Winnowing with Gradient Descent

Ehsan Amid EAMID@GOOGLE.COM
Google Brain, Mountain View, CA 94043

Manfred K. Warmuth MANFRED@GOOGLE.COM
Google Brain, Mountain View, CA 94043

Editors: Jacob Abernethy and Shivani Agarwal

Abstract

The performance of multiplicative updates is typically logarithmic in the number of features when the targets are sparse. Strikingly, we show that the same property can also be achieved with gradient descent updates. We obtain this result by rewriting the non-negative weights w_i of multiplicative updates by u_i^2 and then performing a gradient descent step w.r.t. the new u_i parameters. We apply this method to the Winnow update, the Hedge update, and the unnormalized and normalized exponentiated gradient (EG) updates for linear regression. When the original weights w_i are scaled to sum to one (as done for Hedge and normalized EG), then in the corresponding reparameterized update, the u_i parameters are now divided by $\|\boldsymbol{u}\|_2$ after the gradient descent step. We show that these reparameterizations closely track the original multiplicative updates by proving in each case the same online regret bounds (albeit in some cases, with slightly different constants).

As a side, our work exhibits a simple two-layer linear neural network that, when trained with gradient descent, can experimentally solve a certain sparse linear problem (known as the Hadamard problem) with exponentially fewer examples than any kernel method.

1. Introduction

Multiplicative updates started with the Winnow algorithm for learning disjunctions and linear threshold functions (Littlestone, 1988). These algorithms update their weights by multiplicative factors and often learn sparse targets with a logarithmic dependence on the number of features. For example, Winnow makes at most $k \log n$ mistakes on the sequence of n-dimensional boolean vectors labeled consistently with a k-literal disjunction. Another paradigmatic case is the so called "expert setting" where the learner is to perform as well as a single feature/expert. For sequences of n-dimensional feature vectors with one consistent feature, the total loss of the update is typically $\mathcal{O}(\log n)$. The situation is repeated for linear regression; when the weight vector realizing the labels is sparse, then the normalized and unnormalized EG algorithms (Kivinen and Warmuth, 1997) incur loss at most $\mathcal{O}(\log n)$. So far we focused on the noise-free case. In the noisy case (when the solution has non-zero loss), additional square root terms appear in the regret bounds of these algorithms.

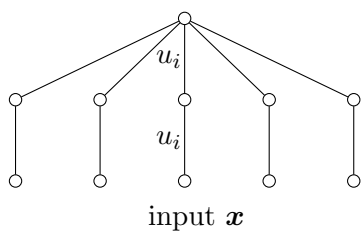


Figure 1: Reparameterizing the weights w_i of a linear neuron by u_i^2.

We first observe that in continuous-time, multiplicative updates can be exactly reparameterized as continuous-time gradient descent (GD). This is done by replacing the linear weights w_i by u_i^2 and applying continuous-time GD w.r.t. the new parameter u_i (See Figure 1). We show how the discretization of the continuous multiplicative updates results in the usual Winnow and EG updates. More importantly, the discretization of the reparameterized continuous multiplicative updates results in discrete-time GD variants of these updates. However, the equivalence between the multiplicative updates and the GD reparameterizations does not hold after discretization. Nevertheless, we show that the GD reparameterizations closely track the multiplicative originals by proving the same regret bounds for the new updates, as were known for the originals. In each case, the constants we obtain for the second order terms are the same or slightly different in either direction.

For the discrete GD updates, lower bounds that grow linearly in the number of features n are known in some settings (Kivinen et al., 1997; Vishwanathan and Warmuth, 2005). In these lower bounds, the instances are essentially the rows of an $n \times n$ Hadamard matrix and the target, in the simplest case, is a single column of this matrix. When averaged over targets, the lower bound for linear regression holds even if the input vectors $\boldsymbol{x}$ are replaced by a feature map $\phi(\boldsymbol{x})$ of any kernel (Vishwanathan and Warmuth, 2005). It was conjectured in (Derezinski and Warmuth, 2014) that the linear lower bound holds for any neural network, as long as it is trained with GD. This conjecture is now contradicted by the linear network in Figure 1. If this network is trained with online GD (i.e. backpropagation) w.r.t. the squared Euclidean loss, then the update coincides with our discrete-time GD reparameterization of the EGU update. Experimentally, GD on this network solves the Hadamard problem as efficiently as EG and EGU (on a single neuron). Interestingly, this simple two-layer linear network, when trained with GD, can realize the $\log n$ dependence on the number of examples. Also surprisingly, when all the missing $n^2 - n$ connections in the bottom layer of Figure 1 are reinstated and initialized to zero, then experimentally the loss of the now fully-connected two-layer linear network again decays only linearly when trained with GD.

Previous work. Online regret bounds for multiplicative updates have a long history (see e.g. (Littlestone, 1988; Vovk, 1990; Kivinen and Warmuth, 1997)). A key insight of Gunasekar et al. (2017) (followed by Woodworth et al. (2019) and Vaskevicius et al. (2019)) was that if the weights w_i of a linear predictor are replaced by u_i^2, then the continuous-time gradient descent update on the u_i's imposes an implicit regularization on the w_i's that makes the reparameterized weight vector $\boldsymbol{w}$ converge to the minimum L_1-norm solution.[1] The EG updates are known to connect with the L_1-norm regularization. Indeed, the relative entropy (the divergence motivating the EG updates) is strongly convex w.r.t. the L_1-norm (Shalev-Shwartz et al., 2012). We show that the continuous-time unnormalized EG algorithm is in fact equivalent to the continuous-time GD on the u_i's. Curiously enough, this reparameterization method has been used by game theorists to convert problems defined on the unit simplex to the unit sphere (Akin, 1979; Sandholm, 2010). In particular, the replicator dynamics of evolutionary game theory corresponds to EG, and GD equivalents have been investigated for this dynamic. In the continuous case, it is easy to see the

1. Note that Gunasekar et al. (2017) analyzed the much richer matrix context. For the sake of simplicity, we focus on the diagonal (i.e. vector) case.

equivalence between the EG updates and the corresponding reparameterizations as GD. However, the main contribution of this paper is showing that essentially the same regret bounds hold for the discretized GD reparameterizations of the multiplicative updates.

Note that obtaining an online regret bound is one of the most stringent learning criterion since it must hold for worst-case sequences of examples. Standard conversions to randomized settings also exist (see e.g. (Kivinen and Warmuth, 1997)). Furthermore, EGU is a special case of mirror descent where the link function is the component-wise logarithm. A more general framework for reparameterizing mirror descent updates was developed in a recent paper (Amid and Warmuth, 2020).

Outline. In the next section, we show the equivalence between the continuous-time exponentiated gradient updates and their reparameterized gradient descent versions. We also motivate the discretizations of the continuous updates. In the following three chapters, we reprove the regret bounds for the discrete gradient descent variants of Winnow, the Hedge algorithm, and the EG and EGU algorithms for linear regression. Lower bounds and some implications of neural network training are briefly discussed in Section 6. We conclude the paper with a number of open problems in Section 7.

Notation. We use $\odot$ and $\oslash$ to denote element-wise vector multiplication and division, respectively. The dot symbol denotes the time derivative, i.e. $\dot{\boldsymbol{v}}(t) \coloneqq \frac{\partial}{\partial t}\boldsymbol{v}(t)$.

2. Reparameterizing the Continuous-time Exponentiated Gradient

In the following, we discuss the continuous-time EGU and EG updates[2] and derive their reparameterized forms. Although the reparameterization method presented here is more versatile and applies to a wider class of continuous-time mirror descent updates (Amid and Warmuth, 2020), we only focus on these two main cases in this paper.

The continuous-time EGU update can be seen as the solution of the following ordinary differential equation (ODE) defined on $\mathbb{R}^n_{\geq 0}$:

$$\dot{\log \boldsymbol{w}}(t) = -\eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t)), \text{ with initial condition } \boldsymbol{w}(t{=}0) {=} \boldsymbol{w}^0 \in \mathbb{R}^n_{\geq 0}\,. \tag{1}$$

Here, $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t))$ denotes the gradient of the loss $L(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$ evaluated at $\boldsymbol{w}(t)$ and η is a positive learning rate. Typically, the loss $L(\,\cdot\,)$ also depends on a given example $\boldsymbol{x}(t)$ and possibly a label $y(t)$. Although the dynamic of the continuous-time EGU (1) is fundamentally different than a dynamic based on gradient descent, the following theorem shows that the same update can be realized using gradient descent via a simple reparameterization. Namely, we substitute $\boldsymbol{w}(t) = q\big(\boldsymbol{u}(t)\big) \coloneqq \boldsymbol{u}(t) \odot \boldsymbol{u}(t)$ where the new parameter $\boldsymbol{u}(t) \in \mathbb{R}^n$ is updated via gradient descent on the composite loss $L{\circ}q(\cdot)$.[3]

Theorem 1 (Reparameterized continuous-time EGU) *The solution of the ODE (1) is equal to $\boldsymbol{w}(t) = q\big(\boldsymbol{u}(t)\big)$, for all $t \geq 0$, where $q(\boldsymbol{u}) \coloneqq \boldsymbol{u} \odot \boldsymbol{u}$ and the new parameter $\boldsymbol{u}(t)$ is the solution of the following ODE defined on $\mathbb{R}^n$:*

$$\dot{\boldsymbol{u}}(t) = -\eta/4\, \nabla_{\boldsymbol{u}}\, L{\circ}q\big(\boldsymbol{u}(t)\big), \;\; s.t. \;\; \boldsymbol{u}(t{=}0) = \boldsymbol{u}^0 \in \mathbb{R}^n \;\; and \;\; \boldsymbol{u}^0 \odot \boldsymbol{u}^0 = \boldsymbol{w}^0. \tag{2}$$

2. Recall that for any loss $L(\boldsymbol{w})$, where $\boldsymbol{w} \in \mathbb{R}^n_{\geq \boldsymbol{0}}$, the unnormalized exponentiated gradient update (EGU) resets $\boldsymbol{w}$ to $\boldsymbol{w} \odot \exp(-\eta \nabla L(\boldsymbol{w})) \in \mathbb{R}^n_{\geq \boldsymbol{0}}$. The normalized version (EG) divides by the sum of the updated weights, thus projecting the weight vector onto the unit simplex Δ^{n-1}.

3. Even if $L(\boldsymbol{w})$ is convex in $\boldsymbol{w}$, the composite loss $L{\circ}q\,(\boldsymbol{u})$ might not be convex in the new parameter $\boldsymbol{u}$.

Proof By the chain rule, $\dot{\log \boldsymbol{w}}(t) = \dot{\boldsymbol{w}}(t) \oslash \boldsymbol{w}(t)$. Thus the ODE (1) can be written as

$$\dot{\boldsymbol{w}}(t) = -\eta\, \boldsymbol{w}(t) \odot \nabla_{\boldsymbol{w}} L(\boldsymbol{w}(t))\,. \tag{3}$$

Similarly,
$$\dot{\boldsymbol{w}}(t) = \frac{\partial}{\partial t}\big(\boldsymbol{u}(t) \odot \boldsymbol{u}(t)\big) = 2\,\boldsymbol{u}(t) \odot \dot{\boldsymbol{u}}(t)\,. \tag{4}$$

Plugging (2) into (4) and using $\nabla_{\boldsymbol{u}}\, L{\circ}q\big(\boldsymbol{u}(t)\big) = 2\,\boldsymbol{u}(t) \odot \nabla_{\boldsymbol{w}}\, L(\boldsymbol{w}(t))$ and $\boldsymbol{w}(t) = q\big(\boldsymbol{u}(t)\big)$ yields (3) and concludes the proof. ■

The discrete-time EGU can be derived as the finite difference approximation of (1) with a step-size of one, that is, $\log \boldsymbol{w}^{t+1} - \log \boldsymbol{w}^t = -\eta\, \nabla_{\boldsymbol{w}} L(\boldsymbol{w}^t)$. Alternatively, this is also the solution to the following minimization problem:

$$\boldsymbol{w}^{t+1} = \operatorname*{argmin}_{\widetilde{\boldsymbol{w}} \in \mathbb{R}^n_{\geq 0}} \Big\{ {}^1\!/_\eta\, D_{\mathrm{RE}}(\widetilde{\boldsymbol{w}}, \boldsymbol{w}^t) + \hat{L}(\widetilde{\boldsymbol{w}} \,|\, \boldsymbol{w}^t) \Big\}\,, \tag{5}$$

where $D_{\mathrm{RE}}(\widetilde{\boldsymbol{w}}, \boldsymbol{w}) = \sum_{i=1}^n \widetilde{w}_i \log \frac{\widetilde{w}_i}{w_i} - \widetilde{w}_i + w_i$ denotes the relative entropy divergence and $\hat{L}(\widetilde{\boldsymbol{w}} \,|\, \boldsymbol{w}^t) = L(\widetilde{\boldsymbol{w}}) + (\widetilde{\boldsymbol{w}} - \boldsymbol{w}^t) \cdot \nabla_{\boldsymbol{w}} L(\boldsymbol{w}^t)$ is the Taylor approximation[4] of the loss $L(\widetilde{\boldsymbol{w}})$ at $\boldsymbol{w}^t$. Similarly, discretizing (2) yields the discrete-time reparameterized EGU update

$$\boldsymbol{u}^{t+1} - \boldsymbol{u}^t = -{}^\eta\!/_4\, \nabla_{\boldsymbol{u}}\, L{\circ}q\big(\boldsymbol{u}^t\big) = -{}^\eta\!/_2\, \boldsymbol{u}^t \odot \nabla_{\boldsymbol{w}}\, L\big(\boldsymbol{w}^t\big)\,, \tag{6}$$

with $\boldsymbol{w}^t = \boldsymbol{u}^t \odot \boldsymbol{u}^t$. This can be seen as the solution to the minimization problem

$$\boldsymbol{u}^{t+1} = \operatorname*{argmin}_{\widetilde{\boldsymbol{u}} \in \mathbb{R}^n} \Big\{ {}^2\!/_\eta\, \|\widetilde{\boldsymbol{u}} - \boldsymbol{u}^t\|_2^2 + \widehat{L{\circ}q}\big(\widetilde{\boldsymbol{u}} \,|\, \boldsymbol{u}^t\big) \Big\}\,, \tag{7}$$

where $\widehat{L{\circ}q}\big(\widetilde{\boldsymbol{u}} \,|\, \boldsymbol{u}^t\big)$ is the first-order Taylor series approximation of $L{\circ}q(\widetilde{\boldsymbol{u}})$ at $\boldsymbol{u}^t$:

$$\widehat{L{\circ}q}\big(\widetilde{\boldsymbol{u}} \,|\, \boldsymbol{u}^t\big) = L{\circ}q\big(\boldsymbol{u}^t\big) + (\widetilde{\boldsymbol{u}} - \boldsymbol{u}^t) \cdot \nabla_{\boldsymbol{u}}\, L{\circ}q\big(\boldsymbol{u}^t\big)\,. \tag{8}$$

A similar continuous-time dynamic can be constructed for the normalized EG update and its equivalent reparameterized form.[5] This involves applying projected gradient updates to maintain the constraint $\|\boldsymbol{w}(t)\|_1 = 1$ (respectively, $\|\boldsymbol{u}(t)\|_2^2 = 1$). Here, we state the result for the discretized reparameterized form and refer the reader to (Amid and Warmuth, 2020) for further details. Adding the Lagrange multiplier λ to (7) to enforce the constraint $\big(\boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}\big) \cdot \mathbf{1} = 1$, we have

$$\boldsymbol{u}^{t+1} = \operatorname*{argmin}_{\widetilde{\boldsymbol{u}} \in \mathbb{R}^n} \Big\{ {}^1\!/_\eta\, \|\widetilde{\boldsymbol{u}} - \boldsymbol{u}^t\|_2^2 + \widehat{L{\circ}q}\big(\widetilde{\boldsymbol{u}} \,|\, \boldsymbol{u}^t\big) + \lambda \Big(\big(\widetilde{\boldsymbol{u}} \odot \widetilde{\boldsymbol{u}}\big) \cdot \mathbf{1} - 1 \Big) \Big\}\,, \tag{9}$$

which results in the reparameterized EG update

$$\boldsymbol{u}^{t+1} = \frac{\boldsymbol{u}^t - {}^\eta\!/_2\, \nabla_{\boldsymbol{u}}\, L{\circ}q\big(\boldsymbol{u}^t\big)}{\big\|\boldsymbol{u}^t - {}^\eta\!/_2\, \nabla_{\boldsymbol{u}}\, L{\circ}q\big(\boldsymbol{u}^t\big)\big\|_2}\,, \quad \text{thus} \quad \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1} \in \Delta^{n-1}\,. \tag{10}$$

4. Note that a backward Euler approximation of (1) yields the *prox* or *implicit form* of the update (Rockafellar, 1976; Kivinen et al., 2006), which corresponds to using $L(\widetilde{\boldsymbol{w}})$ instead of $\hat{L}(\widetilde{\boldsymbol{w}} \,|\, \boldsymbol{w}^t)$ in (5).
5. In the remainder of the paper, we use $\boldsymbol{w} = q(\boldsymbol{u}) = \boldsymbol{u} \odot \boldsymbol{u}$ as the reparameterization. The constants can be absorbed into the learning rates.

Clearly, after the discretization, the EGU and EG updates and their reparameterized forms are no longer equivalent. The key question that naturally arises is the following: *How well does the discretized reparameterized EGU update (and its normalized form) approximate the original EGU (respectively, the original EG) update?* In the following, we address this question by comparing the worst-case regrets of the original (un)normalized EG and its reparameterizations on three problems, namely, the Winnow algorithm for binary classification, the Hedge algorithm for the expert setting, and EGU and EG for linear regression. The following regret bounds for the reparameterizations are proven by bounding the progress towards a comparator. Curiously enough, the progress is measured i.t.o. the relative entropy divergence even though the reparameterized updates are motivated by regularizing with the squared Euclidean distance.

3. The Reparameterization of the Winnow algorithm

The Winnow algorithm learns a linear threshold function for the task of binary classification. It is a special case of EGU when the loss is the hinge loss. This loss on example $(\boldsymbol{x}^t, y^t)$ (for $y^t \in \{\pm 1\}$) is defined as $L_H(\boldsymbol{w}|\, \boldsymbol{x}^t, y^t) = \left[-\, y^t\, (\boldsymbol{w} \cdot \boldsymbol{x}^t - \theta)\right]_+$, where $\theta \in \mathbb{R}$ is a threshold. GD w.r.t. the same loss results in the Perceptron update. (See (Kivinen et al., 1997) for a comparative study.) Below we give the Winnow algorithm and its reparameterization as GD. Both algorithms update their weights only when a mistake occurs because when the prediction is correct, the gradient of the hinge loss is zero (Gentile and Warmuth, 1998).

Algorithm 1 Winnow Algorithm

Parameters initial weight $w^1 > 0$, learning rate $\eta > 0$, threshold $\theta > 0$

Initialize $\boldsymbol{w}^1 = w^1\, \mathbf{1}_n$

for $t = 1$ to T **do**

Receive instance $\boldsymbol{x}^t \in [0,1]^n$

Predict $\hat{y}^t = \begin{cases} +1 & \text{if } \boldsymbol{w}^t \cdot \boldsymbol{x}^t \geq \theta \\ -1 & \text{otherwise} \end{cases}$

Receive label y^t and update:

$$\boldsymbol{w}^{t+1} = \begin{cases} \boldsymbol{w}^t & \text{if } \hat{y}^t = y^t \\ \boldsymbol{w}^t \odot \exp(\eta\, y^t\, \boldsymbol{x}^t) & \text{otherwise} \end{cases}$$

Algorithm 2 Reparameterized Winnow

Parameters initial weight $u^1 \in \mathbb{R}$, learning rate $\eta > 0$, threshold $\theta > 0$

Initialize $\boldsymbol{u}^1 = u^1\, \mathbf{1}_n$

for $t = 1$ to T **do**

Receive instance $\boldsymbol{x}^t \in [0,1]^n$

Predict $\hat{y}^t = \begin{cases} +1 & \text{if } (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \geq \theta \\ -1 & \text{otherwise} \end{cases}$

Receive label y^t and update:

$$\boldsymbol{u}^{t+1} = \begin{cases} \boldsymbol{u}^t & \text{if } \hat{y}^t = y^t \\ \boldsymbol{u}^t + \eta\, y^t\, (\boldsymbol{u}^t \odot \boldsymbol{x}^t) & \text{otherwise} \end{cases}$$

Theorem 2 (Winnow bound (Littlestone, 1988; Warmuth, 2007)) *Given any sequence of examples $(\boldsymbol{x}^t, y^t)$ such that $\boldsymbol{x}^t \in [0,1]^n$, the labels y^t are ± 1, and there is a weight $\boldsymbol{r}$ with k ones and $n-k$ zeros such that*

$$\boldsymbol{r} \cdot \boldsymbol{x}^t = \begin{cases} \geq \frac{1}{2} & \text{if } \; y^t = +1 \\ 0 & \text{otherwise}\,, \end{cases}$$

then the Winnow algorithm makes at most $7.18\, k \log \frac{n}{k}$ mistakes on this sequence, when $\eta \approx 1.28$, $\theta = 0.19$, and $w^1 = n/k$.

Note for the sake of simplicity we only address the case when there exists a consistent disjunction. In the more general case, there are the additional terms in the mistake bound that involve the number of attribute errors w.r.t. the best disjunction.

Theorem 3 (Reparameterized Winnow bound) *Given that the assumptions of Theorem 2 hold, for any sequence of examples $(\boldsymbol{x}^t, y^t)$ the reparameterized Winnow algorithm makes at most $5.66\, k \log \frac{n}{k}$ mistakes on this sequence, when $\eta \approx 0.85$, $\theta = 0.18$, and $u^1 = \sqrt{n/k}$.*

Proof We lower bound the per trial progress $D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1})$ towards any comparator $\boldsymbol{r}$ which satisfies the constraints in Theorem 2. Note that if no mistake occurs, $\boldsymbol{u}^{t+1} = \boldsymbol{u}^t$. Otherwise, $\boldsymbol{u}^{t+1}$ is updated to $\boldsymbol{u}^t + \eta\, y^t\, (\boldsymbol{u}^t \odot \boldsymbol{x}^t) = \boldsymbol{u}^t \odot (\boldsymbol{1} + \eta\, y^t\, \boldsymbol{x}^t)$. Assuming $\eta \leq 1$, and using the facts that $\log(1 + \eta y^t x_i^t) \geq x_i^t \log(1 + \eta y^t)$ and $(x_i^t)^2 \leq x_i^t$ for $x_i^t \in [0, 1]$, we have

$$\begin{aligned} D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) &= 2\boldsymbol{r} \cdot \log(\boldsymbol{1} + \eta y^t \boldsymbol{x}^t) - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot (2\eta y^t \boldsymbol{x}^t + \eta^2 \boldsymbol{x}^t \odot \boldsymbol{x}^t) \\ &\geq 2\, \boldsymbol{r} \cdot \boldsymbol{x}^t \, \log(1 + \eta\, y^t) - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \, (2\, \eta\, y^t + \eta^2)\,. \end{aligned}$$

A mistake occurs if $\hat{y}^t \neq y^t$. If $y^t = -1$, then $\boldsymbol{r} \cdot \boldsymbol{x}^t = 0$ by the assumption and $(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \geq \theta$. Thus, the lower bound on the progress becomes

$$(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t (2\, \eta - \eta^2) \geq \theta (2\, \eta - \eta^2)\,.$$

If $y^t = +1$, then we have $\boldsymbol{r} \cdot \boldsymbol{x}^t \geq \frac{1}{2}$ by the assumption and also $(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \leq \theta$. Thus in this case the progress is lower bounded by

$$\log(1 + \eta) - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t \, (2\, \eta + \eta^2) \geq \log(1 + \eta) - \theta (2\, \eta + \eta^2)\,.$$

Setting the two values to be equal, we obtain $\theta = \frac{\log(1+\eta)}{4\,\eta}$. With this choice of θ, the progress per mistake is always lower bounded by $1/4\,(2 - \eta) \log(1 + \eta)$ which is maximized for $\eta \approx 0.85$. This choice of η yields $\theta \approx 0.18$ and the progress is ≈ 0.18. Summing over all trials and denoting the number of mistakes by M, we have

$$\underbrace{D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1)}_{k \log \frac{n}{k}} - \underbrace{D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{T+1} \odot \boldsymbol{u}^{T+1})}_{\geq 0} \geq 0.18\, M\,, \text{ and thus } M \leq 5.56\, k \log \frac{n}{k}. \qquad \blacksquare$$

4. The Reparameterization of the Hedge Algorithm

An important algorithm in the online expert setting is the Randomized Weighted Majority algorithm (Littlestone and Warmuth, 1994). Here, we only discuss the simplified version known as the Hedge algorithm (Freund and Schapire, 1997). This algorithm maintains a non-negative probability vector $\boldsymbol{w}^t \in \Delta^{n-1}$ such that $\sum_i w_i^t = 1$. At trial t, the algorithm draws an expert/feature i with probability w_i^t and upon receiving the loss vector $\boldsymbol{\ell}^t \in [0,1]^n$, it incurs the expected loss $\boldsymbol{w}^t \cdot \boldsymbol{\ell}^t$. The weights are then updated by a multiplicative exponentiated gradient term and re-normalized (see Algorithm 3) to assure that the non-negative weights sum to one.

The Hedge update and its reparameterization are a special case of EG and reparameterized EG when the losses are the dot loss $\widetilde{\boldsymbol{w}} \cdot \boldsymbol{\ell}^t$ and $\widetilde{\boldsymbol{u}} \odot \widetilde{\boldsymbol{u}} \cdot \boldsymbol{\ell}^t$, respectively. The Hedge update at round t is motivated by minimizing the relative entropy to the current weight $\boldsymbol{w}^t$ plus the dot loss:

$$\boldsymbol{w}^{t+1} = \operatorname*{argmin}_{\widetilde{\boldsymbol{w}} \in \Delta^{n-1}} \left\{ 1/\eta \, D_{\text{RE}}(\widetilde{\boldsymbol{w}}, \boldsymbol{w}^t) + \widetilde{\boldsymbol{w}} \cdot \boldsymbol{\ell}^t \right\}. \tag{11}$$

Algorithm 3 Hedge Algorithm

Parameters initial probability vector $\boldsymbol{w}^1 \in \Delta^{n-1}$, learning rate $\eta > 0$

for $t = 1$ to T **do**

Draw expert i with probability w_i^t

Incur loss ℓ_i^t & expected loss $\boldsymbol{w}^t \cdot \boldsymbol{\ell}^t$

Update:

$$\boldsymbol{w}^{t+1} = \frac{\boldsymbol{w}^t \odot \exp(-\eta \, \boldsymbol{\ell}^t)}{\sum_i w_i^t \exp(-\eta \, \ell_i^t)}$$

Algorithm 4 Reparameterized Hedge Alg.

Parameters initial weight vector $\boldsymbol{u}^1 \in \mathbb{R}^n$ s.t. $\|\boldsymbol{u}^1\|_2 = 1$, learning rate $\eta > 0$

for $t = 1$ to T **do**

Draw expert i with probability $(u_i^t)^2$

Incur loss ℓ_i^t & expected loss $(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t$

Update:

$$\boldsymbol{u}^{t+1} = \frac{\boldsymbol{u}^t - \eta \, \boldsymbol{u}^t \odot \boldsymbol{\ell}^t}{\|\boldsymbol{u}^t - \eta \, \boldsymbol{u}^t \odot \boldsymbol{\ell}^t\|_2}$$

Theorem 4 (Hedge bound (Freund and Schapire, 1997)) *For any sequence of loss vectors $\{\boldsymbol{\ell}^t\}_{t=1}^T \in [0,1]^n$ such that $\min_{i \in [n]} \sum_{t=1}^T \ell_i^t \le L$, any comparator $\boldsymbol{r} \in \Delta^{n-1}$, and any start vector $\boldsymbol{w}^1 \in \Delta^{n-1}$ such that $D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1) \le D \le \log n$, the total expected loss of the Hedge algorithm with start vector $\boldsymbol{w}^1$ and learning rate $\eta = \log(1 + \sqrt{2D/L})$ is bounded as*

$$\sum_t \boldsymbol{w}^t \cdot \boldsymbol{\ell}^t \le \sum_t \boldsymbol{r} \cdot \boldsymbol{\ell}^t + \sqrt{2LD} + D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1).$$

In the reparameterized Hedge update, $\boldsymbol{w}$ is replaced with $\boldsymbol{u} \odot \boldsymbol{u}$ where $\boldsymbol{u} \in \mathbb{R}^n$ and $\|\boldsymbol{u}\|_2 = 1$. That is, the reparameterized weight $\boldsymbol{u}$ lies on the unit sphere and the squared weight $\boldsymbol{u} \odot \boldsymbol{u}$ corresponds to an n-dimensional probability vector. The update is motivated by minimizing the squared Euclidean distance as the inertia term plus the expected loss,[6]

$$\boldsymbol{u}^{t+1} = \operatorname*{argmin}_{\widetilde{\boldsymbol{u}} \text{ s.t. } \|\widetilde{\boldsymbol{u}}\|_2 = 1} \left\{ 1/\eta \, \|\widetilde{\boldsymbol{u}} - \boldsymbol{u}^t\|_2^2 + (\widetilde{\boldsymbol{u}} \odot \widetilde{\boldsymbol{u}}) \cdot \boldsymbol{\ell}^t \right\}. \tag{12}$$

Using a Lagrange multiplier to enforce the constraint that $\sum_i (u_i^{t+1})^2 = 1$ and solving for $\boldsymbol{u}^{t+1}$ yields the reparameterized Hedge update of Algorithm 4.

Theorem 5 (Reparameterized Hedge bound) *For any sequence of loss vectors $\{\boldsymbol{\ell}^t\}_{t=1}^T \in [0,1]^n$ such that $\min_{i \in [n]} \sum_{t=1}^T \ell_i^t \le L$, any comparator $\boldsymbol{r} \in \Delta^{n-1}$ and any start vector $\boldsymbol{u}^1 \in \boldsymbol{R}^n$ such that $\|\boldsymbol{u}^1\|_2 = 1$ and $D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) \le D \le \log n$, the total expected loss of reparameterized Hedge with learning rate $\eta = (1 + \sqrt{L/D})^{-1}$ is bounded as*

$$\sum_t (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t \le \sum_t \boldsymbol{r} \cdot \boldsymbol{\ell}^t + 2\sqrt{LD} + D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1).$$

6. A similar first-order Taylor approximation to (8) is required to obtain the explicit update.

Proof We lower bound the progress of the algorithm towards an arbitrary comparator $\boldsymbol{r} \in \Delta^{n-1}$. Assuming $\eta < 1$, the progress can be written as

$$\begin{aligned} & D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\ & \qquad = 2\,\boldsymbol{r} \cdot \log\left(\mathbf{1} - \eta\,\boldsymbol{\ell}^t\right) - \log\left((\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot (\mathbf{1} - 2\eta\,\boldsymbol{\ell}^t + \eta^2\,\boldsymbol{\ell}^t \odot \boldsymbol{\ell}^t)\right) \\ & \qquad \geq 2\,\boldsymbol{r} \cdot \boldsymbol{\ell}^t \log(1-\eta) - \log\left(1 - (2\eta - \eta^2)\,(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t\right). \end{aligned}$$

Using $\log(1-x) \geq -x/(1-x)$ and $-\log(1-x) \geq x$ for $0 \leq x < 1$ yields

$$D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \geq -\frac{2\,\eta}{1-\eta}\,\boldsymbol{r} \cdot \boldsymbol{\ell}^t + (2\eta - \eta^2)\,(\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t\,.$$

Summing over all trials and re-arranging the terms results in the following bound on the loss of the algorithm

$$\sum_t (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{\ell}^t \leq \frac{\frac{2\,\eta}{1-\eta} \sum_t \boldsymbol{r} \cdot \boldsymbol{\ell}^t + D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{T+1} \odot \boldsymbol{u}^{T+1})}{2\,\eta - \eta^2}\,.$$

Setting η to $(1 + \sqrt{L/D})^{-1}$ and substituting the values of L and D yields the bound. ■

Note that the regret bound of Theorem 5 for the reparameterized Hedge has an additional $\sqrt{2}$ factor before the square root term in its regret bound. By plotting the progress, we can show that this additional factor disappears if you use the alternate learning rate $\eta = \sqrt{D/(D + 2\,L)}$. Based on this evidence, we conjecture that with the alternate tuning, the reparameterized Hedge has the same regret bound as the original.

Another approximation of the Hedge update has been analysed in (Cesa-Bianchi et al., 2007), called the *Prod* update. It replaces the exponential factors $\exp(-\eta\,\ell_i)$ used in Hedge by their Taylor expansions $(1 - \eta\,\ell_i)$ and normalizes multiplicatively.[7] Our reparameterized GD update is subtly different: it is a GD update w.r.t. the square roots of the weights as the parameters.

5. Reparameterizations of EGU and EG for Linear Regression

The regret bounds for linear regression using GD, EG, and EGU have been analyzed extensively in (Kivinen and Warmuth, 1997). Here, we derive similar bounds for the reparameterized EG and EGU updates.

We first recall the original EGU algorithm for linear regression which maintains a weight vector $\boldsymbol{w}^t \in \mathbb{R}^n_{\geq 0}$. Upon receiving input $\boldsymbol{x}^t \in \mathbb{R}^n$ at round t, the algorithm predicts with $\hat{y}^t = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$. It then receives the response y^t, incurs loss $(y^t - \hat{y}^t)^2$, and updates as:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t \odot \exp\left(-\,2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t\right).$$

Kivinen and Warmuth (1997) analyze a slight variant of EGU: given a sequence of trials $(\boldsymbol{x}^t, y^t)$ for which, $y^t \in [0, Y]$ for all t and some $Y > 0$, the variant uses the following clipped prediction in its update: $\hat{y}^t = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$ if $\hat{y}^t \leq Y$, and $\hat{y}^t = Y$ otherwise.

7. The same Taylor expansion is used in the *Approximated EG* update of (Kivinen and Warmuth, 1997), but that update keeps the weight sum as one by subtracting a term. It is motivated by the χ^2-divergence.

Theorem 6 (Linear regression with EGU (Kivinen and Warmuth, 1997)) *Let $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ be any sequence such that $\boldsymbol{x}^t \in [0, X]^n$ and $y^t \in [0, Y]$ for some constants $X, Y > 0$. Then for any comparator $\boldsymbol{r} \in \mathbb{R}^n_{\geq 0}$, EGU with learning rate $\eta = 1/(3XY)$ and arbitrary start point $\boldsymbol{w}^1 \in \mathbb{R}^n_{\geq 0}$ satisfies the total loss bound*

$$\sum_{t=1}^{T} (y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t)^2 \leq 3 \left(\sum_t (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 + X\, Y D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1) \right). \tag{13}$$

Furthermore, let L and D be constants such that $\sum_t (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 \leq L$ and $D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1) \leq D$ and let

$$\eta = \frac{\sqrt{D}}{\sqrt{2LXY} + 2XY\sqrt{D}}. \tag{14}$$

Then we have

$$\sum_{t=1}^{T} (y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t)^2 \leq \sum_{t=1}^{T} (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 + 2\sqrt{2LXYD} + 2\, XY D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1). \tag{15}$$

Reparameterized EGU uses $\boldsymbol{u}^t \odot \boldsymbol{u}^t$ as its weights,[8] where $\boldsymbol{u}^1 \in \mathbb{R}^n$, and updates as

$$\boldsymbol{u}^{t+1} = \boldsymbol{u}^t - \eta\, (\hat{y}^t - y^t)\, \boldsymbol{u}^t \odot \boldsymbol{x}^t, \tag{16}$$

where $\hat{y}^t$ is again the clipped prediction: $\min\bigl(\boldsymbol{w}^t \cdot \boldsymbol{x}^t, Y\bigr)$.

Theorem 7 (Reparameterized EGU linear regression) *Let $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ be any sequence such that $\boldsymbol{x}^t \in [0, X]^n$ and $y^t \in [0, Y]$ for some constants $X, Y > 0$. Then for any comparator $\boldsymbol{r} \in \mathbb{R}^n_{\geq 0}$, reparameterized EGU with learning rate $\eta = 1/(3XY)$ and arbitrary start point $\boldsymbol{w}^1 \in \mathbb{R}^n_{\geq 0}$ satisfies the same total loss bound (13) as the original EGU when replacing $\boldsymbol{w}^t$ with $\boldsymbol{u}^t \odot \boldsymbol{u}^t$. Moreover, for the same constants L and D as in Theorem 6 and by setting η to (14), the reparameterized EGU achieves the same bound as in (15).*

Proof sketch We lower bound the progress $D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1})$ of the algorithm towards any comparator $\boldsymbol{r} \in \mathbb{R}^n_{\geq 0}$ by $a\,(y^t - \hat{y})^2 - b\,(y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2$, for some constants $a, b \geq 0$. We can lower bound the progress as

$$\begin{aligned} &D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\ &\qquad \geq \frac{2\, \boldsymbol{r} \cdot \boldsymbol{x}^t \, \log(1 - \eta\, (\hat{y}^t - y^t)\, X)}{X} + \bigl(2\, \eta\, (\hat{y}^t - y^t) - \eta^2\, (\hat{y}^t - y^t)^2\, X\bigr)\, (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t. \end{aligned}$$

Denoting by $s \coloneqq \boldsymbol{r} \cdot \boldsymbol{x}^t$ and $p \coloneqq (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t$, it suffices to show that $G(p, \hat{y}, y, s) \leq 0$ where (omitting the superscript t)

$$G(p, \hat{y}, y, s) = -\frac{2s \log(1 - \eta\, (\hat{y} - y) X)}{X} - \bigl(2\, \eta\, (\hat{y} - y) - \eta^2\, (\hat{y} - y)^2\, X\bigr)\, p - a(y - \hat{y})^2 + b(y - s)^2.$$

It suffices to show the result for $p = \hat{y}$. The function $G(\hat{y}, \hat{y}, y, s)$ is maximized for $s = y - \log(1 - \eta(\hat{y} - y)X)/(Xb)$. Setting $\eta = b/(1 + 2XYb)$, the inequality holds for any

8. Note that the squared loss $(\boldsymbol{w} \cdot \boldsymbol{x} - y)^2$ is convex in $\boldsymbol{w}$, but $(\boldsymbol{u} \odot \boldsymbol{u} \cdot \boldsymbol{x} - y)^2$ is not convex in $\boldsymbol{u}$.

$a \le b/(1+2XYb)$. Setting $b = 1/(XY)$, the bound is achieved for $\eta = a = 1/(3XY)$. ■

Algorithm 5 EGU Linear Regression

Parameters initial weight $\boldsymbol{w}^1 \in \mathbb{R}^n_{\ge 0}$, learning rate $\eta > 0$

for $t = 1$ to T **do**

Receive instance $\boldsymbol{x}^t \in [0, X]^n$

$$\text{Predict } \hat{y}^t = \begin{cases} \boldsymbol{w}^t \cdot \boldsymbol{x}^t & \text{if } \boldsymbol{w}^t \cdot \boldsymbol{x}^t \le Y \\ Y & \text{otherwise} \end{cases}$$

Receive label y^t and update:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t \odot \exp\big(-2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t\big)$$

Algorithm 6 Reparam. EGU Lin. Reg.

Parameters initial weight $\boldsymbol{u}^1 \in \mathbb{R}^n$, learning rate $\eta > 0$

for $t = 1$ to T **do**

Receive instance $\boldsymbol{x}^t \in [0, X]^n$

$$\text{Predict } \hat{y}^t = \begin{cases} \boldsymbol{u}^t \odot \boldsymbol{u}^t \cdot \boldsymbol{x}^t & \text{if } \boldsymbol{u}^t \odot \boldsymbol{u}^t \cdot \boldsymbol{x}^t \le Y \\ Y & \text{otherwise} \end{cases}$$

Receive label y^t and update:

$$\boldsymbol{u}^{t+1} = \boldsymbol{u}^t - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{u}^t \odot \boldsymbol{x}^t$$

Alternatively, the (normalized) EG algorithm maintains a probability vector $\boldsymbol{w}^t \in \Delta^{n-1}$ as its weight vector. The update for EG is the same as EGU, except for a multiplicative normalization:

$$\boldsymbol{w}^{t+1} = \frac{\boldsymbol{w}^t \odot \exp\big(-2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t\big)}{\sum_i w_i^{t+1} \exp\big(-2\,\eta\,(\hat{y}^t - y^t)\,x_i^t\big)}\,.$$

The following theorem from (Kivinen and Warmuth, 1997) expresses the worst-case bound of running EG for linear regression.

Theorem 8 (EG linear regression (Kivinen and Warmuth, 1997)) *Let $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ be any sequences such that $\max_i x_i^t - \min_i x_i^t \le R$ for some $R \ge 0$. Then for any comparator $\boldsymbol{r} \in \Delta^{n-1}$, the EG algorithm with learning rate $\eta = 2/(3R^2)$ and arbitrary start point $\boldsymbol{w}^1 \in \Delta^{n-1}$ satisfies the total loss bound*

$$\sum_{t=1}^T (y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t)^2 \le \frac{3}{2}\Big(\sum_{t=1}^T (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 + R^2 D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1)\Big)\,. \tag{17}$$

Furthermore, let L and D be constants such that $\sum_t (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 \le L$ and $D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1) \le D$ and let

$$\eta = \frac{2\sqrt{D}}{R\sqrt{2L} + R^2\sqrt{D}}\,. \tag{18}$$

Then we have

$$\sum_{t=1}^T (y^t - \boldsymbol{w}^t \cdot \boldsymbol{x}^t)^2 \le \sum_{t=1}^T (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 + R\sqrt{2LD} + \frac{R^2}{2} D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{w}^1)\,. \tag{19}$$

Using (10), the reparameterized EG update for linear update can be written as

$$\boldsymbol{u}^{t+1} = \frac{\boldsymbol{u}^t - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{u}^t \odot \boldsymbol{x}^t}{\|\boldsymbol{u}^t - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{u}^t \odot \boldsymbol{x}^t\|_2}\,. \tag{20}$$

The algorithm for EG on linear regression is similar to Algorithm 6, except the initial weight $\boldsymbol{u}^1$ satisfies $\|\boldsymbol{u}^1\|_2 = 1$ and the update is replaced with (20). In the following, we

show the regret for the reparameterized EG update for linear regression. For simplicity, we first consider the case where the input $\boldsymbol{x} \in [0, X]^n$ for some $X > 0$.

Theorem 9 (Reparameterized EG linear regression) *Let $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^T$ be any sequences such that $\boldsymbol{x}^t \in [0, X]^n$ for some $X \geq 0$. Then for any comparator $\boldsymbol{r} \in \Delta^{n-1}$, the Reparameterized EG algorithm with learning rate $\eta = 1/(3X^2)$ and arbitrary start point $\boldsymbol{u}^1 \in \boldsymbol{R}^n$ such that $\|\boldsymbol{u}^1\|_2 = 1$ satisfies the total loss bound*

$$\sum_{t=1}^{T} (y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t)^2 \leq 3 \left(\sum_t (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 + X^2 D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) \right). \tag{21}$$

Furthermore, let L and D be constants such that $\sum_t (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 \leq L$ and $D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) \leq D$ and let

$$\eta = \frac{2\sqrt{D}}{X\sqrt{2L} + X^2\sqrt{D}}. \tag{22}$$

Then we have

$$\sum_{t=1}^{T} (y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t)^2 \leq \sum_{t=1}^{T} (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 + 2X\sqrt{2\,LD} + 2X^2\, D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{w}^1). \tag{23}$$

The proof is given in the appendix. In the following, we claim a bound for the reparameterized EG for the more general case where $\boldsymbol{x}^t \in [-X, X]^n$ for all t. We provide a proof sketch for the claim in the appendix. Our partial proof shows strong evidence for the existence of such regret bound. However, finding an analytical proof by simplifying the final form is left for future work.

Claim 1 (More general bound for reparameterized EG linear regression) *For the same setting as in Theorem 9 but a more general case where $\boldsymbol{x}^t \in [-X, X]^n$ for all t, the reparameterized EG algorithm achieves the same bound (23) in which X is replaced by $2X$.*

6. Lower Bounds and Simulations

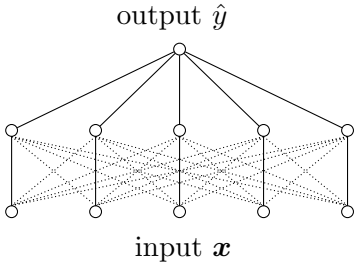
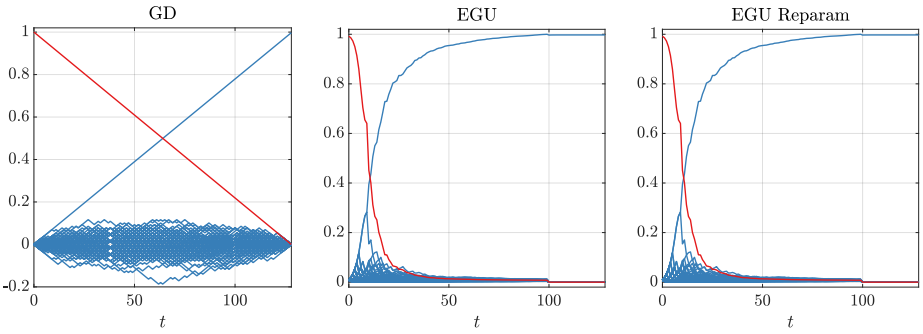


Figure 2: Complete two-layer linear network. The dotted weights are initialized to zero.

Linear lower bounds for GD have been studied in the previous work (Kivinen et al., 1997; Vishwanathan and Warmuth, 2005) for the hinge and squared loss. These lower bounds are all based on the "Hadamard problem": the instances are the n rows of the n-dimensional Hadamard matrix in random order and the target is one of the columns.[9] Thus target is a unit weight vector selecting the right column. For example, for linear regression, as online GD passes over the n examples, it brings up the weight of the right column slowly and the average squared loss on all n instances decreases linearly (Figure 3(a)). Multiplicative updates such as EGU bring up the target weight dramatically faster and

9. Similar behavior is observed if a random ± 1 matrix is used.



(a) (b) (c)

Figure 3: GD, EGU, and Reparameterized EGU on the online Hadamard problem ($n=128$): at round t, we train until consistency on the past t examples. For $1 \leq t \leq 128$, all 128 weights are shown in blue and the average loss over all 128 examples is shown in red.

the average squared loss decays essentially after $\log n$ examples (Figure 3(b)). Surprisingly, GD on the u_i's of the sparse linear network of Figure 1 has visually identical weights[10] (Figure 3(c)) and average loss trajectories as EGU (Figure 3(b)). Furthermore, if we run GD on the fully-connected two-layer network of Figure 2 with the dotted weights initialized to zero, then the combined linear weights of both layers again behave as when a simple linear neuron is trained with GD (Appendix D). This behavior seems to be related to the observation that GD focuses on keeping the highest weight as small as possible and uses the additional weights to overfit.

7. Open Problems

There are a number of technical open problems associated with the current paper. A complete analytical proof is still needed for the regret of the reparameterized EG when the instance domain is two-sided, i.e. $\boldsymbol{x}^t \in [-X, X]^n$. Note also that regret bounds for the two-sided domain do not exist for the original and reparameterized EGU. Experimentally however, the two updates show the typical $\log n$ behavior of multiplicative updates. For the lower bound discussion, a next step would be to show that for the Hadamard problem, GD on a two-layer linear network with any initialization has the typical linear decaying loss. Also in this paper, we were able to repeat the regret bounds for the Reparameterized Hedge algorithm (i.e. when the loss is the dot loss and the loss components lie in $[0, 1]$). However, so far we were not able to repeat $\mathcal{O}(\log n)$ regret bounds (Vovk, 1990; Haussler et al., 1998) for a reparameterized version of the multiplicative expert algorithm when the loss is the dot loss and the loss components are the squared loss (a.k.a. the Brier scores). We conjecture that $\mathcal{O}(\log n)$ regret bounds are not possible for the reparameterized version of the expert algorithm when the loss components are log losses. The reason is the unboundedness of the log loss. Lastly, there are many open problems regarding neural networks. For example, are reparameterized multiplicative updates useful for training large networks and how the optimization routines such as AdaGrad and Adam interact with the new reparameterized updates?

Acknowledgement We thank Zakaria Mhammedi for valuable feedback. Also E. Amid was partially supported by the NSF grant IIS 1546459 and a gift grant from the Intel Corp.

10. The trajectories are very close, but not numerically equal.

References

Ethan Akin. *The geometry of population genetics*, volume 31 of *Lecture Notes in Biomathematics*. Springer-Verlag, Berlin-New York, 1979.

Ehsan Amid and Manfred K. Warmuth. Reparameterizing mirror descent as gradient descent. *arXiv preprint arXiv:2002.10487*, 2020.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning (ICML)*, 2018.

Nicolo Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2-3):321–352, 2007.

Michal Derezinski and Manfred K. Warmuth. The limits of squared Euclidean distance regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2807–2815, 2014.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

C. Gentile and M. K. Warmuth. Hinge loss and average margin. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1998.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6151–6159, 2017.

David Haussler, Jyrki Kivinen, and Manfred K Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44 (5):1906–1925, 1998.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

Jyrki Kivinen, Manfred K Warmuth, and Peter Auer. The Perceptron algorithm versus Winnow: linear versus logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 97(1-2):325–343, 1997.

Jyrki Kivinen, Manfred K Warmuth, and Babak Hassibi. The p-norm generalization of the LMS algorithm for adaptive filtering. *IEEE Transactions on Signal Processing*, 54(5): 1782–1793, 2006.

N Littlestone and MK Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

William H Sandholm. *Population games and evolutionary dynamics*. MIT Press, 2010.

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

Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2968–2979, 2019.

S.V.N. Vishwanathan and M.K. Warmuth. Leaving the span. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, 2005.

Volodimir G Vovk. Aggregating strategies. In *Proceedings of the third annual workshop on Computational learning theory*, pages 371–386, 1990.

Manfred K. Warmuth. Winnowing subspaces. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 999–1006, 2007.

Blake Woodworth, Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Kernel and deep regimes in overparametrized models. *arXiv preprint arXiv:1906.05827*, 2019.

Appendix A. Proof of Theorem 7

Proof We first establish a lower bound of the form,

$$a\,(y^t-\hat{y}^2)^2 - b\,(y^t - \boldsymbol{r}\cdot\boldsymbol{x}^t)^2 \le D_{\text{RE}}(\boldsymbol{r},\boldsymbol{u}^t\odot\boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{r},\boldsymbol{u}^{t+1}\odot\boldsymbol{u}^{t+1})\,, \tag{24}$$

on the progress of the algorithm towards the comparator $\boldsymbol{r}$, for some constants $0 \le a, b$. Assuming that $\eta \le 1/(2XY)$, we have $(1-\eta\,(\hat{y}^t-y^t)\,x_i^t) \ge 0$ for all i. Thus, we can lower bound the progress as

$$\begin{aligned}
&D_{\text{RE}}(\boldsymbol{r},\boldsymbol{u}^t\odot\boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{r},\boldsymbol{u}^{t+1}\odot\boldsymbol{u}^{t+1}) \\
&\quad = 2\,\boldsymbol{r}\cdot\log(\boldsymbol{1}-\eta\,(\hat{y}^t-y^t)\,\boldsymbol{x}^t) - \log\left(\boldsymbol{w}^t\cdot(\boldsymbol{1}-2\,\eta\,(\hat{y}^t-y^t)\,\boldsymbol{x}^t+\eta\,(\hat{y}^t-y^t)^2\,\boldsymbol{x}^t\odot\boldsymbol{x}^t-\boldsymbol{1}\right).
\end{aligned}$$

Applying the inequalities $\log(1-\eta\,(\hat{y}^t-y^t)x_i^t) \ge \frac{x_i^t\log(1-\eta\,(\hat{y}^t-y^t)X)}{X}$ and $\log(1+x)\le x$ for $x>-1$ and using the fact that $(x_i^t)^2 \le x_i^t\,X$, we have

$$\begin{aligned}
&D_{\text{RE}}(\boldsymbol{r},\boldsymbol{u}^t\odot\boldsymbol{u}^t) - D_{\text{RE}}(\boldsymbol{r},\boldsymbol{u}^{t+1}\odot\boldsymbol{u}^{t+1}) \\
&\quad \ge \frac{2\,\boldsymbol{r}\cdot\boldsymbol{x}^t\,\log(1-\eta\,(\hat{y}^t-y^t)\,X)}{X} + \left(2\,\eta\,(\hat{y}^t-y^t) - \eta^2\,(\hat{y}^t-y^t)^2\,X\right)(\boldsymbol{u}^t\odot\boldsymbol{u}^t)\cdot\boldsymbol{x}^t\,.
\end{aligned}$$

Denoting by $s \coloneqq \boldsymbol{r}\cdot\boldsymbol{x}^t$ and $p \coloneqq (\boldsymbol{u}^t\odot\boldsymbol{u}^t)\cdot\boldsymbol{x}^t$, it suffices to show that $G(p,\hat{y},y,s)\le 0$ where (omitting the superscript t)

$$G(p,\hat{y},y,s) = -\frac{2s\log(1-\eta\,(\hat{y}-y)X)}{X} - \left(2\,\eta\,(\hat{y}-y) - \eta^2\,(\hat{y}-y)^2\,X\right)p + a(y-\hat{y})^2 - b(y-s)^2\,.$$

Recall that the prediction $\hat{y}^t$ of the reparameterized EGU is given by $\hat{y}^t = p = (\boldsymbol{u}^t\odot\boldsymbol{u}^t)\cdot\boldsymbol{x}^t$ if $p\le Y$ holds; otherwise $\hat{y}^t = Y$. Thus, we need to show $G(p,\hat{y},y,s)\le 0$ for two cases: when $\hat{y}=p$, and for $0\le\hat{y}=Y<p$. Recall that by the assumption $0\le y\le Y$ and $\eta\le 1/(2XY)$. Therefore, $G(p,\hat{y},y,s)$ is non-increasing in p for $\hat{y}\ge y$. Hence, the condition $G(p,\hat{y},y,s)\le 0$ for $\hat{y}=Y<p$ is satisfied if $G(Y,Y,y,s)\le 0$ holds. Thus, it suffices to show the result for $0\le\hat{y}=p\le Y$.

For fixed $\hat{y}$ and y, the function $G(\hat{y},\hat{y},y,s)$ is maximized for

$$s = y - A/(Xb)\,,\quad \text{where}\quad A \coloneqq \log(1-\eta(\hat{y}-y)X)\,.$$

Plugging in this value, we have $H(\hat{y},y) \coloneqq G(\hat{y},\hat{y},y,y-A/(Xb))$ where

$$H(\hat{y},y) = -2\,\eta\,(\hat{y}-y)\hat{y} + \eta^2\,(\hat{y}-y)^2\,\hat{y}\,X + a\,(\hat{y}-y)^2 + A^2/(bX^2) - 2\,A\,y/X\,.$$

In order to obtain the bound (24), we show that $H(\hat{y},y)\le 0$ holds for the choice of $\eta = b/(1+2XYb)$ and $a = b/(1+2XYb)$. Substituting these values for η and a, we have $H(y,y) = \frac{\partial H(\hat{y},y)}{\partial y}\big|_{\hat{y}=y} = 0$ and Furthermore,

$$\frac{\partial^2 H(\hat{y},y)}{\partial y^2}\Big|_{\hat{y}=y} = -\frac{4b^2X(Y-y)}{(1+2XYb)^2} \le 0 \quad \text{for all} \quad 0\le y\le Y\,.$$

Thus, it suffices to show that $y = \hat{y}$ is the only maximum of the function. For this, we first write

$$\frac{\partial^2 H(\hat{y}, y)}{\partial y^2} = -\frac{1}{\left(\frac{1}{2} + X\big(Y + \frac{y}{2} - \frac{\hat{y}}{2}\big)b\right)^2 \left(\frac{1}{2} + XYb\right)^2} \times \Bigg(\Big(\frac{1}{2} + XYb\Big)^2 \log\Big(1 + \frac{X(y-\hat{y})b}{1+2XYb}\Big)b + \Big(Y^3 - \big(\frac{\hat{y}+y}{2}\big)Y^2 + \big(\frac{\hat{y}^2 - y^2}{4}\big)Y + \frac{\hat{y}(\hat{y}-y)}{8}\Big)b^4 X^3 + \Big(Y^2 - \big(\frac{\hat{y}+y}{2}\big)Y + \frac{\hat{y}^2 - y^2}{8}\Big)b^3 X^2 + \Big(\frac{Y}{4} + \frac{\hat{y}-y}{8}\Big)b^2 X \Bigg) .$$

Using the inequality $\log(1+x) \le x$ for $x > -1$, we can upper bound the log-term as

$$\log\Big(1 + \frac{X(y-\hat{y})b}{1+2XYb}\Big)b \le \frac{X(y-\hat{y})b^2}{1+2XYb} ,$$

and write the new function as $Q(\hat{y}, y)$ such that $\frac{\partial^2 H(\hat{y},y)}{\partial y^2} \le Q(\hat{y}, y)$. It is trivial to check that the derivative

$$\frac{\partial Q(\hat{y}, y)}{\partial y} = \frac{3\Big(\Big(Y + \frac{y}{6} - \frac{\hat{y}}{2}\Big)\Big(Y + \frac{y}{2} - \frac{\hat{y}}{2}\Big)bX + \frac{Y}{2} + \frac{y}{3} - \frac{\hat{y}}{2}\Big)X^2 b^3}{2\Big(\frac{1}{2} + X\Big(Y + \frac{y}{2} - \frac{\hat{y}}{2}\Big)b\Big)^4} \ge 0 ,$$

for any $0 \le y \le Y$ and $0 \le \hat{y} \le Y$. Hence, it remains to check that $Q(\hat{y}, Y) \le 0$ holds:

$$Q(\hat{y}, Y) = -\frac{1}{9\Big(\frac{1}{3} + X\big(Y - \frac{\hat{y}}{3}\big)b\Big)^3 \big(\frac{1}{2} + XYb\big)^2} \times$$

$$(Y-\hat{y})Xb^2\Big(\frac{1}{3} + \big(Y - \frac{\hat{y}}{3}\big)(Y-\hat{y})\big(Y - \frac{\hat{y}}{2}\big)b^3X^3 + \frac{5\big(Y^2 - \frac{3}{5}Y\hat{y} + \frac{2}{15}\hat{y}^2\big)b^2X^2}{2} + \frac{5\big(Y - \frac{\hat{y}}{5}\big)bX}{3}\Big) \le 0 ,$$

which holds for any $0 \le \hat{y} \le Y$. This implies $\frac{\partial^2 H(\hat{y},y)}{\partial y^2} \le Q(\hat{y}, y) \le 0$ for any $0 \le \hat{y} \le Y$ and $0 \le y \le Y$.

The remainder of the proof follows similarly to (Kivinen and Warmuth, 1997). Specifically, by setting setting $b = c/(XY)$, we obtain

$$\sum_{t=1}^{T} (y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t) \cdot \boldsymbol{x}^t)^2 \le \big(1 + 2\,c\big) \sum_{t=1}^{T} (y^t - \boldsymbol{r} \cdot \boldsymbol{x}^t)^2 + \big(2 + \frac{1}{c}\big)XY D_{\text{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1) .$$

Setting $c = 1$, the bound in (13) is achieved for $\eta = a = 1/(3XY)$. Using the values L and D and tuning for c achieves (15) for the choice of η as in (14).

■

Appendix B. Proof of Theorem 9

Similar to the proof of Theorem 7, we establish a lower bound of the form

$$a\,(y^t - \hat{y}^t)^2 - b\,(y^t - \boldsymbol{r}\cdot\boldsymbol{x}^t)^2 \le D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1})\,, \tag{25}$$

on the progress of the algorithm towards the comparator $\boldsymbol{r} \in \Delta^{n-1}$, and for some constants $a, b \ge 0$. Assuming that $\eta \le 1/(2X^2)$, we have $(1 - \eta\,(\hat{y}^t - y^t)\,x_i^t) \ge 0$ for all i. Thus, we can lower bound the progress as

$$\begin{aligned} &D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\ &\quad = 2\,\boldsymbol{r}\cdot\log(\mathbf{1} - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t) - \log\left(\boldsymbol{w}^t\cdot(\mathbf{1} - 2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t + \eta\,(\hat{y}^t - y^t)^2\,\boldsymbol{x}^t \odot \boldsymbol{x}^t\right). \end{aligned}$$

Applying the inequalities $\log(1 - \eta\,(\hat{y}^t - y^t)x_i^t) \ge \frac{x_i^t \log(1 - \eta\,(\hat{y}^t - y^t)X)}{X}$ and $-\log(1-x) \ge x$ for $0 \le x \le 1$ and using the fact that $(x_i^t)^2 \le x_i^t\,X$, we have

$$\begin{aligned} &D_{\mathrm{RE}}(\boldsymbol{r},\, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\ &\qquad \ge \frac{2\,\boldsymbol{r}\cdot\boldsymbol{x}^t\,\log(1 - \eta\,(\hat{y}^t - y^t)\,X)}{X} + \left(2\,\eta\,(\hat{y}^t - y^t) - \eta^2\,(\hat{y}^t - y^t)^2\,X\right)\hat{y}^t\,. \end{aligned}$$

Denoting by $s \coloneqq \boldsymbol{r}\cdot\boldsymbol{x}^t$, it suffices to show that $G(\hat{y}, y, s) \le 0$ where (omitting the superscript t)

$$G(\hat{y}, y, s) = -\frac{2r\log(1 - \eta\,(\hat{y} - y)X)}{X} - \left(2\,\eta\,(\hat{y} - y) - \eta^2\,(\hat{y} - y)^2\,X\right)\hat{y} + a(y - \hat{y})^2 - b(y - s)^2\,.$$

For fixed $\hat{y}$ and y, the function $G(\hat{y}, y, s)$ is maximized for

$$s = y - A/(Xb)\,, \quad \text{where} \quad A \coloneqq \log(1 - \eta(\hat{y} - y)X)\,.$$

Plugging in this value, we have $H(\hat{y}, y) \coloneqq G(\hat{y}, y, y - A/(Xb))$ where

$$H(\hat{y}, y) = -2\,\eta\,(\hat{y} - y)\hat{y} + \eta^2\,(\hat{y} - y)^2\,\hat{y}\,X + a\,(\hat{y} - y)^2 + A^2/(bX^2) - 2\,A\,y/X\,.$$

It is easy to verify that $H(y, y) = \frac{\partial H(\hat{y}, y)}{\partial y}\big|_{\hat{y}=y} = 0$. Moreover, notice that

$$\frac{\partial^2 H(\hat{y}, y)}{\partial y^2}\Big|_{\hat{y}=y} = \frac{(4Xy + 2)\,\eta^2 - 4\,b\,\eta + 2\,ab}{b}\,,$$

which is minimized at $y = X$ for $\eta = b/(1 + 2\,X^2 b)$. Plugging in this value, we have $\frac{\partial^2 H(\hat{y}, y)}{\partial y^2} \le 0$ for $a \le b/(1 + 2\,X^2 b)$. Now there remains to show that $\hat{y} = y$ is the only maximum of the function $H(\hat{y}, y)$. This is easily verified by plugging in $a = b/(1 + 2\,X^2 b)$ and observing

$$\begin{aligned} \frac{\partial^2 H(\hat{y}, y)}{\partial y^2} = -\frac{b}{\left(\frac{1}{2} + X^2 b\right)^2\left(\frac{1}{2} + \left(X + \frac{y}{2} - \frac{\hat{y}}{2}\right)Xb\right)^2} \times \\ \left(\frac{1}{2}\left(\frac{1}{2} + X^2 b\right)^2 \log\left(1 + \frac{(y - yh)\,Xb}{1 + 2X^2 b}\right)\right. \\ + Xb\left(\left(X^3 - \left(\frac{y + yh}{2}\right)X^2 + \left(\frac{\hat{y}^2 - y^2}{4}\right)X - \frac{\hat{y}(y - \hat{y})^2}{8}\right)X^2 b^2\right. \\ \left.\left. + X^2 b\left(X^2 - \left(\frac{y + yh}{2}\right)X + \left(\frac{\hat{y}^2 - y^2}{8}\right)\right) + \frac{X^2}{4} - \frac{y + \hat{y}}{8}X\right) \le 0\,, \end{aligned}$$

for $0 \le y, \hat{y} \le X$ and

$$\frac{\partial^2 H(\hat{y}, y)}{\partial y^2}\Big|_{\hat{y}=y} = -\frac{4\,b^2 X(X-y)}{(1+2\,X^2 b)^2}\,.$$

Finally, setting $b = c/(X^2)$ for some $c > 0$, we have

$$\sum_{t=1}^{T}(y^t - (\boldsymbol{u}^t \odot \boldsymbol{u}^t)\cdot \boldsymbol{x}^t)^2 \le \big(1+2\,c\big)\sum_{t=1}^{T}(y^t - \boldsymbol{r}\cdot\boldsymbol{x}^t)^2 + \big(2+\frac{1}{c}\big)XY D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^1 \odot \boldsymbol{u}^1)\,.$$

Setting $c = 1$ establishes the first bound. Similarly, optimizing for c yields

$$c = \frac{X\sqrt{D}}{\sqrt{2L}}\,.$$

For this choice of c and η as in (22), we obtain the second bound.

Appendix C. Proof Sketch of Claim 1

We can lower bound the progress of the algorithm as (assuming $\eta \le 1/(2\,X^2)$),

$$\begin{aligned}
&D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^t \odot \boldsymbol{u}^t) - D_{\mathrm{RE}}(\boldsymbol{r}, \boldsymbol{u}^{t+1} \odot \boldsymbol{u}^{t+1}) \\
&\quad = 2\,\boldsymbol{r}\cdot\log(\mathbf{1} - \eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t) - \log\big(\boldsymbol{w}^t\cdot(\mathbf{1} - 2\,\eta\,(\hat{y}^t - y^t)\,\boldsymbol{x}^t + \eta\,(\hat{y}^t - y^t)^2\,\boldsymbol{x}^t \odot \boldsymbol{x}^t\big) \\
&\quad \ge \frac{(s+X)}{X}\,\log\Big(\frac{1-\eta\,(\hat{y}^t - y^t)\,X}{1+\eta\,(\hat{y}^t - y^t)\,X}\Big) + 2\,\log\big(1+\eta\,(\hat{y}^t - y^t)\,X\big) \\
&\qquad - \log\big(1 - 2\,\eta\,(\hat{y}^t - y^t)\,\hat{y}^t + \eta^2\,(\hat{y}^t - y^t)^2\,X^2\big)\,,
\end{aligned}$$

where $s = \boldsymbol{r}\cdot\boldsymbol{x}^t$. For fixed y and $\hat{y}$, the lower bound can be maximized for (omitting t)

$$s = y - \frac{1}{Xb}\,\log\Big(\frac{1-\eta\,(\hat{y}-y)\,X}{1+\eta\,(\hat{y}-y)\,X}\Big)\,.$$

Thus, plugging back for s and introducing the new variable $\delta = y - \hat{y}$, it suffices to show that the function $G(a, b, \eta, \delta, \hat{y}) \le 0$ for the choices of a, b, and η in the claim, and for all $-X \le \hat{y} \le X$, where

$$\begin{aligned}
G(a,b,\eta,\delta,\hat{y}) \coloneqq \log(1 + 2\,\eta\,\delta\,\hat{y} + \eta^2\,\delta^2 X^2) + \frac{1}{4X^2 b}\,\log\Big(\frac{1+\eta\delta X}{1-\eta\delta X}\Big)^2 \\
- \frac{1}{X}\,(X + \delta + \hat{y})\,\log\Big(\frac{1+\eta\delta X}{1-\eta\delta X}\Big) - 2\,\log(1-\eta\,\delta X) + a\,\delta^2\,.
\end{aligned}$$

For a fixed δ, this function is maximized at

$$\hat{y}_{\mathrm{opt}} = \frac{X}{\log\big(\frac{1+\eta\delta X}{1-\eta\delta X}\big)} - \frac{1}{2}\eta\delta X^2 \quad \text{since} \quad \frac{\partial^2}{\partial \hat{y}^2}\,G(a,b,\eta,\delta,\hat{y})\big|_{\hat{y}=\hat{y}_{\mathrm{opt}}} = -\frac{\log\big(\frac{1+\eta\delta X}{1-\eta\delta X}\big)^2}{X}\,.$$

Substituting this value for $\hat{y}$, we need to show $H(a,b,\eta) := G(a,b,\eta,\delta,\hat{y}_{\text{opt}}) \leq 0$ where

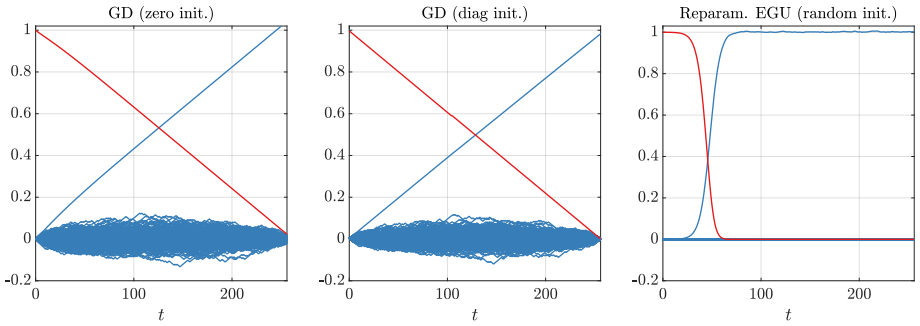
$$H(a,b,\eta) = \log\left(\frac{1+\eta\delta X}{1-\eta\delta X}\right)\left(\frac{1}{4X^2 b}\log\left(\frac{1+\eta\delta X}{1-\eta\delta X}\right) + \frac{\eta\delta X}{2} - \frac{\delta}{X} + \frac{1}{2\eta\delta X}\right) + \log\left(\frac{\eta\delta X}{\log\left(\frac{1+\eta\delta X}{1-\eta\delta X}\right)}\right) + a\,\delta^2 - \log(1-\eta^2\delta^2X^2) + \log(2) - 1\,.$$

This function can be simplified further by substituting $a = \eta = b/(1+2X^2b)$. Also, we can introduce two new variables by defining $b = c/X^2$ for some $c > 0$ and $\delta = p\,X$ for $-2 \leq p \leq 2$. As final step of the proof, there remains to show that the function

$$K(c,p) := \log\left(\frac{1+(2+p)c}{1+(2-p)c}\right)\left(\frac{1}{4c}\log\left(\frac{1+(2+p)c}{1+(2-p)c}\right) + \frac{pc}{2(2c+1)} - p + \frac{2c+1}{2pc}\right) + \log\left(\frac{pc}{(2c+1)\log\left(\frac{1+(2+p)c}{1+(2-p)c}\right)}\right) + \frac{p^2c}{2c+1} - \log\left((1+2c)^2 - p^2\right) + \log(2) - 1 \leq 0\,,$$

for all values of $c > 0$ and $-2 \leq p \leq 2$.

Appendix D. Behavior of GD and Reparameterized EGU with Different Initializations



(a) (b) (c)

Figure 4: Results of two-layer linear networks on the online Hadamard problem ($n=128$). We perform one pass over the examples. The product of the weights of the first and the second layer (128 weights in total) are shown in blue and the average loss over all 128 examples is shown in red. Results of the two-layer network of Figure 2 using GD where the first layer weights are (a) initialized to zero, (b) initialized uniformly on both diagonals, (c) results of the sparse network of Figure 1 where the weights are initialized randomly.

Here, we discuss two interesting observations on the two-layer networks in Figure 1 and Figure 2. We extend the experiments on learning a column of the Hadamard matrix to two-layer networks. We consider the online Hadamard problem ($n=128$) where we perform one pass over the examples. The results are shown in Figure 4. The figures stress two points about training two-layer linear networks with GD. First, if all the weights in the first layer are initialized to zero (Figure 4(a)), then the product of the weight matrix of the

first layer times the weight vector of the second layer behaves qualitatively the same as the weights of a single linear neuron trained with GD (compare to Figure 3(a)). The same is true if the dotted weights in Figure 2 are initialized to zero and the remaining weights (solid diagonal weights of the first layer and the weights of the second layer) are all set to uniform (Figure 4(b)). This is essentially the same initialization used for the reparameterized EGU algorithm, except in this case, the dotted connections are not removed.

Second, the initialization of the sparse linear network (Figure 1) is rather robust experimentally. Equal initializations or random initializations of both diagonals all make the combined weight behave qualitatively like EGU on a single neuron (Figure 4(c) compared to Figure 3(b)). In (Arora et al., 2018) it was observed that multiplying the weight vector $\boldsymbol{w}$ of a linear neuron by a scalar weight w' and training both with GD behaves similarly to a momentum update. We experimentally show that even when the weights of the sparse network are initialized randomly, then training it with GD shows a drastically different dynamic than training a single neuron with GD. Notice that in the sparse network, each weight on the bottom layer is now multiplied by a separate scalar weight (on the top layer), instead of one common weight as in (Arora et al., 2018).

Finally, note that if we initialize any consecutive weights (along the path from the input to the output) of the sparse linear network in Figure 1 to equal values, then these weights will remain equal throughout the training using GD (since the two edges will have equal gradients at any time). Thus, keeping both weights equal during training does not require any additional constraints on the network.