\mathring{w}\|}\right) \end{aligned}$$

*where* $Z = \sum_{t=1}^{T}\left(g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|}\right)^2$.

In words, we have written the regret of RECURSIVEOPTIMIZER as a kind of tradeoff between $Z$, which is proportional to the quantity inside the square root of a full-matrix bound, and the regret of the INNEROPTIMIZER. This makes it easier to compute the regret when INNEROPTIMIZER's regret bound does not satisfy the conditions of Theorem 1.

**Theorem 2.** *Suppose $\|g_t\|_\infty \le 1$ for all $t$. Then Algorithm 2 guarantees regret $R_T(\mathring{w})$ at most:*

$$d\epsilon + O\left(\sum_{i=1}^d |\mathring{w}_i| \max\left\{\sqrt{\frac{G_i}{\eta}\log\left(\frac{|\mathring{w}_i|G_i^\eta\sqrt{\frac{G_i}{\eta}}}{\epsilon}\right)},\ \log\left(\frac{|\mathring{w}_i|G_i^\eta\sqrt{G_i/\eta}}{\epsilon}\right)\right\}\right)$$

*for all $\mathring{w}$ with $\|\mathring{w}\|_\infty \le 1/2$, where $G_i = \sum_{t=1}^T g_{t,i}^2$. Further, by using $\epsilon/d$ instead of $\epsilon$ and setting $\eta = 1/2$, we can also re-write this as:*

$$R_T(\mathring{w}) \le \epsilon + \|\mathring{w}\|_\infty G_T(\mathring{w}/\|\mathring{w}\|_\infty),$$

*where*

$$G_T(x) = O\left(\sum_{i=1}^d |x_i| \max\left\{\sqrt{G_i\log\left(\frac{dZ_i}{\epsilon}\right)},\ \log\left(\frac{dG_i}{\epsilon}\right)\right\}\right)$$

Let us briefly unpack this bound. Ignoring log factors to gain intuition, the bound is $\sum_{i=1}^d |\mathring{w}_i|\sqrt{\sum_{t=1}^T g_{t,i}^2}$. Note that this improves upon the diagonal Adagrad bound (1) by virtue of depending on each $|\mathring{w}_i|$ rather than the norm $\|\mathring{w}\|_\infty$, and by Cauchy-Schwarz it is bounded by $\|\mathring{w}\|_2\sqrt{\sum_{t=1}^T \|g_t\|_2^2}$, which matches classic "dimension-free" bounds. Note however that this bound is *not* strictly dimension-free due to the $d\epsilon$ term. Even setting $\epsilon = 1/d$ will incur a $\log(d)$ penalty due to the $\log(1/\epsilon)$ factor. Most importantly, however, DIAGOPTIMIZER satisfies the conditions on INNEROPTIMIZER in Theorem 1.

Theorem 2 is also notable for its logarithmic factor, which can be made $O(\log(|\mathring{w}_i|G_i^{\eta+1/2}/\epsilon\sqrt{\eta}))$ for any $\eta$. This is an improvement over prior bounds such as (Cutkosky & Orabona, 2018) in that the power of the $O(T)$ term $G_i$ inside the logarithm is smaller However, the optimal value for this exponent is $1/2$ (McMahan & Orabona, 2014), which this bound cannot obtain. Instead, we show in Appendix C.1 that a simple doubling-trick scheme does allow us to obtain the optimal rate. To our knowledge this is the first time such a rate has been achieved: prior works achieve the optimal log factor, but have worse adaptivity to the values of $g_t$, depending on $T$ or $\sum_{t=1}^T |g_t|$ instead of $\sum_{t=1}^T g_t^2$ (McMahan & Orabona, 2014; Orabona, 2014).

*Proof Sketch of Theorem 2.* First, we observe that Algorithm 2 is running $d$ copies of a 1-dimensional optimizer. Because we have

$$R_T(\mathring{w}) = \sum_{t=1}^T g_t \cdot (w_t - \mathring{w}) = \sum_{i=1}^d \sum_{t=1}^T g_{t,i}(w_{t,i} - \mathring{w}_i)$$

we may analyze each dimension individually and then sum the regrets. So let us focus on a single 1-dimensional optimizer, and drop all $i$ subscripts for simplicity.

Next, we address the truncation of $w_t$ and modifications to $g_t$. This is a 1-dimensional specialization of the unconstrained-to-constrained reduction of (Cutkosky & Orabona, 2018). Let $g_t$ be the (original, unmodified) gradient, and let $\tilde{g}_t$ be the modified gradient (so $\tilde{g}_t = g_t$ or $\tilde{g}_t = 0$). A little calculation shows that

$$\tilde{g}_t(x_t - \mathring{w}) \ge g_t(w_t - \mathring{w})$$

for any $\mathring{w} \in [-1/2, 1/2]$. Therefore, the regret $\sum_{t=1}^T g_t(w_t - \mathring{w})$ is upper-bounded by $\sum_{t=1}^T \tilde{g}_t(x_t - \mathring{w})$. This quantity is simply the regret of an algorithm that uses gradients $\tilde{g}_t$ and outputs $x_t$. Now we interpret $x_t$ as the predictions of a coin-betting algorithm that uses betting fractions $v_t$ in response to the gradients $\tilde{g}_t$. Thus we may analyze the regret of the $x_t$ with respect to the $\tilde{g}_t$ using coin-betting machinery. To this end, observe that $A_t = 5 + \sum_{t'=1}^t z_{t'}^2$, so that

$$v_{t+1} = \operatorname*{argmin}_{v\in[-1/2,1/2]} \sum_{t'=1}^t z_{t'}v + \tfrac{v^2}{4\eta}\left(5 + \sum_{t'=1}^t z_{t'}^2\right)$$

Since $z_t$ is the derivative of $\log(1 - \tilde{g}_t v)$ evaluated at $v_t$, we see that $v_t$ are the outputs of a Follow-the-Regularized-Leader (FTRL) algorithm with regularizers $v^2\frac{1}{4\eta}(5 + \sum_{t=1}^t z_{t'}^2)$. That is, we are actually using Algorithm 1 with INNEROPTIMIZER equal to this FTRL algorithm. Using the FTRL analysis of (McMahan, 2017), we then have

$$R_T^v(\mathring{v}) \le \tfrac{\mathring{v}^2}{4\eta}\left(5 + \sum_{t=1}^T z_t^2\right) + \sum_{t=1}^T \frac{\eta z_t^2}{5 + \sum_{t'=1}^{t-1} z_{t'}^2}$$

Next, by convexity we have $\log(a) + b/(a+4) \le \log(a+b)$ for all $a > 0$, $0 < b < 4$. Since $|w_t| \le 1/2$ and $|g_t| \le 1$, $z_t^2 \le 4$. Therefore by induction we can show:

$$\sum_{t=1}^T \frac{z_t^2}{5+\sum_{t'=1}^{t-1} z_{t'}^2} \le \log\left(1 + \sum_{t=1}^T z_t^2\right)$$

so that since $|z_t| \le 2|\tilde{g}_t| \le 2|g_t|$,

$$R_T^v(\mathring{v}) \le \quad \le \tfrac{\mathring{v}^2}{4\eta}\left(5 + 4\sum_{t=1}^T g_t^2\right) + \eta\log\left(1 + 4\sum_{t=1}^T g_t^2\right)$$

Therefore by Lemma 2 we have

$$\begin{aligned} R_T(\mathring{w}) &\le \epsilon + \frac{|\mathring{w}|}{c}(\log(|\mathring{w}|/c\epsilon) - 1) + |\mathring{w}|cZ \\ &\quad + \frac{|\mathring{w}|c}{\eta}\left(5 + 4\sum_{t=1}^T g_t^2\right) + \frac{|\mathring{w}|\eta}{c}\log\left(1 + 4\sum_{t=1}^T g_t^2\right) \\ &= \epsilon + O\left(\frac{|\mathring{w}|}{c}\log\left(\frac{|\mathring{w}|G^\eta}{c\epsilon}\right) + \frac{|\mathring{w}|cG}{\eta}\right) \end{aligned}$$

for all $c \in [0, 1/2]$, where we have observed that $Z = \sum_{t=1}^T g_t^2 = G$ in one dimension, and dropped various constants for simplicity. Optimizing for $c$ we have

$$R_T(\mathring{w}) \le \epsilon + O\left(|\mathring{w}| \max\left\{\sqrt{\frac{G}{\eta}\log\left(\frac{|\mathring{w}|G^\eta\sqrt{\frac{Z}{\eta}}}{\epsilon}\right)},\right.\right.$$
$$\left.\left.\log\left(\frac{|\mathring{w}|G^\eta\sqrt{G/\eta}}{\epsilon}\right)\right\}\right)$$

The statement for $G_T$ comes from simply observing that $|\mathring{w}|_\infty \le 1/2$. □

## 5. Full Regret Bound

Now we combine the DIAGOPTIMIZER of the previous section with RECURSIVEOPTIMIZER. There are only a few details to address. First, since the analysis of DIAGOPTIMIZER is specific to the infinity-norm, we set $\|\cdot\|$ to be the infinity-norm in RECURSIVEOPTIMIZER and Theorem 1, which has $\|\cdot\|_1$ as dual norm. Second, note that the gradients provided to INNEROPTIMIZER satisfy $\|z_t\|_\infty = \|g_t/(1 - g_t \cdot v_t)\|_\infty \le 2\|g_t\|_\infty \le 2$ since $1 \ge \|g_t\|_1 \ge \|g_t\|_\infty$. Since the analysis of DIAGOPTIMIZER requires gradients bounded by 1, we rescale the gradients by a factor of 2, which scales up the regret by the same constant factor of 2. Therefore, we see that DIAGOPTIMIZER satisfies the hypotheses of Theorem 1 with

$$G_T(v/\|v\|_\infty) = O\left(\sum_{i=1}^d \frac{|v_i|}{\|v\|_\infty} \max\left\{\sqrt{G_i \log\left(\frac{dG_i}{\epsilon}\right)},\right.\right.$$
$$\left.\left.\log\left(\frac{dG_i}{\epsilon}\right)\right\}\right)$$

Thus by Theorem 1, in all cases we have the diagonal bound:

$$R_T(\mathring{w}) \le \tilde{O}\left(\epsilon + \sum_{i=1}^d |\mathring{w}_i| \sqrt{\sum_{t=1}^T g_{t,i}^2}\right) \tag{11}$$

and whenever $-\sum_{t=1}^T g_t \cdot \frac{\mathring{w}}{\|\mathring{w}\|_\infty} \ge 2G_T(\mathring{w}/\|\mathring{w}\|) = \tilde{O}(\sqrt{T})$, we have

$$R_T(\mathring{w}) \le \tilde{O}\left(\epsilon + \sqrt{\sum_{t=1}^T (g_t \cdot \mathring{w})^2}\right)$$

Note that this may be even a factor of $\sqrt{d}$ better than the standard full-matrix regret bounds (3), (4).

We recall that by Cauchy-Schwarz, the bound (11) also implies:

$$R_T(\mathring{w}) \le \tilde{O}\left(\|\mathring{w}\|_2 \sqrt{\sum_{t=1}^T \|g_t\|_2^2}\right)$$

which is the standard non-diagonal adaptive regret bound. Note that this may be a factor of $\sqrt{d}$ better than the diagonal Adagrad bound (1) when both $\mathring{w}$ and the gradients are dense.

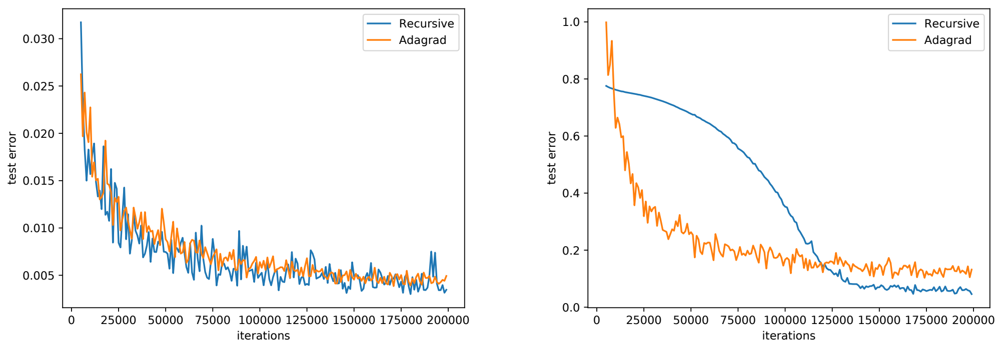
## 6. Experiments

We implemented RECURSIVEOPTIMIZER in TensorFlow (Abadi et al., 2016) and ran benchmarks on both synthetic data as well as several deep learning tasks (see Appendix D for full details).[1] We found that using the recent algorithm SCINOL of (Kempka et al., 2019) as the inner optimizer instead of Algorithm 2 provided qualitatively similar results on synthetic data but better empirical performance on the deep learning tasks, so we report results using SCINOL as the inner optimizer. SCINOL has essentially the same regret bound as in Theorem 2 (with slightly worse log factors), so this maintains our theoretical guarantees while allowing us to inherit the scale-invariance properties of SCINOL. This actually highlights another advantage of our reduction: we can take advantage of orthogonal advances in optimization.

Our synthetic data takes the form $\ell_t(w) = |x_t \cdot w - x_t \cdot \mathring{w}|$ where $x_t$ is randomly generated from a 100-dimensional $\mathcal{N}(0, \Sigma)$ and $\mathring{w}$ is some pre-selected optimal point. We generate $\Sigma$ to be a random matrix with exponentially decaying eigenvalues and condition number 750. We consider two cases: either $\mathring{w}$ is the eigenvector with smallest eigenvalue of $\Sigma$, or the eigenvector with largest eigenvalue. We compared RECURSIVEOPTIMIZER to diagonal ADAGRAD, both of which come with good theoretical guarantees. We used an ONS-based rather than FTRL-based inner-optimizer for RECURSIVEOPTIMIZER as this showed qualitatively similar but quantitatively slightly better performance. We stress that since our results are reductions, this change only effects the power of $T$ inside the logarithm in our bounds. For full implementation details, see Appendix D. The performance on a holdout set is shown in Figure 1. RECURSIVEOPTIMIZER enjoys an advantage in the poorly-conditioned regime while maintaining performance in the well-conditioned problem.

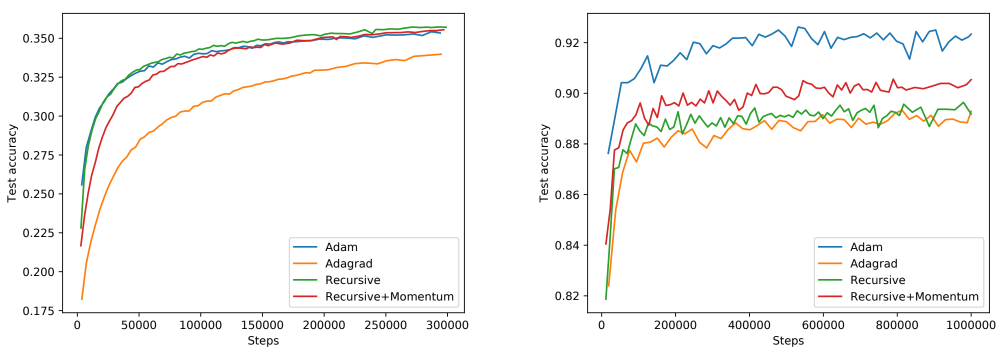
The dynamics of RECURSIVEOPTIMIZER in the poorly-conditioned problem bear some discussion. Recall that our full-matrix regret bounds actually do not appear until the sum of the gradients grows to a certain degree. It appears that this may provide a period of "slow convergence" during which the inner optimizer is presumably finding the optimal $\mathring{v}$, which is hard on poorly-conditioned problems. Once this is located, the algorithm makes progress very quickly.
We also test RECURSIVEOPTIMIZER on benchmark deep learning models. Specifically, we test performance with the ResNet-32 (He et al., 2016) model on the CIFAR-10 image recognition dataset (Krizhevsky & Hinton, 2009) and the

[1] Code available at: https://github.com/google-research/google-research/tree/master/recursive_optimizer




(a) (b)

*Figure 1.* Test Error in Synthetic Experiments. (a): $\mathring{w}$ is eigenvector with maximum eigenvalue (well-conditioned). (b): $\mathring{w}$ is eigenvector with minimum eigenvalue (poorly-conditioned).




(a) (b)

*Figure 2.* Deep Learning Experiments (a): Transformer test accuracy on LM1B. (b). ResNet-32 test accuracy on CIFAR-10. See Appendix D.2 for details on the momentum heuristic.

Transformer model (Vaswani et al., 2017; 2018) on LM-1B (Chelba et al., 2013) and other textual datasets. We record train and test error, both as a function of number of iterations as well as a function of wall time. We compare performance to the commonly used Adam (Kingma & Ba, 2014) and Adagrad optimizers (see Figure 2, and Appendix D). Even though our analysis relies heavily on duality and global properties of convexity, RECURSIVEOPTIMIZER still performs well on these non-convex tasks: we are competitive in all benchmarks, and marginally the best in the Transformer tasks. This suggests an interesting line of future research: most popular optimizers used in deep learning operate in a proximal manner, producing each iterate as an offset from the previous iterate. This seems more appropriate to the non-convex setting as gradients provide only local information. It may therefore be valuable to develop a proximal version of RECURSIVEOPTIMIZER that performs even better in the non-convex setting.

## 7. Conclusion

We have presented an algorithm that successfully obtains full-matrix style regret guarantees in certain settings without sacrificing runtime, space or regret in less favorable settings. The favorable settings we require for full-matrix performance are those in which the sum of the gradients exceeds the regret of some base algorithm, which should be $\tilde{O}(\sqrt{T})$. This suggests that any gradients with some systematic bias will satisfy our condition and exhibit full-matrix regret bounds for sufficiently large $T$. Our algorithmic design is based on techniques for unconstrained online learning, which necessitates an extra log factor in our regret over a mirror-descent algorithm tuned to the value of $\|\mathring{w}\|$. As such, it is an interesting question whether a mirror-descent style analysis can achieve similar results with oracle tuning.

## Acknowledgements

We thank all the reviewers for their thoughtful comments and Ryan Sepassi for answering our questions about TensorFlow and Tensor2Tensor.

## References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, 2016.

Agarwal, N., Bullins, B., Chen, X., Hazan, E., Singh, K., Zhang, C., and Zhang, Y. The case for full-matrix adaptive regularization. *arXiv preprint arXiv:1806.02958*, 2018.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

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

Cutkosky, A. and Boahen, K. A. Stochastic and adversarial online learning without hyperparameters. In *Advances in Neural Information Processing Systems*, pp. 5059–5067, 2017b.

Cutkosky, A. and Orabona, F. Black-box reductions for parameter-free online learning in Banach spaces. In *COLT*, 2018. URL https://arxiv.org/abs/1802.06293.

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

Gonen, A. and Shalev-Shwartz, S. Faster sgd using sketched conditioning. *arXiv preprint arXiv:1506.02649*, 2015.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned stochastic tensor optimization. *arXiv preprint arXiv:1802.09568*, 2018.

Hazan, E., Agarwal, A., and Kale, S. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Hazan, E. et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kempka, M., Kotłowski, W., and Warmuth, M. K. Adaptive scale-invariant online algorithms for learning linear models. *arXiv preprint arXiv:1902.07528*, 2019.

Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*, 2014.

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

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Luo, H., Agarwal, A., Cesa-Bianchi, N., and Langford, J. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems*, pp. 902–910, 2016.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

McMahan, B. and Streeter, M. No-regret algorithms for unconstrained online convex optimization. In *Advances in neural information processing systems*, pp. 2402–2410, 2012.

McMahan, H. B. A survey of algorithms and analysis for adaptive online learning. *The Journal of Machine Learning Research*, 18(1):3117–3166, 2017.

McMahan, H. B. and Orabona, F. Unconstrained online linear learning in hilbert spaces: Minimax algorithms and normal approximations. In *Conference on Learning Theory*, pp. 1020–1039, 2014.

McMahan, H. B. and Streeter, M. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

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

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

Orabona, F. and Pál, D. Coin betting and parameter-free online learning. In *Advances in Neural Information Processing Systems*, pp. 577–585, 2016.

Orabona, F. and Tommasi, T. Training deep networks without learning rates through coin betting. In *Advances in Neural Information Processing Systems*, pp. 2160–2170, 2017.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, L., Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N., and Uszkoreit, J. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018. URL http://arxiv.org/abs/1803.07416.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.