

# Noise misleads rotation invariant algorithms on sparse targets

**Manfred K. Warmuth**

MANFRED@GOOGLE.COM

*Google Inc.*

**Wojciech Kotłowski**

WKOTLOWSKI@CS.PUT.POZNAN.PL

*Institute of Computing Science, Poznań University of Technology, Poznań, Poland*

**Matt Jones**

MCJ@COLORADO.EDU

*University of Colorado Boulder, Colorado, USA*

**Ehsan Amid**

EAMID@GOOGLE.COM

*Google Inc.*

## Abstract

It is well known that the class of rotation invariant algorithms are suboptimal even for learning sparse linear problems when the number of examples is below the “dimension” of the problem. This class includes any gradient descent trained neural net with a fully-connected input layer (initialized with a rotationally symmetric distribution). The simplest sparse problem is learning a single feature out of  $d$  features. In that case the classification error or regression loss grows with  $1 - k/d$  where  $k$  is the number of examples seen. These lower bounds become vacuous when the number of examples  $k$  reaches the dimension  $d$ .

Nevertheless we show that when noise is added to this sparse linear problem, rotation invariant algorithms are still suboptimal after seeing  $d$  or more examples. We prove this via a lower bound for the Bayes optimal algorithm on a rotationally symmetrized problem. We then prove much lower upper bounds on the same problem for simple non-rotation invariant algorithms. Finally we analyze the gradient flow trajectories of many standard optimization algorithms in some simple cases and show how they veer toward or away from the sparse targets.

We believe that our trajectory categorization will be useful in designing algorithms that can exploit sparse targets and our method for proving lower bounds will be crucial for analyzing other families of algorithms that admit different classes of invariances.

**Keywords:** rotation invariance, feed forward nets, lower bounds, multiplicative updates, sparsity.

Any gradient descent trained neural net with a fully-connected input layer is rotation invariant when initialized with a rotationally symmetric distribution. The reason is that if the input instances are rotated, then weights rotate in the same way and therefore the dot products (logits) feeding into the non-linearities of the first layer remain unchanged. It is well known that rotation invariant algorithms are fundamentally inferior when learning certain sparse linear problems (Warmuth and Vishwanathan, 2005; Ng, 2004; Li et al., 2021; Warmuth et al., 2021). These bounds are surprising because they hold for such a general class of algorithms and are complemented by simple non-rotation invariant algorithms that need exponentially fewer examples to achieve the same accuracy. However, these lower bounds are not practically relevant because they all assume that the number of examples is less than the dimension of the problem.

The simplest sparse linear problem is learning one out of  $d$  features. When the instances are for example the  $d$  orthogonal rows of a Hadamard matrix and a rotation invariant algorithm is given  $k \leq d$  training instances<sup>1</sup> then its generalization error on all  $d$  instances is at least  $1 - k/d$ . There

1. Learning is harder for sampling with replacement. However lower bounds are stronger for sampling without replacement and the boundary of getting a full rank instance set is clearer. We present bounds for both types of models.

are classification (Ng, 2004; Li et al., 2021) and regression versions of this problem (Warmuth and Vishwanathan, 2005; Warmuth et al., 2021) but the lower bound is essentially the same, proven with different techniques. However, the lower bounds become vacuous when  $k \geq d$ , which is the case in most relevant settings. For example linear regression (which is rotation invariant) finds the unique consistent feature after seeing a noise-free training set of full rank. In contrast this paper provides lower bounds on the loss of any rotation invariant algorithm when the sample size (without replacement) exceeds the input dimension of the problem. Now any weight vector (including sparse ones) can be expressed as a linear combination of the instances. The key point of our paper is that, when the noise is added to the sparse target, rotation invariant algorithms still produce poor solutions compared to simple non-invariant algorithms.

Our lower bound technique creates a Bayesian setup where the learning is presented with a randomly rotated version of the input instances. We prove a lower bound on the Bayes-optimal algorithm for this case, and we prove any rotational invariant algorithm on the original problem can do no better than the Bayes-optimal algorithm on the randomly rotated problem.<sup>2</sup> We then show that there are trivial non-rotationally invariant algorithms that can learn noisy sparse linear much more efficiently: multiplicative updates on a linear neuron or gradient descent on a ‘spindly’ two-layer linear net where every linear weight  $w_i$  is replaced by a product of parameter  $u_i v_i$  (Figure 1). Finally we also prove the same upper bound for a novel “priming method” that was conjectured to learn sparse linear problems in (Warmuth and Amid, 2023): First find the linear least squares solution  $w$  and then after reweighing the  $i$ th feature with  $w_i$ , apply tuned ridge regression.

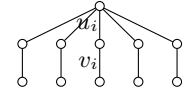


Figure 1: Spindly.

We are not interested in linear problems per se, but any fancy non-linear model should at least be able to handle the linear case. We visualize how the performance gap between rotationally invariant and non-invariant algorithms arises by computing their trajectories in weight space over the course of learning. Previous work has focused on how rotational symmetry can be broken adding a L1 penalty to the loss (Ng, 2004; Li et al., 2021). This changes the minimum to which the learning algorithms converge to. Here we focus on symmetry breaking due to the differences in the iterative learning algorithm, while keeping the loss function fixed (as in Li et al. (2021)). So all algorithms must converge to the same minimum but follow trajectories that differ in how close they get to the sparse target. The optimal early stopping point is the trajectory point that is closest to the target.

We observe that when the number of examples is larger than the input dimension and the target is sparse linear, then multiplicative updates, spindly networks, and priming all produce trajectories that pass extremely close to the sparse target before finally veering away and learning the noise. Rotation invariant algorithms cannot produce such a bias, even when the input is unbalanced (anisotropic covariance) provided it is drawn from a rotationally symmetric distribution. In essence, they learn the signal and the noise at the same rate. Interestingly, adaptive learning rate algorithms such as Adagrad (Duchi et al., 2011a) and Adam (Kingma and Ba, 2015) show the opposite bias, producing trajectories that are curved away from sparse solutions and learn the noise before the signal.

We arrive at these observations by deriving closed-form solutions for the continuous time (gradient flow) weight trajectories of each algorithm for sparse noisy regression problems. We found forms for the isotropic covariance case but also visualize the trajectories of the anisotropic case. We show how the behavior of the different algorithms can be understood from a geometric analysis

2. This proof technique is interesting in its own right and is different from the regression techniques used for the under-constrained case (Warmuth et al., 2021).

that recasts mirror descent and other preconditioned gradient methods as gradient descent under an altered geometry, where the preconditioner acts as a Riemannian metric.

Finally we make some preliminary experimental observations for nonlinear models by contrasting gradient descent training on a simple three layer feed forward neural net when each logit on the first layer is connected to all inputs (fully connected) versus connected to all inputs via the spindly network (Figure 1). Our experiments show in case b, gradient descent produces sparse solutions for the input layer, is much less confused by additional noisy features and can learn quickly from informative features.

#### ADDITIONAL RELATED WORK:

There is a long history of contrasting the generalization ability of additive versus multiplicative updates (see e.g. [Kivinen et al. \(1997\)](#); [Kivinen and Warmuth \(1997\)](#)). Surprisingly there is a connection between both update families rooted in the observation that when the weights are products of parameters, then gradient descent is biased towards sparsity ([Gunasekar et al., 2017](#); [Kerekes et al., 2021](#)). More precisely ([Amid and Warmuth, 2020](#)), the gradient flow of multiplicative updates on a linear neuron equals the gradient flow of gradient descent on the spindly network of Figure 1.

In this paper we prove lower bounds for sparse noisy regression problems. It might also be possible to obtain lower bounds for the classification setting when the number of examples exceeds the VC dimension, building on the setup of ([Ng, 2004](#); [Li et al., 2021](#)). However for technical reasons the additional loss in the VC dimension / Rademacher complexity has a square root term that seems to make this bounding methodology non-optimal in the noisy case. Our upper bounds for the spindly network of Figure 1 have also been proven using the R.I.P. assumption ([Vaskevicius et al., 2019](#)). For completeness we gave a self-contained proof in the appendix that also works for multiplicative updates and their approximations.

Note that we have hardness results for a fixed noisy sparse linear problem: First feature plus noise. We prove that the gap in performance is due to rotation invariance. So we avoid using staircase, cork screw or cryptographically hard functions ([Abbe and Boix-Adsera, 2022](#)) for proving lower bounds. Also in [Shamir \(2018\)](#), the target class is much more complicated and the hard input distribution is algorithm specific and therefore no fixed problem is given that is hard for all algorithms in the invariance class.

## 1. The lower bound method

### 1.1. Rotation invariance and problem setup

An *example*  $(x, y)$  is a  $d$ -dimensional vector, followed by a real-valued label  $y \in \mathbb{R}$ . We specify a training set as a tuple  $(\underset{n,d}{\mathbf{X}}, \underset{n}{\mathbf{y}})$  containing  $n$  training examples, where the rows of *input matrix*  $\mathbf{X}$  are the  $n$  (transposed) input vectors and the *target*  $\mathbf{y}$  is a vector of their labels.

A *learning algorithm* is a mapping which, given the training set  $(\mathbf{X}, \mathbf{y})$ , produces a real-valued *prediction function*  $\mathbb{R}^d \ni \mathbf{x} \mapsto \hat{y}(\mathbf{x} | \mathbf{X}, \mathbf{y}) \in \mathbb{R}$ . An algorithm is called *rotation invariant* ([Warmuth and Vishwanathan, 2005](#); [Warmuth et al., 2021](#)) if for any orthogonal matrix  $\underset{d,d}{\mathbf{U}}$  and any input  $\mathbf{x} \in \mathbb{R}^d$ :

$$\hat{y}(\mathbf{U}\mathbf{x} | \mathbf{X}\mathbf{U}^\top, \mathbf{y}) = \hat{y}(\mathbf{x} | \mathbf{X}, \mathbf{y}) \quad (1)$$

In other words, the prediction  $\hat{y}(\mathbf{x} | \mathbf{X}, \mathbf{y})$  remains the same if we rotate both  $\mathbf{x}$  and all examples from  $\mathbf{X}$  by the same orthogonal matrix  $\mathbf{U}$ . If the algorithm is randomized, based on an internal

random variable  $Z$ ,<sup>3</sup> then  $\hat{y}(\mathbf{x}_{\text{te}}|\mathbf{X}, \mathbf{y})$  is a random variable given by some function  $f_{\mathbf{x}_{\text{te}}, \mathbf{X}, \mathbf{y}}(Z)$ , and the equality sign in equation (1) should be interpreted as “identically distributed”.

Our lower bounds in sections 1.2-1.3 hold for any rotation invariant algorithm. In particular, Warmuth et al. (2021) have shown that any neural network with a fully-connected input layer (and arbitrary remaining layers), in which the weights in the input layer are initialized randomly with a rotation invariant distribution (e.g. i.i.d. Gaussians) and are trained by gradient descent, is rotation invariant and is thus subject to our lower bound. The reason is that such a network can be written as  $f(\mathbf{w}_1 \cdot \mathbf{x}, \mathbf{w}_2 \cdot \mathbf{x}, \dots, \mathbf{w}_h \cdot \mathbf{x}, \boldsymbol{\theta})$ , and the gradient  $\nabla_{\mathbf{w}_i} f$  for unit  $i$  in the first hidden layer is equal to the instance  $\mathbf{x}$  times a scalar that depends on  $\mathbf{x}$  only via  $\mathbf{w}_i \cdot \mathbf{x}$ , and updating of the later layer weights  $\boldsymbol{\theta}$  depends on the input only via  $\mathbf{w}_i \cdot \mathbf{x}$  (i.e., via the computation in the first layer). Therefore it is easy to show by induction on  $t$ , that rotating all instances results in the same rotation of all  $\mathbf{w}_{i,t}$ . Thus the rotation has no effect on the dot products  $\mathbf{x} \cdot \mathbf{w}_i$  computed at the input layer. In contrast, learning with  $f((\mathbf{u}_1 \odot \mathbf{v}_1) \cdot \mathbf{x}, \dots, (\mathbf{u}_h \odot \mathbf{v}_h) \cdot \mathbf{x}, \boldsymbol{\theta})$ , where the parameters  $\mathbf{w}_i$  connected to the input  $\mathbf{x}$  are replaced by  $\mathbf{u}_i \odot \mathbf{v}_i$  (“spindlified”), is not rotation invariant.

## 1.2. Lower bounds for rotation invariant algorithms

Our method for proving lower bounds builds on the following observation: Given any rotation invariant algorithm and any learning problem, the algorithm will achieve the same loss on all rotated versions of that problem. We can therefore consider a Bayesian setting where the problem is sampled uniformly from all rotated versions, and the optimal solution provides a lower bound on the loss of the algorithm. Intuitively, being rotation invariant forces an algorithm to be agnostic over all possible rotations of the problem, and hedging its bets in this way prevents it from excelling at any specific problem instance. In Section 1.3 we apply this reasoning to linear regression to show that a rotation invariant algorithm cannot efficiently learn sparse solutions, because it must be equally efficient at finding any other solution (including rotated, non-sparse ones).

Formally, let the learning problem be defined by (a) an input distribution  $p_{\text{in}}(\tilde{\mathbf{X}})$  with the input matrix  $\tilde{\mathbf{X}}_{n+1,d} = [\mathbf{X}, \mathbf{x}_{\text{te}}^\top]$  consisting of the training matrix  $\mathbf{X}$  and the test example  $\mathbf{x}_{\text{te}}$  (b) an observation model  $q(\tilde{\mathbf{y}}|\tilde{\mathbf{X}})$  which gives the joint conditional distribution over  $n$  training outcomes and a test outcome,  $\tilde{\mathbf{y}}_{n+1} = [\mathbf{y}, y_{\text{te}}]$ , and (c) a loss function  $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ . We assume the input distribution is rotationally symmetric, meaning  $p_{\text{in}}(\tilde{\mathbf{X}}) = p_{\text{in}}(\tilde{\mathbf{X}}\mathbf{U}^\top)$  for any orthogonal  $\mathbf{U}$ . The task of any algorithm will be to produce predictions  $\hat{y}(\mathbf{x}_{\text{te}}|\mathbf{X}, \mathbf{y})$  to minimize the loss on the test outcomes,  $\mathcal{L}(\hat{\mathbf{y}}, y_{\text{te}})$ . Note that this setup allows arbitrary conditional dependencies among observations (not just iid problems), including dependencies between the training and the test sets.

For any orthogonal  $\mathbf{U}_{d,d}$ , define the rotated observation model as  $q_{\mathbf{U}}(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}) = q(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}\mathbf{U}^\top)$ . Now define a new learning problem by first sampling  $\mathbf{U}$  uniformly (under the Haar measure  $p_{\text{H}}$ ) and then generating observations according to  $q_{\mathbf{U}}$ . This is equivalent to a symmetrized observation model  $\hat{q}$  that is a mixture over all  $q_{\mathbf{U}}$ :

$$\hat{q}(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}) = \int q_{\mathbf{U}}(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}) dp_{\text{H}}(\mathbf{U})$$

3. For example, in neural networks  $Z$  would correspond to a random initialization of the parameter vector.

The Bayes optimal prediction can be expressed by computing a posterior over  $\mathbf{U}$  and integrating expected loss over this posterior:

$$\hat{\mathbf{y}}^*(\mathbf{x}_{\text{te}}|\mathbf{X}, \mathbf{y}) = \arg \min_{\hat{\mathbf{y}}} \int \mathbb{E}_{y_{\text{te}} \sim q_{\mathbf{U}}(\cdot|\tilde{\mathbf{X}}, \mathbf{y})} [\mathcal{L}(\hat{\mathbf{y}}, y_{\text{te}})] p(\mathbf{U}|\tilde{\mathbf{X}}, \mathbf{y}) d\mathbf{U}.$$

Thus  $\hat{\mathbf{q}}$  is difficult, especially for large  $d$ , because equal prior probability must be given to all possible rotations. We define the optimal expected loss on this problem as

$$L_{\mathcal{B}}(\hat{\mathbf{q}}) = \mathbb{E}_{\tilde{\mathbf{X}} \sim p_{\text{in}}, \tilde{\mathbf{y}} \sim \hat{\mathbf{q}}(\cdot|\tilde{\mathbf{X}})} [\mathcal{L}(\hat{\mathbf{y}}^*(\mathbf{x}_{\text{te}}|\mathbf{X}, \mathbf{y}), y_{\text{te}})].$$

Our first result is that the performance of any rotation invariant algorithm on the original problem, defined by  $q$ , is lower bounded by  $L_{\mathcal{B}}(\hat{\mathbf{q}})$ .

**Theorem 1** *Given a rotationally symmetric  $p_{\text{in}}(\tilde{\mathbf{X}})$ , an observation model  $q(\tilde{\mathbf{y}}|\tilde{\mathbf{X}})$ , a loss function  $\mathcal{L}$ , and a rotationally invariant learning algorithm  $\hat{\mathbf{y}}_n(\cdot|\mathbf{X}, \mathbf{y})$ , define the expected loss*

$$L_{\hat{\mathbf{y}}}(q) = \mathbb{E}_{\tilde{\mathbf{X}} \sim p_{\text{in}}, \tilde{\mathbf{y}} \sim q(\cdot|\tilde{\mathbf{X}}), Z} [\mathcal{L}(\hat{\mathbf{y}}(\mathbf{x}_{\text{te}}|\mathbf{X}, \mathbf{y}), y_{\text{te}})],$$

*This loss is bounded by  $L_{\hat{\mathbf{y}}}(q) \geq L_{\mathcal{B}}(\hat{\mathbf{q}})$ .*

The proof in Appendix A shows any rotationally invariant algorithm will satisfy  $L_{\hat{\mathbf{y}}}(q_{\mathbf{U}}) = L_{\hat{\mathbf{y}}}(q)$  for all  $\mathbf{U}$ . This implies  $L_{\hat{\mathbf{y}}}(q) = L_{\hat{\mathbf{y}}}(\hat{\mathbf{q}}) \leq L_{\mathcal{B}}(\hat{\mathbf{q}})$ .

As we show in this paper, a consequence of Theorem 1 is that rotational invariance prevents efficient learning of problems characterized by properties that are not rotationally invariant, such as sparsity. Algorithms with inductive biases for such properties are necessarily not rotation invariant and can give dramatically better performance. Although we have stated the theorem in terms of rotational invariance, it is easily extended to other transformation groups  $\mathcal{T}$  on the input (by replacing  $\mathbf{U}$  with elements of  $\mathcal{T}$  and requiring  $p_{\text{in}}$  to be symmetric under  $\mathcal{T}$ ). For example, natural gradient descent (NGD) is often touted for being invariant to arbitrary smooth reparameterization (Amari and Douglas, 1998), but this invariance comes at a cost of being unable to efficiently learn in environments that are not invariant in this way. In particular, since spindly and fully connected networks are related by smooth reparameterization, NGD performs equivalently on both (see discussion in (Kerekes et al., 2021)). Thus the lower bounds we prove apply to NGD on the network in Figure 1, whereas we show vanilla gradient descent on this network breaks the lower bound.

### 1.3. Lower bound for least-squares regression

We demonstrate how Theorem 1 can provide a quantitative lower bound for a specific class of learning problems. We then show how this bound is easily beaten by non-rotation-invariant algorithms in Section 2. In the specific problem class we consider, the number of training examples is  $n = md$  for some integer  $m$ , and  $\mathbf{X}$  consists of  $m$  stacked copies of a matrix  $\mathbf{H} = \sqrt{d} \mathbf{V}_{d,d}$  (i.e.  $\mathbf{X} = [\mathbf{H}; \dots; \mathbf{H}]_{\times m}$ ), where  $\mathbf{V}$  is a random orthogonal matrix distributed according to the Haar measure.<sup>4</sup> Thus  $p_{\text{in}}(\mathbf{X})$  is rotationally symmetric. The test input is one of the rows of  $\mathbf{H}$ ,  $\mathbf{x}_{\text{te}} = \mathbf{h}_k$ ,

4. We multiply  $\mathbf{V}$  by  $\sqrt{d}$  in order to keep the lengths of examples  $\|\mathbf{x}_i\|$  (rows of  $\mathbf{X}$ ) equal to  $\sqrt{d}$ , so that their coordinates  $x_{ij}$  (individual features) are of order 1 on average; any other scaling would work as well, resulting only in a relative change of the effective label noise level.

with index  $k$  drawn uniformly at random. We assume the labels are the first feature of  $\tilde{\mathbf{X}}$  plus Gaussian noise, that is

$$q(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}) = \mathcal{N}(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}\mathbf{e}_1, \sigma^2\mathbf{I}_{n+1}), \quad \text{where } \mathbf{e}_1 = (1, 0, \dots, 0)^\top, \quad (2)$$

which can be equivalently written as

$$\mathbf{y} = \mathbf{X}\mathbf{e}_1 + \boldsymbol{\xi}, \text{ where } \boldsymbol{\xi} \sim N(\mathbf{0}, \sigma^2\mathbf{I}_{md}), \text{ and } y_{te} = \mathbf{h}_k^\top \mathbf{e}_1 + \xi_{te}, \text{ where } \xi_{te} \sim N(0, \sigma^2).$$

Note that while the inputs are shared in the training and the test parts, the test label is generated using a ‘fresh’ copy of the noise variable  $\xi_{te}$ . The fixed choice of  $\mathbf{e}_1$  as opposed to any other  $\mathbf{e}_i$  is made w.l.o.g. Indeed, Theorem 1 implies that a rotationally invariant algorithm will have the same loss for  $\mathbf{X}\mathbf{e}_1$  as it will for  $\mathbf{X}\mathbf{w}$  for any other unit vector  $\mathbf{w}$ .

The accuracy of prediction  $\hat{\mathbf{y}} = \hat{\mathbf{y}}(\cdot|\mathbf{X}, \mathbf{y})$  on the test set  $(\mathbf{h}_k, y_{te})$  is measured by the *squared loss*  $\mathcal{L}(\hat{\mathbf{y}}, y_{te}) = (\hat{\mathbf{y}} - y_{te})^2$ . Let  $\hat{\mathbf{y}}$  denote the vector of predictions for all possible test instances,  $\hat{y}_k = \hat{\mathbf{y}}(\mathbf{h}_k|\mathbf{X}, \mathbf{y})$ , and let  $\mathbf{y}_{te}$  denote the vector of test labels,  $y_{te_k} = \mathbf{h}_k^\top \mathbf{e}_1 + \xi_{te}$ , which can be jointly written as  $\mathbf{y}_{te} = \mathbf{H}\mathbf{e}_1 + \xi_{te}\mathbf{1}$  with  $\mathbf{1} = (1, \dots, 1)$ . The expected value of the loss over the random choice of  $k \in \{1, \dots, d\}$  and over the independent test label noise is given by:

$$\begin{aligned} \mathbb{E}_{k, \xi_{te}}[\mathcal{L}(\hat{\mathbf{y}}, y_{te})] &= \frac{1}{d} \mathbb{E}_{\xi_{te}} [\|\hat{\mathbf{y}} - \mathbf{y}_{te}\|^2] = \frac{1}{d} \mathbb{E}_{\xi_{te}} [\|\hat{\mathbf{y}} - \mathbf{H}\mathbf{e}_1 + \xi_{te}\mathbf{1}\|^2] \\ &= \frac{1}{d} \|\hat{\mathbf{y}} - \mathbf{H}\mathbf{e}_1\|^2 + \frac{2}{d} \underbrace{\mathbb{E}_{\xi_{te}}[\xi_{te}]}_{=0} (\hat{\mathbf{y}} - \mathbf{H}\mathbf{e}_1)^\top \mathbf{1} + \frac{1}{d} \mathbb{E}_{\xi_{te}} \underbrace{[\xi_{te}^2]}_{=\sigma^2} \|\mathbf{1}\|^2 \\ &= \frac{1}{d} \|\hat{\mathbf{y}} - \mathbf{H}\mathbf{e}_1\|^2 + \sigma^2. \end{aligned}$$

Clearly, the expression above is minimized by setting the prediction vector to  $\hat{\mathbf{y}}^* = \mathbf{H}\mathbf{e}_1$ , and thus the smallest achievable expected loss is equal to  $\sigma^2$ . Subtracting this loss, we get the expression for the excess risk of the learning algorithm, which we call the *error* of  $\hat{\mathbf{y}}$ :

$$e(\hat{\mathbf{y}}) = \mathbb{E}_{k, \xi_{te}}[\mathcal{L}(\hat{\mathbf{y}}, y_{te})] - \mathbb{E}_{k, \xi_{te}}[\mathcal{L}(\hat{\mathbf{y}}^*, y_{te})] = \frac{1}{d} \|\hat{\mathbf{y}} - \mathbf{H}\mathbf{e}_1\|^2.$$

When the prediction is *linear*,  $\hat{\mathbf{y}} = \mathbf{H}\hat{\mathbf{w}}$  for some weight vector  $\hat{\mathbf{w}} \in \mathbb{R}^d$ , we can also refer to the error of  $\hat{\mathbf{w}}$  as the error of its predictions:

$$e(\hat{\mathbf{w}}) = \frac{1}{d} \|\mathbf{H}\hat{\mathbf{w}} - \mathbf{H}\mathbf{e}_1\|^2 = \frac{1}{d} (\hat{\mathbf{w}} - \mathbf{e}_1)^\top \underbrace{\mathbf{H}^\top \mathbf{H}}_{d\mathbf{I}} (\hat{\mathbf{w}} - \mathbf{e}_1) = \|\hat{\mathbf{w}} - \mathbf{e}_1\|^2. \quad (3)$$

We prove the following lower bound for rotationally invariant algorithms on this problem:

**Theorem 2** Let  $\mathbf{V}$  be a random orthogonal matrix, and let  $\mathbf{H} = \sqrt{d}\mathbf{V}$ . Let  $(\mathbf{X}, \mathbf{y})$  be the training set with  $\mathbf{X} = [\mathbf{H}; \dots; \mathbf{H}]$  and labels  $\mathbf{y}$  generated according to (2). Then the expected error (with respect to  $\mathbf{V}$ ) of any rotation-invariant learning algorithm is at least

$$\mathbb{E}_{\mathbf{V}}[e(\hat{\mathbf{y}})] \geq \frac{d-1}{d} \frac{\sigma^2}{\sigma^2 + m}.$$



The proof in Appendix B uses Theorem 1 and closely follows Section 1.2: We start with a Bayesian setting with the rotated observation model  $q(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}\mathbf{U}^\top)$  for a random orthogonal matrix  $\mathbf{U}$ . This is equivalent to simply rotating the target weight vector by  $\mathbf{U}^\top$ , because  $\tilde{\mathbf{X}}\mathbf{U}^\top\mathbf{e}_1 = \tilde{\mathbf{X}}(\mathbf{U}^\top\mathbf{e}_1)$ . Thus we can equivalently consider a linear model  $\tilde{\mathbf{y}} = \tilde{\mathbf{X}}\mathbf{w} + \tilde{\boldsymbol{\xi}}$ , where  $\mathbf{w}$  is drawn uniformly from the unit sphere  $\mathcal{S}^{d-1} = \{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\| = 1\}$ . Given the square loss function and linear observation model, the optimal Bayes predictor is based on the posterior mean,  $\mathbb{E}[\mathbf{w}|\mathbf{X}, \mathbf{y}]$ .

Even though the posterior mean does not have a simple analytic form for the prior distribution over a unit sphere, we use results from (Marchand, 1993; Dicker, 2016) to show that the Ridge Regression (RR) predictor (which is the Bayes predictor for the Gaussian prior) with appropriately chosen regularization constant has expected error at most  $\frac{1}{d} \frac{\sigma^2}{\sigma^2+m}$  larger than that of the Bayes predictor. Thus, it suffices to analyze the RR predictor which we show to be at least  $\frac{\sigma^2}{\sigma^2+m}$ . This implies the Bayes error is at least  $\frac{d-1}{d} \frac{\sigma^2}{\sigma^2+m}$ , and no other algorithm can achieve any better error for this problem. Finally, due to the rotation-symmetric distribution of the inputs, we can now apply Theorem 1 which implies that every rotation invariant algorithm has error at least  $\frac{d-1}{d} \frac{\sigma^2}{\sigma^2+m}$  for the original sparse linear regression problem  $\tilde{\mathbf{y}} = \tilde{\mathbf{X}}\mathbf{e}_1 + \tilde{\boldsymbol{\xi}}$ . For the sake of completeness, we also give an i.i.d. version of this lower bound in Appendix C.

Previous lower bound proofs for sparse problems for the case when the number of examples is less than the input dimension used fixed choices for input matrix  $\mathbf{X}$  such as the  $d$  dimensional Hadamard matrix (Kivinen et al., 1997; Warmuth and Vishwanathan, 2005). In the over-constrained case, the lower bound does not hold for any fixed choice of  $\mathbf{X}$  (of full rank). For any fixed full-rank  $\mathbf{X}$ , there exists a row vector  $\mathbf{v}$  s.t.  $\mathbf{v}\mathbf{X} = \mathbf{e}_1^\top$ , and the linear algorithm  $\hat{y}(\mathbf{x}|\mathbf{X}, \mathbf{y}) = \mathbf{v}\mathbf{X}\mathbf{x}$  achieves minimal loss  $\sigma^2$  while being trivially rotationally invariant because  $\mathbf{v}\mathbf{X}\mathbf{U}^\top\mathbf{U}\mathbf{x} = \mathbf{v}\mathbf{X}\mathbf{x}$ . Thus the assumption of rotationally symmetric input distribution is essential.<sup>5</sup>

This counterexample requires knowledge of the order of the examples (i.e., the algorithm makes different predictions if the rows of  $\mathbf{X}$  are permuted). Now consider  $\mathbf{X} = [\mathbf{H}; \dots; \mathbf{H}] \text{Diag}(\mathbf{p})$  with  $p_1 = 2, p_{i>1} = 1$ . Then the target  $\mathbf{e}_1$  is embedded as the first principal component  $\mathbb{PC}_1(\mathbf{X}^\top\mathbf{X})$ , so an algorithm that used this as its weight vector,  $\hat{y}(\mathbf{x}|\mathbf{X}, \mathbf{y}) = \mathbb{PC}_1(\mathbf{X}^\top\mathbf{X})^\top\mathbf{x}$ , would achieve minimal loss. This algorithm is rotationally invariant but fails if the input matrix  $\mathbf{X}$  was rotated.

## 2. Upper bounds

We now show how to break the above lower bound of  $\frac{d-1}{d} \frac{\sigma^2}{\sigma^2+m}$  on the error (excess risk) of rotation invariant algorithms. We focus on the upper bounds for  $\text{EG}^\pm$  and Approximated  $\text{EGU}^\pm$ . (Similar upper bounds for the spindly network, as well as the novel priming method (Warmuth and Amid, 2023) are given in appendices F and G, respectively.) We use the same setup as in the lower bound, i.e.  $\mathbf{X}$  consists of  $m$  stacked copies of a matrix  $\mathbf{H} = \sqrt{d}\mathbf{V}$ , when  $\mathbf{V}$  is a rotation matrix and  $\mathbf{y}$  is sparse linear (the first component) plus Gaussian noise with variance  $\sigma^2$ . The only difference is that for the lower bound,  $\mathbf{V}$  was randomly chosen, but the upper bounds hold for *any* rotation matrix  $\mathbf{V}$ . The upper bound on the error that we obtain<sup>6</sup> is essentially  $O(\sigma^2 \frac{\log d}{md})$ .

We begin with a bound for the normalized version of the multiplicative update called  $\text{EG}^\pm$ . The proof relies on the facts (a) under large learning rate, the  $\text{EG}^\pm$  weight estimate becomes argmax

5. In the Hadamard matrix based lower bounds of Warmuth and Vishwanathan (2005), the ease of learning say  $\mathbf{e}_1$  is overcome by averaging over all  $n$  targets  $\mathbf{e}_i$ .

6. Note that at this point, we do not consider general input matrices  $\mathbf{X}$  in the upper bounds. Allowing arbitrary covariance structure makes the analysis much more complicated and the general case is left for future research.

over the negative gradient coordinates, and (b) with high probability, the least component of the gradient is the first one, so that the argmax is  $e_1$ .  $\text{EG}^\pm$  makes use of the fact that the norm of the linear target  $e_1$  is 1 and this additional knowledge allows the speedup.<sup>7</sup> We then prove our main upper bound for the unnormalized  $\text{EGU}^\pm$ .

### 2.1. Upper bound for the Exponentiated Gradient update

We consider a batch version of the *2-sided Exponentiated Update algorithm* ( $\text{EG}^\pm$ ) (Kivinen and Warmuth, 1997). The batch  $\text{EG}^\pm$  algorithm maintains two vectors,  $\mathbf{v}_t^+$  and  $\mathbf{v}_t^-$ , and its weight estimate is given by  $\mathbf{w}_t = \mathbf{v}_t^+ - \mathbf{v}_t^-$ . It starts with a set of weights  $\mathbf{v}_1^+ = \mathbf{v}_1^- = \frac{1}{2d}\mathbf{1}$  (so that  $\|\mathbf{v}_1^+\|_1 + \|\mathbf{v}_1^-\|_1 = 1$ ) and updates according to

$$\mathbf{v}_{t+1}^+ \propto \mathbf{v}_t^+ \odot e^{-\eta \nabla L(\mathbf{w}_t)}, \quad \mathbf{v}_{t+1}^- \propto \mathbf{v}_t^- \odot e^{\eta \nabla L(\mathbf{w}_t)},$$

where  $\odot$  is component-wise multiplication, and the normalization enforces  $\|\mathbf{v}_{t+1}^+\|_1 + \|\mathbf{v}_{t+1}^-\|_1 = 1$ .

**Theorem 3** *The expected error of the batch  $\text{EG}^\pm$  algorithm after the first iteration is bounded by*

$$e(\mathbf{w}_2) \leq 2de^{-\eta} + 8de^{-\frac{md}{32\sigma^2}}.$$

**Proof sketch:** The gradient of the loss can be written as  $2(\mathbf{w} - \mathbf{e}_1 - \boldsymbol{\zeta})$ , where  $\boldsymbol{\zeta}$  are i.i.d. Gaussian noise variables which are combinations of the original noise variables. Using the deviation bound for Gaussians together with union bound, with probability at least  $1 - 2d \exp\{-md/(32\sigma^2)\}$  all noise variables are bounded by  $1/4$ . Thus, the first coordinate of the negative gradient is larger than all other coordinates by at least 1, so that the first weight exponentially dominates all the other weights already after one step of the algorithm, and the error drops to  $2de^{-\eta}$ . If the high probability event does not hold, we bound the error by its maximum value 4. See Appendix D for full proof. ■

### 2.2. Upper bound for the Approximated Unnormalized Exponentiated Gradient update

We now drop the normalization of  $\text{EG}^\pm$  and use the approximation  $\exp(x) \approx 1 + x$ .<sup>8</sup> The resulting approximation of the unnormalized  $\text{EGU}^\pm$  algorithm updates its weights as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sqrt{\mathbf{w}_t^2 + 4\beta^2} \nabla L(\mathbf{w}_t), \quad (4)$$

where  $\beta > 0$  is a parameter and all operations (squaring the weights and taking square root) are done component-wise (see Appendix E for a derivation). When  $\beta = 0$  this update is closely related to gradient descent on the spindly network of Figure 1. We start with  $\mathbf{w}_1 = \mathbf{0}$ . Note that unlike  $\text{EG}^\pm$  the update does not constrain its weights by normalizing. Nevertheless, we show that the algorithm achieves an upper bound on the error that is essentially  $O(\frac{d}{\log d})$  better than the error of any rotation invariant algorithm:

7. We can also provide an upper bound on the error of the online version of  $\text{EG}^\pm$  for an arbitrary input matrix  $\mathbf{X}$  with fixed feature range via a standard worst-case regret analysis followed by the online-to-batch conversion (see e.g. Kivinen and Warmuth (1997)). The bound so obtained would however give a slower rate of order  $O(\sqrt{\log d}/(dm))$ , which still has a substantially better dependence on dimension  $d$  than the lower bound from the previous section.

8. This approximated version of EGU was introduced in (Kivinen and Warmuth, 1997). It was also used in the normalized update PROD (Cesa-Bianchi et al., 2007).



**Theorem 4** Assume  $d \geq 4$  is such that  $\sqrt{d}$  is an integer. Consider the Approximated  $EGU^\pm$  algorithm (4) with  $\beta = 1/(2d)$  and learning rate  $\eta = 1/4$  and let  $m \geq 8\sigma^2 \ln \frac{2d}{\delta} = \Omega(\sigma^2 \log(d/\delta))$ . With probability at least  $1 - \delta$ , the algorithm run for  $T = 4\sqrt{d}$  steps achieves error bounded by:

$$e(\mathbf{w}_{T+1}) \leq \frac{10\sigma^2 \ln \frac{2d}{\delta}}{md} + 9e^{-\frac{8}{3}\sqrt{d}+2\ln d} = O\left(\frac{\sigma^2 \log d}{md} + e^{-\frac{8}{3}\sqrt{d}}\right)$$

**Proof sketch:** (full proof in Appendix E) Similarly as for  $EG^\pm$ , with high probability all noise variables are small. We can interpret the weight update (4) as the gradient descent update on the weights with the *effective learning rates*  $\eta\sqrt{w_t^2 + 4\beta^2}$ . Since  $\beta = 1/(2d)$  and  $\mathbf{w}_1 = \mathbf{0}$ , these learning rates are initially small and of order  $O(1/d)$ . We then show by a careful analysis of the update that the weights  $w_{t,i}$  remain small for all coordinates  $i$  except  $i = 1$ , and so do the associated learning rates. Therefore, after  $T$  steps, the algorithm does not move significantly away from zero on these coordinates. Meanwhile, the weight on the first coordinate  $w_{t,1}$  increases towards 1. While the initial rate of increase is small as well, it accelerates over time as  $\eta\sqrt{w_{t,1}^2 + 4\beta^2}$  increases due to increasing  $w_{t,1}$ . Eventually  $w_{T+1,1}$  gets very close to 1 after  $T$  steps of the algorithm. ■

### 3. Gradient flow trajectories

In our setting of overconstrained optimization without explicit regularization, all algorithms operate on the same loss landscape but follow different paths to the minimum. To understand these differences, we derive exact expressions for the weight trajectories for the continuous time (gradient flow) versions of several representative algorithms. We then use these results to analyze the algorithms' ability to learn sparse targets without overfitting to noise.

#### 3.1. Preconditioning, mirror descent, reparameterization, and Riemannian descent

Gradient descent (GD) on a given loss function can be generalized in several ways, including preconditioned GD, mirror descent (MD), GD on reparameterized variables, and Riemannian GD. We show how they are all interchangeable in the continuous-time case.

Preconditioned GD premultiplies the gradient with a matrix  $\mathbf{P}$  that can depend on the current weights or the training history. Explicit preconditioning methods include adaptive learning rate algorithms such as Adagrad (Duchi et al., 2011b), natural gradient descent which preconditions with the inverse Fisher information matrix (Amari, 1998), and variational Bayesian methods that precondition with the prior or posterior covariance (Lambert et al., 2022; Chang et al., 2023).

$$\dot{\mathbf{w}} = -\mathbf{P}\nabla_{\mathbf{w}}L \quad (5)$$

Reparameterizing refers to defining some injective function  $\hat{\mathbf{w}} = g(\mathbf{w})$  and performing GD on the new variable:

$$\dot{\hat{\mathbf{w}}} = -\nabla_{\hat{\mathbf{w}}}L \quad (6)$$

MD (Nemirovsky and Yudin, 1983) involves an invertible mirror map  $\tilde{\mathbf{w}} = f(\mathbf{w})$  where  $f$  is the gradient of a convex function, with the update

$$\dot{\tilde{\mathbf{w}}} = -\nabla_{\tilde{\mathbf{w}}}L \quad (7)$$

Riemannian GD generalizes to parameter spaces with non-Euclidean geometry (specifically, differentiable manifolds) (Bonnabel, 2013). In Riemannian geometry the update  $\dot{\mathbf{w}}$  lies in the tangent space  $T_{\mathbf{w}}$  (a direction of motion based at  $\mathbf{w}$ ) whereas the gradient  $\nabla_{\mathbf{w}} L$  lies in the cotangent space  $T_{\mathbf{w}}^*$  (the dual of  $T_{\mathbf{w}}$ ). Unlike in Euclidean geometry,  $T_{\mathbf{w}}$  and  $T_{\mathbf{w}}^*$  cannot be identified and instead mapping between them requires a metric  $\Gamma_{\mathbf{w}} : T_{\mathbf{w}} \rightarrow T_{\mathbf{w}}^*$  or equivalently  $\Gamma_{\mathbf{w}} : T_{\mathbf{w}} \times T_{\mathbf{w}} \rightarrow \mathbb{R}$ . This leads to an update that is the direction of steepest descent with respect to the geometry:

$$\dot{\mathbf{w}} = -\Gamma_{\mathbf{w}}^{-1} \nabla_{\mathbf{w}} L \quad (8)$$

**Theorem 5** *The continuous-time versions of reparameterized GD, MD, and Riemannian GD are all equivalent to preconditioned GD under the relation*

$$\mathbf{P} = \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}} \left( \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}} \right)^{\top} = \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}} = \Gamma_{\mathbf{w}}^{-1} \quad (9)$$

Appendix H gives the proof and then uses this result (a) to derive a close connection between EGU and the spindly network and (b) to show how EGU implicitly operates on a non-Euclidean geometry (Figure 3) that tends to hold it in sparse regions of parameter space. Rotationally invariant algorithms cannot do this because their implicit geometry must be rotationally symmetric.

### 3.2. Trajectory solutions

We derive analytic trajectories for the simple linear regression problem from Sections 1.3 and 2, for the continuous-time versions of EGU, EGU $\pm$ , primed GD, and Adagrad (Duchi et al., 2011a). For space reasons we present the expressions for the trajectories here and provide the derivations and full details in Appendix I.

For continuous EGU the trajectory is (with  $c_i$  a constant depending on initial conditions):

$$w_i(t) = \frac{1}{2} w_i^{\text{LS}} (1 + \tanh(w_i^{\text{LS}} t + c_i)) \quad (10)$$

For continuous EGU $\pm$  the trajectory is (with  $\tau_i$  a transform of  $t$ ):

$$w_i(t) = \frac{w_i^{\text{LS}} \sinh \tau_i + 1}{\sinh \tau_i - w_i^{\text{LS}}} \quad (11)$$

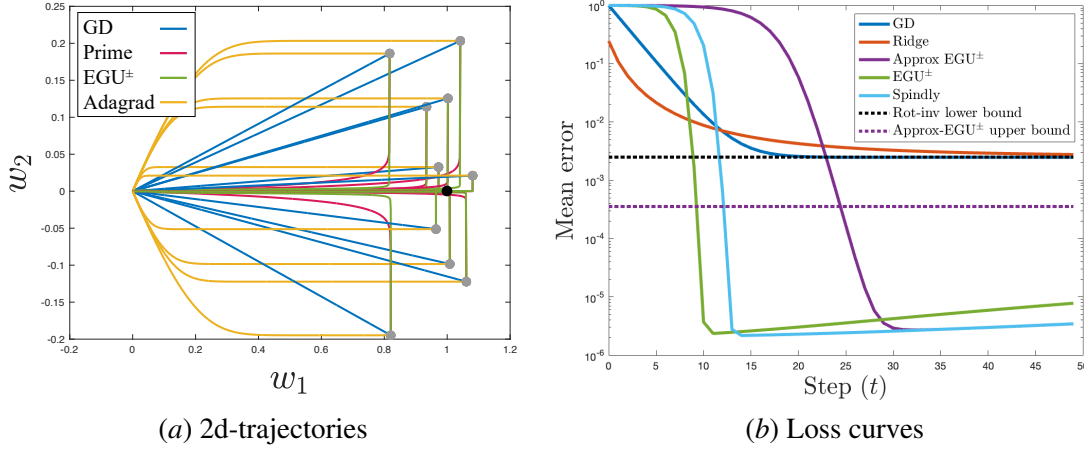
For primed gradient flow, which is similar to the priming method analyzed elsewhere in this paper except the second stage uses gradient flow instead of ridge regression, the trajectory is

$$w_i(t) = w_i^{\text{LS}} + (w_i(0) - w_i^{\text{LS}}) e^{-2t(w_i^{\text{LS}})^2} \quad (12)$$

For continuous-time Adagrad the trajectory is (with constants  $k_i, \ell_i$  depending on initial conditions):

$$w_i(t) = w_i^{\text{LS}} - \text{sign}(w_i^{\text{LS}} - w_i(0)) \sqrt{-\frac{8}{\beta k_i} W(-e^{-k_i(t+\ell_i)})} \quad (13)$$

with Lambert's W function defined by  $x = W(x)e^{W(x)}$ .



**Figure 2:** (a): Trajectories for key algorithms on 10 sampled datasets in toy 2d regression. Black dot shows true sparse target. Grey dots show least squares solution for each dataset. All algorithms converge to LS solution but differ in the order of learning the signal versus noise. EGU $\pm$  and priming learn the sparse target efficiently, overfitting to the noise only at the very end of training. Because GD is rotation invariant it cannot produce this behavior and instead learns the signal and noise at equal rates. Intriguingly, Adagrad shows the opposite behavior and is especially inefficient at learning a sparse target, because of its per-parameter adaptive learning rates. (b): Excess test loss averaged over 100 runs of linear regression with  $d = 1024$ . Dotted lines show the lower bound for rotationally invariant algorithms from Theorem 2 (applies to GD and Ridge) and the upper bound for Approximated EGU $\pm$  from Theorem 4 (using  $\delta = .001$ ).

### 3.3. Experimental visualization

The analytic trajectories derived above are visualized in Figure 2a for a 2d version of our noisy sparse regression problem. This toy problem is sufficient to bring out key differences among the algorithms. Because all considered algorithms optimize the same loss they all converge to the same least squares solution for each training set (grey dots), but they follow qualitatively different trajectories. Priming and EGU $\pm$  pass close to the sparse target (black dot) while GD goes straight to the LLS solution in each run. Surprisingly Adagrad veers away from the sparse target.<sup>9</sup>

These differences in trajectory have dramatic implications for excess test loss as shown in Figure 2b. Priming and EGU $\pm$  have deep dips in the curves corresponding to the fact that their trajectories pass close by the target while the dips of GD and ridge regression are too small to be visible. All algorithms require early stopping to “catch” bottom dip of their error curves. Note that the rotation invariant algorithms both reach but do not beat the lower bound of Theorem 2 while the non-rotation invariant algorithms greatly outperform the upper bound of Theorem 4, showing that the bound we were able to prove is conservative.

Finally, the bounds of Sections 1.3 and 2 and the curves of Figure 2 assume the instance matrix  $\mathbf{X}$  comprises  $m$  copies of a scaled  $d$  dimensional rotation matrix. This unusual setup was needed for technical reasons. However, if we used a Gaussian instance matrix  $\mathbf{X}_{md,d}$  together with sparse linear targets plus Gaussian noise, then the curves would remain essentially unchanged (not shown).

9. In Theorem 3 we also prove exponential convergence of the error of EG $\pm$ . We deemphasize this algorithm because it normalizes based on the norm of the true weight vector and this may be unreasonable. Indeed using learning rate  $\eta = 200$ , EG $\pm$  achieves error below  $10^{-300}$  (not shown). Larger learning rates cause numerical instabilities.

#### 4. Noise experiments on Fashion MNIST

The assumption of rotationally symmetric input distribution is too strong for real datasets. Therefore it is important to complement our theoretical results with experimental demonstration that standard rotationally invariant algorithms cannot exploit asymmetries in real data. Here we report experiments on the Fashion MNIST (Xiao et al., 2017) dataset. One could trivially cheat this task by hard-coding a lookup table of the test set, and this can even be done with a rotation invariant algorithm: If  $\mathbf{X}^\top \mathbf{X}$  is the (known, asymmetric) covariance of the original training set, then when presented with rotated training data  $\mathbf{Z} = \mathbf{X}\mathbf{U}^\top$ , the algorithm infers the unknown  $\mathbf{U}$  to high precision by solving  $\mathbf{Z}^\top \mathbf{Z} = \mathbf{U}\mathbf{X}^\top \mathbf{X}\mathbf{U}^\top$ . It then un-rotates  $\mathbf{z}_{te}$  and uses the lookup table. Such a bespoke algorithm is of course not of interest; the question is whether standard rotation invariant algorithms based on gradient descent can do anything similar, or whether they are limited by their rotation invariance just as they are in the idealized symmetric-input setting.

We use a multilayer feedforward network with two hidden layers of size 256 each. We consider two cases for the input layer weights: 1) fully-connected (each 1st layer hidden node is connected to all inputs) and 2) “spindly” (each 1st layer hidden node is connected to all inputs via the network of Figure 1). In the noise-free case, both variants of the network achieve the same top one accuracy of 85% (although the spindly version takes longer to train). Next, we double the number of features of the examples by augmenting each example with uniformly sampled noise pixels in the range  $[-1, 1]$ . With noisy augmentation, the spindly network still achieves 85% test accuracy while the fully-connected network gets only to 71% (after a much longer training phase). In addition, the learned weights by the two networks are significantly different: The fully-connected network assigns almost the same magnitude of weights to the noisy features as to the image features, while the spindly network allocates much larger weights to the image features. This illustrates that rotation invariant algorithms have a harder time ignoring the noisy features.

Finally, to further compare the different sensitivity to the informativeness of features, we augment each example on top of the noise with its one-hot representation of the 10 target class labels. This splits the features into three categories in terms of their informativeness: 1) highly informative label features, 2) less informative image features, and 3) noise features with no (structured) information. The spindly network achieves 100% test accuracy while the fully-connected network gets to 98%. We observe that the spindly network assigns much larger weights (in magnitude) to the label features while almost ignoring the rest. This phenomenon is less prominent in the fully-connected network. We defer all the details to Appendix K.

#### 5. Open problems

The “full versions” of many common optimization algorithms (Abdulkadirov et al., 2023) such as AdaGrad, Fisher, Adam, RMS Prop are all rotation invariant. However in practice diagonalized versions of these updates are used. In a major step forward we were able to solve the differential equations for all these algorithms when  $\Sigma = \lambda \mathbf{I}$  (the arbitrary  $\Sigma$  cases are challenging open problems). This lets us analyze the inductive biases of all these algorithms on linear neurons based on their gradient flow trajectories. For example, the diagonalized AdaGrad is biased away from sparsity. So using sparsity as a yard stick, we show that already for linear neurons, dramatic differences can occur compared to the optimal algorithms. The question is whether these differences remain when running the same algorithms on neural nets with any number of hidden layers.

## References

- Emmanuel Abbe and Enric Boix-Adsera. On the non-universality of deep learning: quantifying the cost of symmetry. In *Advances in Neural Information Processing Systems*, volume 35, pages 17188–17201. Curran Associates, Inc., 2022.
- Ruslan Abdulkadirov, Pavel Lyakhov, and Nikolay Nagornov. Survey of optimization algorithms in modern neural networks, 04 2023.
- S. Amari and S.C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, volume 2, pages 1213–1216 vol.2, 1998.
- Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.
- Ehsan Amid and Manfred K. Warmuth. Reparameterizing mirror descent as gradient descent. In *Proceedings of Advances in Neural Information Processing Systems*, volume 33, pages 8430–8439, 2020.
- James O. Berger. *Statistical decision theory and Bayesian analysis*. Springer, 1985.
- Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.
- 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.
- Peter Chang, Gerardo Duràn-Martín, Alexander Y Shestopaloff, Matt Jones, and Kevin Murphy. Low-rank extended kalman filtering for online learning of neural networks from streaming data. *arXiv preprint arXiv:2305.19535*, 2023.
- Lee H. Dicker. Ridge regression and asymptotic minimax estimation over spheres of growing dimension. *Bernoulli*, 22(1):1–37, 2016.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011a.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011b.
- Sébastien Gerchinovitz, Pierre Ménard, and Gilles Stoltz. Fano’s Inequality for Random Variables. *Statistical Science*, 35(2):178 – 201, 2020.
- Suriya Gunasekar, Blake E. Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 6151–6159, 2017.
- Prateek Jain, Brian J. Kulis, and Inderjit S. Dhillon. Online linear regression using burg entropy. Technical Report TR-07-08, The University of Texas at Austin, feb 2007.

- Anna Kerekes, Anna Mészáros, and Ferenc Huszár. Depth without the magic: Inductive bias of natural gradient descent. *ArXiv*, abs/2111.11542, 2021.
- Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- 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.
- Marc Lambert, Silvere Bonnabel, and Francis Bach. The recursive variational gaussian approximation (r-vga). *Statistics and Computing*, 32(1):10, 2022.
- Zhiyuan Li, Yi Zhang, and Sanjeev Arora. Why are convolutional nets more sample-efficient than fully-connected nets? In *International Conference on Learning Representations*, 2021.
- Eric Marchand. Estimation of a multivariate mean with constraints on the norm. *The Canadian Journal of Statistics*, 21(4):359–366, 1993.
- Arkadi Nemirovsky and David Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley & Sons, 1983.
- Andrew Y. Ng. Feature selection,  $l_1$  vs.  $l_2$  regularization, and rotational invariance. In *Proceedings of the 21-st International Conference on Machine Learning, Banff, Canada, July 4-8*. ACM New York, NY, 2004.
- Philippe Rigollet and Jan-Christian Hütter. High-dimensional statistics, 2023.
- Ohad Shamir. Distribution-specific hardness of learning neural networks. *Journal of Machine Learning Research*, 19(32):1–29, 2018.
- Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 2968–2979, 2019.
- Manfred K. Warmuth and Ehsan Amid. Open problem: Learning sparse linear concepts by priming the features. In *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 5937–5942. PMLR, Jul 2023.
- Manfred K. Warmuth and Arun Jagota. Continuous and discrete time nonlinear gradient descent: relative loss bounds and convergence. In R. Greiner E. Boros, editor, *Electronic Proceedings of Fifth International Symposium on Artificial Intelligence and Mathematics*. Electronic, <http://rutcor.rutgers.edu/~amai>, 1998.
- Manfred K. Warmuth and S.V.N. Vishwanathan. Leaving the span. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, pages 366–381, 2005.



Manfred K. Warmuth, Wojciech Kotłowski, and Ehsan Amid. A case where a spindly two-layer linear network decisively outperforms any neural network with a fully connected input layer. In *32th International Conference on Algorithmic Learning Theory (ALT)*, volume 132, pages 1–32. PMLR, 2021.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL <http://arxiv.org/abs/1708.07747>. cite arxiv:1708.07747Comment: Dataset is freely available at <https://github.com/zalandoresearch/fashion-mnist> Benchmark is available at <http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/>.

## Appendix A. Proof of the lower bound Theorem 1

**Proof** For any orthogonal  $U$ , the algorithm’s expected loss is

$$\begin{aligned} L_{\hat{y}}(qU) &= \mathbb{E}_{\tilde{\mathbf{X}} \sim p_{\text{in}}, \tilde{\mathbf{y}} \sim q_U(\cdot|\tilde{\mathbf{X}}), Z} [\mathcal{L}(\hat{y}(\mathbf{x}_{\text{te}}|\mathbf{X}, \mathbf{y}), y_{\text{te}})] \\ &= \mathbb{E}_{\tilde{\mathbf{X}} \sim p_{\text{in}}, \tilde{\mathbf{y}} \sim q(\cdot|\tilde{\mathbf{X}}U^\top), Z} [\mathcal{L}(\hat{y}(\mathbf{x}_{\text{te}}|\mathbf{X}, \mathbf{y}), y_{\text{te}})] \\ &= \mathbb{E}_{\tilde{\mathbf{X}}' \sim p_{\text{in}}, \tilde{\mathbf{y}} \sim q(\cdot|\tilde{\mathbf{X}}'), Z} [\mathcal{L}(\hat{y}(\mathbf{x}'_{\text{te}}|\mathbf{X}', \mathbf{y}), y_{\text{te}})] \\ &= L_{\hat{y}}(q), \end{aligned}$$

where  $\tilde{\mathbf{X}}' = \tilde{\mathbf{X}}U^\top$  (we also used  $\mathbf{X}' = \mathbf{X}U^\top$  and  $\mathbf{x}'_{\text{te}} = U\mathbf{x}_{\text{te}}$ ), and where the fourth line uses rotational symmetry of  $p_{\text{in}}$  and rotational invariance of  $\hat{y}$ . Therefore, presented with the problem  $\hat{q}$ , the algorithm will achieve expected loss  $L_{\hat{y}}(q)$  regardless of the value of  $U$ . This implies that  $L_{\hat{y}}(q)$  cannot be less than the optimal value  $L_B(\hat{q})$ .  $\blacksquare$

## Appendix B. Proof of lower bound Theorem 2

Before we give the proof observe that one can equivalently think of the  $m$  stacked copies of  $\mathbf{H}$  as just a single copy of  $\mathbf{H}$  with the variance of label noise reduced by a factor of  $m$ . Indeed, if  $\mathbf{y}_i$  represents the label vector of size  $d$  associated with the  $i$ -th copy of  $\mathbf{H}$ , then the training loss for any weight vector  $\hat{\mathbf{w}}$  is effectively

$$\|\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}\|^2 = \sum_{i=1}^m \|\mathbf{y}_i - \mathbf{H}\hat{\mathbf{w}}\|^2 = m\|\bar{\mathbf{y}} - \mathbf{H}\hat{\mathbf{w}}\|^2 + \sum_{i=1}^m \|\mathbf{y}_i - \bar{\mathbf{y}}\|^2,$$

where  $\bar{\mathbf{y}} = m^{-1} \sum_i \mathbf{y}_i \sim \mathcal{N}(\mathbf{H}\mathbf{e}_1, \sigma^2/m\mathbf{I}_d)$  and the last term on the right-hand side does not depend on  $\hat{\mathbf{w}}$ . Therefore, by varying  $m$ , we effectively alter the level of noise the algorithms encounter. We also note that our model is equivalent to the Gaussian Sequence Model (see, for example, [Rigollet and Hütter \(2023\)](#)), given by  $\mathbf{y} = \boldsymbol{\theta} + \boldsymbol{\xi}$  (with  $\boldsymbol{\theta} = \mathbf{X}\mathbf{e}_1$  and  $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2/m\mathbf{I}_d)$ ), a commonly used model for analyzing nonparametric and high dimensional statistical problems.

**Proof** We consider the rotated observational model described in the main text, which can be defined as follows. Let  $\mathbf{w} \in \mathbb{R}^d$  be a weight vector drawn uniformly from a unit sphere  $\mathcal{S}^{d-1} = \{\mathbf{w} \in \mathbb{R}^d: \|\mathbf{w}\| = 1\}$ . The algorithm is given data set  $(\mathbf{X}, \mathbf{y})$  with  $\mathbf{X} = [\mathbf{H}; \dots; \mathbf{H}]$  being  $m$  copies

of  $\mathbf{H} = \sqrt{d}\mathbf{V}$ , and  $\mathbf{y} = \mathbf{H}\mathbf{w} + \boldsymbol{\xi}$ , where  $\boldsymbol{\xi} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_{dm})$  is a vector of Gaussian i.i.d. noise variables, each having zero mean and variance  $\sigma^2$ . Given  $\mathbf{y}$ , the algorithm is supposed to produce a vector of predictions  $\hat{\mathbf{y}} \in \mathbb{R}^d$  and is evaluated by means of the squared error,  $e(\hat{\mathbf{y}}|\mathbf{w}) = \frac{1}{d} \|\hat{\mathbf{y}} - \mathbf{H}\mathbf{w}\|^2$ . We first note that without loss of generality, the algorithm produces a weight vector  $\hat{\mathbf{w}}$ , based on which the predictions are generated,  $\hat{\mathbf{y}} = \mathbf{H}\hat{\mathbf{w}}$ ; this is due to the fact that  $\mathbf{H}$  is invertible (as multiplicity of an orthogonal matrix), so for every  $\hat{\mathbf{y}}$ , one can have a corresponding weight vector  $\hat{\mathbf{w}} = \mathbf{H}^{-1}\hat{\mathbf{y}}$ . Thus, using (3) the error can be equivalently written as

$$e(\hat{\mathbf{w}}|\mathbf{w}) = \|\hat{\mathbf{w}} - \mathbf{w}\|^2.$$

It is well-known (see, e.g., [Berger \(1985\)](#)) that the expected squared error,  $\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}|\mathbf{w})]$  (with expectation with respect to the prior and the label noise) is minimized by the *posterior mean*  $\hat{\mathbf{w}}^* = \mathbb{E}_{\mathbf{w}|\mathbf{y}}[\mathbf{w}]$ , that is the mean value of  $\mathbf{w}$  with respect to the posterior distribution  $q(\mathbf{w}|\mathbf{X}, \mathbf{y})$ . Even though the posterior mean does not have a nice analytic form, we can still lower bound its expected squared error using a technique borrowed from ([Marchand, 1993](#); [Dicker, 2016](#)). Let  $\hat{\mathbf{w}}_{RR}$  be the ridge regression estimator:

$$\hat{\mathbf{w}}_{RR} = (\mathbf{X}^\top \mathbf{X} + \sigma^2 d \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y},$$

which is the posterior mean itself (and thus optimal) when the prior over  $\mathbf{w}$  is Gaussian with zero mean and covariance  $\frac{1}{d} \mathbf{I}$  (the covariance is multiplied by factor  $d^{-1}$  to have  $\mathbb{E}[\|\mathbf{w}\|^2] = \mathbb{E}[\mathbf{w}\mathbf{w}^\top] = \text{tr}(\frac{1}{d} \mathbf{I}) = 1$  as in the unit sphere prior case). Even though the Gaussian prior differs from the uniform prior over a unit sphere, it turns out that the RR predictor has error only slightly larger than that of the optimal Bayes predictor  $\hat{\mathbf{w}}^*$ :

**Lemma 6**

$$\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}_{RR}|\mathbf{w})] \leq \mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}^*|\mathbf{w})] + \frac{1}{d} \frac{\sigma^2}{\sigma^2 + m}.$$

**Proof** [Dicker \(2016\)](#) considered a Bayesian setting similar to ours, with  $\sigma^2 = 1$  and  $\mathbf{w}$  distributed uniformly over  $\tau \mathcal{S}^{d-1} = \{\mathbf{w} : \|\mathbf{w}\| = \tau\}$ . To account for this setting, we note that in our setup,

$$\sigma^{-1} \mathbf{y} = \mathbf{X}^\top (\sigma^{-1} \mathbf{w}) + \underbrace{\sigma^{-1} \boldsymbol{\xi}}_{\sim N(\mathbf{0}, \mathbf{I}_{dm})},$$

so that we can set  $\tau = \sigma^{-1}$  and assume unit variance of the noise. Their ‘oracle ridge estimator’ is thus  $\hat{\mathbf{w}}_{RR}$ . Furthermore, since  $\|\hat{\mathbf{w}} - \mathbf{w}\|^2 = \sigma^2 \|\sigma^{-1} \hat{\mathbf{w}} - \sigma^{-1} \mathbf{w}\|^2$ , we need to multiply their bound by  $\sigma^2$ . We use their Theorem 2 (adapted to the modifications stated above):

**Theorem 7** (Theorem 2 by [Dicker \(2016\)](#), Theorem 3.1 by [Marchand \(1993\)](#)) *Let  $n = md$  and let  $s_1 \geq \dots \geq s_d$  denote the eigenvalues of  $n^{-1} \mathbf{X}^\top \mathbf{X}$ . Then*

$$\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}_{RR}|\mathbf{w})] \leq \mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}^*|\mathbf{w})] + \frac{\sigma^2}{d} \frac{s_1}{s_d} \text{tr} \left\{ (\mathbf{X}^\top \mathbf{X} + d\sigma^2 \mathbf{I}_n)^{-1} \right\}.$$

Since  $\mathbf{X}^\top \mathbf{X} = n \mathbf{I}$ , we have  $s_1 = s_d = 1$  and  $(\mathbf{X}^\top \mathbf{X} + \mathbf{I}_n)^{-1} = (n + d\sigma^2)^{-1} \mathbf{I}_n$ , and thus,

$$\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}_{RR}|\mathbf{w})] \leq \mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}^*|\mathbf{w})] + \frac{\sigma^2}{n + d\sigma^2} = \mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}^*|\mathbf{w})] + \frac{1}{d} \frac{\sigma^2}{m + \sigma^2}. \quad \blacksquare$$

Now, we compute the expected error of  $\hat{\mathbf{w}}_{RR}$ . Since  $\mathbf{X}^\top \mathbf{X} = md\mathbf{I}$ , we get

$$\hat{\mathbf{w}}_{RR} = \frac{1}{md + \sigma^2 d} \mathbf{X}^\top \mathbf{y} = \frac{1}{md + \sigma^2 d} \mathbf{X}^\top (\mathbf{X}\mathbf{w} + \boldsymbol{\xi}) = \frac{md\mathbf{w} + \mathbf{X}^\top \boldsymbol{\xi}}{md + \sigma^2 d},$$

and thus

$$\begin{aligned} e(\hat{\mathbf{w}}_{RR}|\mathbf{w}) &= \|\hat{\mathbf{w}}_{RR} - \mathbf{w}\|^2 = \left\| \frac{\mathbf{X}^\top \boldsymbol{\xi} - \sigma^2 d \mathbf{w}}{md + \sigma^2 d} \right\|^2 \\ &= \frac{\|\mathbf{X}^\top \boldsymbol{\xi}\|^2}{(md + \sigma^2 d)^2} - \frac{md\mathbf{w}^\top \mathbf{X}^\top \boldsymbol{\xi}}{(md + \sigma^2 d)^2} + \frac{\sigma^4 d^2 \|\mathbf{w}\|^2}{(md + \sigma^2 d)^2}. \end{aligned}$$

We take the expectation over  $\mathbf{w}$ , under which the middle term in the last line vanishes (as  $\mathbb{E}[\mathbf{w}] = \mathbf{0}$  over a unit sphere) and use  $\|\mathbf{w}\| = 1$  to get

$$\mathbb{E}_{\mathbf{w}}[e(\hat{\mathbf{w}}_{RR}|\mathbf{w})] = \frac{\|\mathbf{X}^\top \boldsymbol{\xi}\|^2}{(md + \sigma^2 d)^2} + \frac{\sigma^4 d^2}{(md + \sigma^2 d)^2}.$$

We further take an expectation over  $\boldsymbol{\xi}$  and use

$$\mathbb{E}[\|\mathbf{X}^\top \boldsymbol{\xi}\|^2] = \mathbb{E}[\text{tr}(\mathbf{X}^\top \boldsymbol{\xi} \boldsymbol{\xi}^\top \mathbf{X})] = \text{tr}(\mathbf{X}^\top \mathbb{E}[\boldsymbol{\xi} \boldsymbol{\xi}^\top] \mathbf{X}) = \text{tr}(\mathbf{X}^\top \mathbf{X}) = md \text{tr}(\mathbf{I}) = md^2,$$

to get:

$$\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}_{RR}|\mathbf{w})] = \frac{md^2}{(md + \sigma^2 d)^2} + \frac{\sigma^4 d^2}{(md + \sigma^2 d)^2} = \frac{\sigma^2 d(md + \sigma^2 d)}{(md + \sigma^2 d)^2} = \frac{\sigma^2}{m + \sigma^2}.$$

Using this together with Lemma 6 gives the lower bound on the Bayes optimal predictor in the rotated observational model.

$$\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}}[e(\hat{\mathbf{w}}^*|\mathbf{w})] \geq \frac{d-1}{d} \frac{\sigma^2}{\sigma^2 + m}.$$

We now use the fact that that  $\mathbf{H} = \sqrt{d}\mathbf{V}$  with orthogonal matrix  $\mathbf{V}$  of size  $d \times d$ , drawn uniformly at random (with respect to Haar measure), so that our input distribution is rotation symmetric. This means that we can apply Theorem 1 and conclude that any rotation invariant algorithm has the expected error at least  $\frac{d-1}{d} \frac{\sigma^2}{\sigma^2 + m}$  on the original problem, that is for  $\mathbf{w} = \mathbf{e}_1$ .  $\blacksquare$

Note that the proof would significantly simplify if we assumed from the start that the target weight vector  $\mathbf{w}$  is generated from a Gaussian distribution  $N(\mathbf{0}, \frac{1}{d}\mathbf{I}_d)$  rather than from a unit sphere  $\mathcal{S}^{d-1}$  (both priors give unit squared norm of  $\mathbf{w}$  on expectation), as the Bayes predictor would be exactly the RR predictor, giving even a better lower bound of  $\frac{\sigma^2}{\sigma^2 + m}$ , without the need to apply results from Marchand (1993); Dicker (2016). This would, however, result in a random norm of the sparse target vector. In our proof we opted for a bound with a fixed, unit norm of  $\mathbf{w}$ .

### Appendix C. Lower bound for i.i.d. sampling

Instead of observing copies of the same matrix  $\mathbf{H}$ , one can instead consider a standard linear model framework, where individual inputs  $\mathbf{x}$  are drawn i.i.d. from some input distribution  $p_{\text{in}}(\mathbf{x})$ . The associated labels are given by  $y = \mathbf{x}_i^\top \mathbf{e}_1 + \xi$ , where  $\xi \sim \mathcal{N}(0, \sigma^2)$ . The algorithm observes  $n$  such i.i.d. samples  $(\mathbf{X}, \mathbf{y})$  and produces a weight vector  $\hat{\mathbf{w}}$ , which is then evaluated by the squared error  $\|\hat{\mathbf{w}} - \mathbf{e}_1\|^2$ . Assume  $p_{\text{in}}(\mathbf{x})$  is rotationally symmetric with covariance  $\mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \mathbf{I}_d$  (so that the expected input size matches that from our previous setup); for instance, we could have  $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ . Employing our Bayesian argument from section 1.2, one can establish a similar lower bound for any rotation-invariant algorithm. Note that for large  $d$ , the bound is essentially the same as that of Theorem 2. The proof makes use of information-theoretic arguments by Gerchinovitz et al. (2020).

**Theorem 8** *Let the rows of  $\mathbf{X}$  be drawn i.i.d. from rotationally symmetric  $p_{\text{in}}$  with covariance  $\mathbf{I}_d$ , and the labels given by  $\mathbf{y} = \mathbf{X}\mathbf{e}_1 + \boldsymbol{\xi}$ ,  $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ . Assume  $d \geq 12$ . Then, the expected error (with a random choice of the sample) of any rotation-invariant learning algorithm is at least*

$$\mathbb{E}[e(\hat{\mathbf{w}})] \geq \left( \frac{\sigma^2}{e(\sigma^2 + m/2)} \right)^{d/(d-1)}.$$

**Proof** We consider a Bayesian setup, in which the target weight vector is drawn uniformly from a unit sphere,  $\mathbf{w} \sim \mathcal{S}^{d-1}$ , with  $\mathcal{S}^{d-1} = \{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\| = 1\}$ . We will show a lower bound on the error of any predictor  $\hat{\mathbf{w}}$ ,  $\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}} [\|\hat{\mathbf{w}} - \mathbf{w}\|^2]$ , by exploiting the fact that it is not less than the error of the Bayes optimal predictor  $\hat{\mathbf{w}}^*$  (the posterior mean), followed by a lower bound on the error of  $\hat{\mathbf{w}}^*$ :

$$\mathbb{E}_{\mathbf{w}, \boldsymbol{\xi}} [\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2] \geq \left( \frac{\sigma^2}{e(\sigma^2 + m/2)} \right)^{d/(d-1)}. \quad (14)$$

Using the rotational symmetry of the input distribution  $p_{\text{in}}$ , and giving the same arguments as in the proof of Theorem 2 (or Section 1.2), this will imply the same lower bound (14) for any rotation-invariant algorithm with a sparse target vector  $\mathbf{w} = \mathbf{e}_1$ .

Unfortunately, for prior uniform over a sphere,  $\hat{\mathbf{w}}^*$  does not have a closed form. We will, however, use a continuous version of Fano's inequality (Gerchinovitz et al., 2020) to lower bound its Bayes risk.

From now on, we condition our analysis on  $\mathbf{X}$  drawn i.i.d. from the input distribution. We lower-bound the Bayes risk using Markov's inequality: for any  $\epsilon > 0$ ,

$$\mathbb{E}[\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2] \geq \epsilon P(\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2 \geq \epsilon) = \epsilon (1 - P(\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2 < \epsilon)). \quad (15)$$

where the expectation and the probability is with respect to random  $\mathbf{y}, \mathbf{w}$  (conditioned on  $\mathbf{X}$ ). Let  $P_{\mathbf{w}} = \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}_n)$  be the distribution of  $\mathbf{y}$  given  $\mathbf{w}$  (and  $\mathbf{X}$ ). Given  $\mathbf{w}$ , define event  $\mathcal{A}_{\mathbf{w}} = \{\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2 < \epsilon\}$ . We can write

$$P(\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2 < \epsilon) = \mathbb{E}_{\mathbf{w}} [P_{\mathbf{w}}(\mathcal{A}_{\mathbf{w}})]$$

Using techniques from Gerchinovitz et al. (2020), we have for any distribution  $Q$  over  $\mathbf{y}$ ,

$$\mathbb{E}_{\mathbf{w}} [P_{\mathbf{w}}(\mathcal{A}_{\mathbf{w}})] \leq \mathbb{E}_{\mathbf{w}} [Q(\mathcal{A}_{\mathbf{w}})] + \sqrt{(\mathbb{E}_{\mathbf{w}} [Q(\mathcal{A}_{\mathbf{w}})]) (\mathbb{E}_{\mathbf{w}} [\chi^2(P_{\mathbf{w}}, Q)])} \quad (16)$$

where  $\chi^2(P, Q) = \mathbb{E}_Q \left[ \left( \frac{dP}{dQ} \right)^2 \right]$  is the  $\chi^2$ -divergence between  $P$  and  $Q$ . We now choose  $Q = \mathcal{N}(\mathbf{0}, \Sigma)$  with  $\Sigma = \sigma^2 \mathbf{I}_n + \frac{1}{d} \mathbf{X} \mathbf{X}^\top$ . It follow from the Gaussian integration that

$$\chi^2(P_w, Q) = \sqrt{\det(\sigma^{-2} \Sigma^2 (2\Sigma - \sigma^2 \mathbf{I}_n)^{-1})} e^{\mathbf{w}^\top \mathbf{X}^\top (2\Sigma - \sigma^2 \mathbf{I}_n)^{-1} \mathbf{X} \mathbf{w}} - 1.$$

First note that

$$\sigma^{-2} \Sigma^2 (2\Sigma - \sigma^2 \mathbf{I}_n)^{-1} = \left( \mathbf{I}_n + \frac{1}{d\sigma^2} \mathbf{X} \mathbf{X}^\top \right)^2 \left( \mathbf{I}_n + \frac{2}{d\sigma^2} \mathbf{X} \mathbf{X}^\top \right)^{-1} \preceq \mathbf{I}_n + \frac{1}{2d\sigma^2} \mathbf{X} \mathbf{X}^\top.$$

Indeed, by taking any eigenvalue  $\lambda_i$  of  $\frac{1}{2d\sigma^2} \mathbf{X} \mathbf{X}^\top$ ,

$$\frac{(1 + \lambda_i)^2}{1 + 2\lambda_i} \leq 1 + \frac{\lambda_i^2}{1 + 2\lambda_i} \leq 1 + \frac{\lambda_i}{2}.$$

Moreover, from the property of the determinant, that  $\det(\mathbf{I}_m + \mathbf{A}\mathbf{B}) = \det(\mathbf{I}_n + \mathbf{B}\mathbf{A})$  for  $\mathbf{A}$  of size  $m \times n$  and  $\mathbf{B}$  of size  $n \times m$ , we get

$$\det \left( \mathbf{I}_n + \frac{1}{2d\sigma^2} \mathbf{X} \mathbf{X}^\top \right) = \det \left( \mathbf{I}_m + \frac{1}{2d\sigma^2} \mathbf{X}^\top \mathbf{X} \right).$$

Finally, note that

$$\mathbf{X}^\top (2\Sigma - \sigma^2 \mathbf{I}_n)^{-1} \mathbf{X} = \mathbf{X}^\top \left( \sigma^2 \mathbf{I}_n + \frac{2}{d} \mathbf{X} \mathbf{X}^\top \right)^{-1} \mathbf{X} \preceq \frac{d}{2} \mathbf{X}^\top (\mathbf{X} \mathbf{X}^\top)^\dagger \mathbf{X} = \frac{d}{2} \mathbf{I}_d,$$

so that

$$e^{\mathbf{w}^\top \mathbf{X}^\top (2\Sigma - \sigma^2 \mathbf{I}_n)^{-1} \mathbf{X} \mathbf{w}} \leq e^{\frac{d}{2} \|\mathbf{w}\|^2} = e^{\frac{d}{2}}.$$

Thus, we get:

$$\chi^2(P_w, Q) \leq \sqrt{\det \left( \mathbf{I}_m + \frac{1}{2d\sigma^2} \mathbf{X}^\top \mathbf{X} \right)} e^{\frac{d}{2}}$$

For any positive-definite  $d \times d$  matrix  $\mathbf{A}$  with eigenvalues  $a_1, \dots, a_d$ , we have  $\ln \det \mathbf{A} = \sum_i \ln a_i$ . Using the concavity of negative logarithm, we can bound:

$$\ln \det \mathbf{A} = d \frac{1}{d} \sum_i \ln(a_i) \leq d \ln \left( \frac{1}{d} \sum_i a_i \right) = d \ln \text{tr}(\mathbf{A}/d),$$

which gives  $\det \mathbf{A} \leq \text{tr}(\mathbf{A}/d)^d$  and thus

$$\chi^2(P_w, Q) \leq \left( e \text{tr} \left( \frac{1}{d} \mathbf{I}_m + \frac{1}{2d^2\sigma^2} \mathbf{X}^\top \mathbf{X} \right) \right)^{\frac{d}{2}}.$$

Since this expression does not depend on  $\mathbf{w}$ , taking expectation on both sides (with respect to  $\mathbf{w}$ ) gives the same bound on  $\mathbb{E}_{\mathbf{w}} [\chi^2(P_w, Q)]$ .

We will now upper-bound  $\mathbb{E}_{\mathbf{w}}[Q(\mathcal{A}_{\mathbf{w}})]$ . Let  $\mathbf{1}\{C\}$  denote the Iverson bracket which is 1 if  $C$  holds and 0 otherwise. Using the fact that  $\hat{\mathbf{w}}^* = \hat{\mathbf{w}}^*(\mathbf{y})$  depends on  $\mathbf{y}$  (and  $\mathbf{X}$ ), but not on  $\mathbf{w}$ , and that  $\mathbf{y} \sim Q$  is independent of  $\mathbf{w}$ , we have

$$\begin{aligned} \mathbb{E}_{\mathbf{w}}[Q(\mathcal{A}_{\mathbf{w}})] &= \mathbb{E}_{\mathbf{w}}[Q(\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2 < \epsilon)] = \mathbb{E}_{\mathbf{w} \sim \mathcal{S}^{d-1}, \mathbf{y} \sim Q} [\mathbf{1}\{\|\hat{\mathbf{w}}^*(\mathbf{y}) - \mathbf{w}\|^2 < \epsilon\}] \\ &= \mathbb{E}_{\mathbf{y} \sim Q} [\mathbb{E}_{\mathbf{w} \sim \mathcal{S}^{d-1}} [\mathbf{1}\{\|\hat{\mathbf{w}}^*(\mathbf{y}) - \mathbf{w}\|^2 < \epsilon\}]] \\ &\leq \mathbb{E}_{\mathbf{y} \sim Q} \left[ \max_{\hat{\mathbf{w}}} \mathbb{E}_{\mathbf{w} \sim \mathcal{S}^{d-1}} [\mathbf{1}\{\|\hat{\mathbf{w}} - \mathbf{w}\|^2 < \epsilon\}] \right] = \max_{\hat{\mathbf{w}}} \mathbb{E}_{\mathbf{w}} [\mathbf{1}\{\|\hat{\mathbf{w}} - \mathbf{w}\|^2 < \epsilon\}], \end{aligned}$$

where in the last equality we used the fact that the term inside the expectation over  $\mathbf{y}$  does not depend on  $\mathbf{y}$ . Let us now fix  $\hat{\mathbf{w}}$  and we will bound  $\mathbb{E}_{\mathbf{w}} [\mathbf{1}\{\|\hat{\mathbf{w}} - \mathbf{w}\|^2 < \epsilon\}]$ . Since the distribution of  $\mathbf{w}$  is rotation-invariant (uniform over a sphere), without loss of generality take  $\hat{\mathbf{w}} = c\mathbf{e}_1$  with  $c > 0$  ( $c = 0$  would give 0 for any  $\epsilon < 1$ ). We have

$$\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2 = (c - w_1)^2 + \sum_{j=2}^d w_j^2 = (c - w_1)^2 + 1 - w_1^2 = 1 + c^2 - 2cw_1,$$

and thus

$$\mathbb{E}_{\mathbf{w}} [\mathbf{1}\{\|\hat{\mathbf{w}} - \mathbf{w}\|^2 < \epsilon\}] = \mathbb{E}_{\mathbf{w}} [\mathbf{1}\{1 + c^2 - 2cw_1 < \epsilon\}] = P\left(w_1 > \frac{1 + c^2 - \epsilon}{2c}\right).$$

Since the probability is nonincreasing in  $\frac{1+c^2-\epsilon}{2c}$  we take  $c$  which minimizes this quantity to get an upper bound. By inspecting the derivative

$$\min_c \frac{1 + c^2 - \epsilon}{2c} = \frac{1}{2} \min_c c + \frac{1 - \epsilon}{c} \stackrel{c=\sqrt{1-\epsilon}}{=} \sqrt{1 - \epsilon},$$

we have

$$\mathbb{E}_{\mathbf{w}} [\mathbf{1}\{\|\hat{\mathbf{w}} - \mathbf{w}\|^2 < \epsilon\}] = P(w_1 > \sqrt{1 - \epsilon}) = \frac{1}{2} P(w_1^2 > 1 - \epsilon),$$

where we used the fact that distribution of  $w_1$  is symmetric around zero. To determine the distribution of  $w_1^2$  we use the fact that drawing  $\mathbf{w}$  uniformly over a unit sphere amounts to drawing  $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$  and setting  $\mathbf{w} = \frac{\mathbf{u}}{\|\mathbf{u}\|}$ . Thus,  $w_1^2 = \frac{x}{x+y}$ , where  $x \sim \chi^2(1) = \text{Gamma}(\alpha = 1/2, \beta = 1/2)$  and  $y \sim \chi^2(d-1) = \text{Gamma}(\alpha = (d-1)/2, \beta = 1/2)$  and  $x, y$  independent. Thus,  $w_1^2$  is distributed according to beta distribution  $B(\alpha = 1/2, \beta = (d-1)/2)$ . This gives

$$\begin{aligned} P(w_1^2 > 1 - \epsilon) &= \frac{\Gamma(d/2)}{\Gamma(1/2)\Gamma((d-1)/2)} \int_{1-\epsilon}^1 x^{-1/2}(1-x)^{-(d-1)/2} dx \\ &\leq \frac{\Gamma(d/2)}{\Gamma(1/2)\Gamma((d-1)/2)} (1-\epsilon)^{-1/2} \int_{1-\epsilon}^1 (1-x)^{(d-1)/2-1} dx \\ &= \frac{\Gamma(d/2)}{\Gamma(1/2)\Gamma((d-1)/2)} \frac{2(1-\epsilon)^{-1/2}\epsilon^{(d-1)/2}}{d-1} \end{aligned}$$

Assuming  $\epsilon < 1/2$ , we have  $2(1-\epsilon)^{-1/2} \leq 2\sqrt{2}$ . We also use Gautschi's inequality which states that for any  $x > 0$  and any  $s \in (0, 1)$ ,  $\frac{\Gamma(x+1)}{\Gamma(x+s)} \leq (x+1)^{1-s}$ . Taking  $x = \frac{d}{2} - 1$  and  $s = \frac{1}{2}$ , we can



bound  $\frac{\Gamma(d/2)}{\Gamma((d-1)/2)} \leq \sqrt{d/2} \leq \sqrt{d-1}$ , where we used  $d \geq 2$ . This, together with  $\Gamma(1/2) = \sqrt{\pi}$  allows us to bound

$$P(w_1^2 > 1 - \epsilon) \leq \frac{2\sqrt{2}\epsilon^{(d-1)/2}}{\sqrt{\pi}\sqrt{d-1}}.$$

Thus,  $\mathbb{E}_{\mathbf{w}}[Q(\mathcal{A}_{\mathbf{w}})]$  can be bounded by

$$\mathbb{E}_{\mathbf{w}}[Q(\mathcal{A}_{\mathbf{w}})] = \frac{1}{2}P(w_1^2 > 1 - \epsilon) \leq \frac{\sqrt{2}\epsilon^{(d-1)/2}}{\sqrt{\pi}\sqrt{d-1}}$$

Plugging all bounds to (16) gives:

$$\mathbb{E}_{\mathbf{w}}[P_{\mathbf{w}}(\mathcal{A}_{\mathbf{w}})] \leq \frac{\sqrt{2}\epsilon^{(d-1)/2}}{\sqrt{\pi(d-1)}} + \sqrt{\sqrt{2}\frac{\epsilon^{(d-1)/2}}{\sqrt{\pi(d-1)}} \left( e \operatorname{tr} \left( \frac{1}{d}\mathbf{I}_m + \frac{1}{2d^2\sigma^2}\mathbf{X}^\top\mathbf{X} \right) \right)^{\frac{d}{2}}}.$$

Now, take

$$\epsilon^{-1} = \left( e \operatorname{tr} \left( \frac{1}{d}\mathbf{I}_m + \frac{1}{2d^2\sigma^2}\mathbf{X}^\top\mathbf{X} \right) \right)^{d/(d-1)}.$$

Note that  $\epsilon \leq 1/e$ . This gives

$$\mathbb{E}_{\mathbf{w}}[P_{\mathbf{w}}(\mathcal{A}_{\mathbf{w}})] \leq \frac{\sqrt{2}e^{-(d-1)/2}}{\sqrt{\pi(d-1)}} + \sqrt{\frac{\sqrt{2}}{\sqrt{\pi(d-1)}}} \leq \frac{1}{2},$$

whenever  $d \geq 12$ . We thus have shown

$$P(\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2 \leq \epsilon) \leq \frac{1}{2},$$

which, using (15), gives:

$$\mathbb{E}_{\mathbf{w}, \mathbf{y} | \mathbf{X}}[\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2] \geq \frac{\epsilon}{2}.$$

To get a bound with respect to a random choice of  $\mathbf{X}$ , note that function  $x \mapsto x^{-d/(d-1)}$  is convex, and thus:

$$\begin{aligned} \mathbb{E}[\|\hat{\mathbf{w}}^* - \mathbf{w}\|^2] &\geq \frac{1}{2} \mathbb{E}[\epsilon] \geq \frac{1}{2} \left( e \operatorname{tr} \left( \frac{1}{d}\mathbf{I}_m + \frac{1}{2d^2\sigma^2} \mathbb{E}[\mathbf{X}^\top\mathbf{X}] \right) \right)^{d/(d-1)} \\ &= \left( e \left( 1 + \frac{n\tau^2}{2d\sigma^2} \right) \right)^{-d/(d-1)} = \left( \frac{\sigma^2}{e(\sigma^2 + n\tau^2/(2d))} \right)^{d/(d-1)} \end{aligned}$$

■

### Appendix D. Upper bound for $\text{EG}^\pm$ : Proof of Theorem 3

Here we prove that the batch version of  $\text{EG}^\pm$  achieves small error already after one trial, when the learning rate is set to a sufficiently large value.

The (batch)  $\text{EG}^\pm$  algorithm keeps track of two vectors,  $\mathbf{v}_t^+$  and  $\mathbf{v}_t^-$ , and its prediction vector is given by  $\mathbf{w}_t = \mathbf{v}_t^+ - \mathbf{v}_t^-$ . It starts with a set of weights  $\mathbf{v}_1^+ = \mathbf{v}_1^- = \frac{1}{2d}\mathbf{1}$ , and updates according to

$$\mathbf{v}_{t+1}^+ \propto \mathbf{v}_t^+ \odot e^{-\eta \nabla L(\mathbf{w}_t)}, \quad \mathbf{v}_{t+1}^- \propto \mathbf{v}_t^- \odot e^{\eta \nabla L(\mathbf{w}_t)},$$

where  $\odot$  is component-wise multiplication, and the normalization ensures that  $\|\mathbf{v}_{t+1}^+\|_1 + \|\mathbf{v}_{t+1}^-\|_1 = 1$ , while  $L(\mathbf{w})$  is the average total loss on the training sample:

$$L(\mathbf{w}) = \frac{1}{dm} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2.$$

We compute the expression for the gradient:

$$\begin{aligned} \nabla L(\mathbf{w}) &= \frac{2}{dm} \sum_{t=1}^m \sqrt{d} \mathbf{V}^\top (\sqrt{d} \mathbf{V} \mathbf{w} - \mathbf{y}_t) = \frac{2}{dm} \sum_{t=1}^m \sqrt{d} \mathbf{V}^\top (\sqrt{d} \mathbf{V} (\mathbf{w} - \mathbf{e}_1) - \boldsymbol{\xi}_t) \\ &= \frac{2}{dm} \sum_{t=1}^m \left( d(\mathbf{w} - \mathbf{e}_1) + \sqrt{d} \mathbf{V}^\top \boldsymbol{\xi}_t \right) = 2(\mathbf{w} - \mathbf{e}_1) - \frac{2}{\sqrt{d}} \mathbf{V}^\top \bar{\boldsymbol{\xi}}, \end{aligned}$$

where

$$\bar{\boldsymbol{\xi}} = \frac{1}{m} \sum_{t=1}^m \boldsymbol{\xi}_t \sim N\left(\mathbf{0}, \frac{\sigma^2}{m} \mathbf{I}\right),$$

and the reduction of variance is due to averaging i.i.d. noise variables. Furthermore, we rewrite

$$\nabla L(\mathbf{w}) = 2(\mathbf{w} - \mathbf{e}_1 - \boldsymbol{\zeta}), \tag{17}$$

where the noise vector  $\boldsymbol{\zeta} = \frac{1}{\sqrt{d}} \mathbf{V}^\top \bar{\boldsymbol{\xi}}$  has distribution

$$\boldsymbol{\zeta} \sim N\left(\mathbf{0}, \frac{\sigma^2}{md} \mathbf{V}^\top \mathbf{V}\right) = N\left(\mathbf{0}, \frac{\sigma^2}{md} \mathbf{I}\right).$$

We also bound the error of the algorithm from above:

$$e(\mathbf{w}_t) = \|\mathbf{w}_t - \mathbf{e}_1\|^2 = \|\mathbf{w}_t\|^2 - 2\mathbf{w}_t^\top \mathbf{e}_1 + \|\mathbf{e}_1\|^2 \leq 2 - 2\mathbf{w}_t^\top \mathbf{e}_1 = 2(1 - w_{t,1}) = 2(1 - v_{t,1}^+ + v_{t,1}^-).$$

So it suffices to upper-bound  $1 - v_{t,1}^+$  and  $v_{t,1}^-$ .

Consider the weights of the batch  $\text{EG}^\pm$  algorithm after just a *single* trial, that is  $\mathbf{v}_2^+$  and  $\mathbf{v}_2^-$ . Using (17) and noting that  $\mathbf{w}_1 = \mathbf{0}$ , and  $v_{1,i}^+ = v_{1,i}^- = \frac{1}{2d}$  for all  $i$ , we can concisely write  $v_{2,1}^+$  and  $v_{2,1}^-$  as:

$$v_{2,1}^+ = \frac{e^{2\eta(1+\zeta_1)}}{Z_2}, \quad v_{2,1}^- = \frac{e^{-2\eta(1+\zeta_1)}}{Z_2}, \quad Z_2 = \sum_{i=1}^d e^{2\eta(\delta_{1i}+\zeta_i)} + e^{-2\eta(\delta_{1i}+\zeta_i)}, \tag{18}$$

We will now lower-bound  $v_{2,1}^+$ , and later use a relation which directly follows from (18):

$$v_{2,1}^- = e^{-4\eta(1+\zeta_1)} v_{2,1}^+. \quad (19)$$

Using the deviation bound for zero-mean Gaussian  $z \sim N(0, \tau^2)$ ,  $P(|z| \geq \gamma) \leq 2 \exp \left\{ -\frac{\gamma^2}{2\tau^2} \right\}$ , we get that for any  $i = 1, \dots, d$ ,

$$P(|\zeta_i| \geq 1/4) \leq 2 \exp \left\{ -\frac{md\gamma^2}{32\sigma^2} \right\}.$$

Taking the union bound over  $i = 1, \dots, d$ , we have

$$P(\exists i \ |\zeta_i| \geq 1/4) \leq 2d \exp \left\{ -\frac{md\gamma^2}{32\sigma^2} \right\}.$$

Denoting the probability on the right-hand side by  $\delta$ , we conclude that with probability at least  $1 - \delta$ , all noise variables  $\zeta_i$  are bounded by  $1/4$ . Let us call this event  $E$ , and we condition everything that follows on the fact that  $E$  happened.

Note that for any  $i \geq 2$ ,

$$\frac{\partial v_{2,1}^+}{\partial \zeta_i} = -\frac{v_{2,1}^+}{Z_2} 2\eta \left( e^{2\eta\zeta_i} - e^{-2\eta\zeta_i} \right),$$

which is decreasing for  $\zeta_i > 0$  and increasing for  $\zeta_i < 0$ . So, to lower-bound  $v_{2,1}^+$ , conditioning on event  $E$ , we set  $\zeta_i = 1/4$  for all  $i \geq 2$  in (18) ( $\zeta_i = -1/4$  would result in the same value of these weights). This gives:

$$\begin{aligned} v_{2,1}^+ &\geq \frac{e^{2\eta(1+\zeta_1)}}{e^{2\eta(1+\zeta_1)} + e^{-2\eta(1+\zeta_1)} + (d-1)(e^{\eta/2} + e^{-\eta/2})} \\ &= \frac{1}{1 + e^{-4\eta(1+\zeta_1)} + e^{-2\eta(1+\zeta_1)}(d-1)(e^{\eta/2} + e^{-\eta/2})} \\ &\geq \frac{1}{1 + e^{-4\eta(1-1/4)} + e^{-2\eta(1-1/4)}(d-1)(e^{\eta/2} + e^{-\eta/2})} \\ &\geq \frac{1}{1 + e^{-3\eta} + e^{-3/2\eta}2(d-1)e^{\eta/2}} \\ &\geq \frac{1}{1 + e^{-3\eta} + 2(d-1)e^{-\eta}} \geq \frac{1}{1 + (2d-1)e^{-\eta}} \end{aligned} \quad (20)$$

This gives:

$$1 - v_{2,1}^+ \leq \frac{(2d-1)e^{-\eta}}{1 + (2d-1)e^{-\eta}} = \frac{1}{1 + e^{\eta}/(2d-1)} \leq (2d-1)e^{-\eta}.$$

To upper-bound  $v_{2,1}^-$ , we use (19). Conditioning on  $E$ ,

$$v_{2,1}^- = e^{-4\eta(1+\zeta_1)} v_{2,1}^+ \leq e^{-4\eta(1+\zeta_1)} \leq e^{-2\eta}.$$

Thus, with probability at least  $1 - \delta$ , the error can be bounded by:

$$e(\mathbf{w}_2) \leq 2(1 - v_{2,1}^+ + v_{2,1}^-) \leq (2d-1)e^{-\eta} + e^{-2\eta} \leq 2de^{-\eta}$$

To get the expected error (with respect to the training data), we bound

$$\begin{aligned}\mathbb{E}[e(\mathbf{w}_2)] &= \mathbb{E}[e(\mathbf{w}_2)|E]P(E) + \mathbb{E}[e(\mathbf{w}_2)|E']P(E') \leq \mathbb{E}[e(\mathbf{w}_2)|E] + \delta \mathbb{E}[e(\mathbf{w}_2)|E'] \\ &\leq \mathbb{E}[e(\mathbf{w}_2)|E] + 2\delta = 2de^{-\eta} + 8de^{-\frac{md}{32\sigma^2}},\end{aligned}$$

where we used the fact that  $e(\mathbf{w}_2) \leq 2(1 - v_{2,1}^+ + v_{2,1}^-) \leq 4$  as  $v_{2,1}^+, v_{2,1}^- \in [0, 1]$ , and that maximizing convex function  $e(\mathbf{w})$  give  $\mathbf{w}$ . Thus, taking sufficiently large  $\eta$ , we can drop the error arbitrarily close to  $8de^{-\frac{md}{32\sigma^2}}$ .  $\blacksquare$

## Appendix E. Upper bound for Approximated EGU $^\pm$ : Proof of Theorem 4

We derive the Approximated EGU $^\pm$  algorithm defined by (4) as a first-order approximation of the unnormalized Exponentiated Gradient update.

The vanilla EGU $^\pm$  algorithm keeps track of two vectors,  $\mathbf{v}_t^+$  and  $\mathbf{v}_t^-$ , and the prediction vector is given by  $\mathbf{w}_t = \mathbf{v}_t^+ - \mathbf{v}_t^-$ . Let  $\beta$  denote the initial value of weights, that is  $\mathbf{v}_1^+ = \mathbf{v}_1^- = \beta \mathbf{1}$ . The weights are updated according to

$$\mathbf{v}_{t+1}^\pm = \mathbf{v}_t^\pm \odot e^{\mp \eta \nabla L(\mathbf{w}_t)} = \beta e^{\mp \eta \sum_{j=1}^t \nabla L(\mathbf{w}_j)}. \quad (21)$$

At every timestamp we have  $\mathbf{v}_t^+ \mathbf{v}_t^- = \beta^2$ , which together with  $\mathbf{w}_t = \mathbf{v}_t^+ - \mathbf{v}_t^-$ , allows us to express  $\mathbf{v}_t^+$  and  $\mathbf{v}_t^-$  in terms of  $\mathbf{w}_t$ :

$$\mathbf{v}_t^+ = \frac{\mathbf{w}_t + \sqrt{\mathbf{w}_t^2 + 4\beta^2}}{2}, \quad \mathbf{v}_t^- = \frac{-\mathbf{w}_t + \sqrt{\mathbf{w}_t^2 + 4\beta^2}}{2}. \quad (22)$$

Expanding the EGU $^\pm$  update (21) in the learning rate we get

$$\mathbf{v}_{t+1}^\pm = \mathbf{v}_t^\pm e^{\mp \eta \nabla L(\mathbf{w}_t)} = \mathbf{v}_t^\pm (1 \mp \eta \nabla L(\mathbf{w}_t)) + O(\eta^2).$$

Dropping the  $O(\eta^2)$  term and using (22) gives

$$\mathbf{v}_t^+ + \mathbf{v}_t^- = \sqrt{\mathbf{w}_t^2 + 4\beta^2},$$

so that the update becomes (4):

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sqrt{\mathbf{w}_t^2 + 4\beta^2} \nabla L(\mathbf{w}_t).$$

**Proof of Theorem 4.** Using (17),  $\nabla L(\mathbf{w}_t) = 2(\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta})$ , the update (4) becomes:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - 2\eta \sqrt{\mathbf{w}_t^2 + 4\beta^2} (\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta}) \quad (23)$$

We set  $\beta = 1/(2d)$ ,  $\eta = 1/4$ , and the number of steps of the algorithm  $T = 4\sqrt{d}$ . Note that  $d \geq 4$ . Using the deviation bound for zero-mean Gaussian  $z \sim N(0, \tau^2)$ ,  $P(z \geq -\gamma) \leq \exp\left\{-\frac{\gamma^2}{2\tau^2}\right\}$ , we get that  $P(|\zeta_i| \geq \gamma) \leq 2\exp\left\{-\frac{md\gamma^2}{8\sigma^2}\right\}$ . Taking the union bound we have  $P(\exists i \ |\zeta_i| \geq \gamma) \leq 2d\exp\left\{-\frac{md\gamma^2}{8\sigma^2}\right\}$ . Denoting the probability on the left-hand side by  $\delta$ , we can solve for  $\gamma$ :

$$\gamma = \sigma \sqrt{\frac{\ln \frac{2d}{\delta}}{md}}.$$

This means the with probability at least  $1 - \delta$ ,  $|\zeta_i| \leq \gamma$  for all  $i = 1, \dots, d$ . Let us call this event  $E$ , and we condition everything what follows on the fact that  $E$  happened. Furthermore, due to our assumption that  $m$  grows at least logarithmically with  $d$ ,  $m \geq 8\sigma^2 \ln \frac{2d}{\delta}$ , we have  $\gamma \leq \frac{1}{\sqrt{8d}}$ .

We rewrite the update (23) in terms of  $s_t = w_t - e_1 - \zeta$

$$s_{t+1} = (1 - 2\eta\sqrt{(s_t + e_1 + \zeta)^2 + 4\beta^2})s_t \quad (24)$$

**The analysis for ‘zero signal’ direction.** Since every weights evolves independently of the other weights, we can analyze each coordinate separately. We start with any coordinate  $i \geq 2$  (‘zero signal’ weights), for which the update becomes

$$s_{t+1,i} = (1 - 2\eta\sqrt{(s_{t,i} + \zeta_i)^2 + 4\beta^2})s_{t,i},$$

with  $s_{t,1} = -\zeta_1$ . Now, w.l.o.g. assume  $\zeta_i < 0$  (the analysis for  $\zeta_i > 0$  is analogous). This means that  $s_{1,i} > 0$ , and  $s_{t,i}$  is positive and monotonically decreasing as long as  $2\eta\sqrt{(s_{t,i} + \zeta_i)^2 + 4\beta^2} < 1$ ; this condition is ensured by noticing that

$$(s_{t,i} + \zeta_i)^2 + 4\beta^2 \leq \zeta_i^2 + d^{-2} \leq \gamma^2 + \frac{1}{4} \leq \frac{1}{2},$$

so that using learning rate  $\eta < \sqrt{2}/2$  will do the trick (remind that we use  $\eta = 1/4$ ). Since we know that  $s_{t,i}$  is positive and monotonically decreasing, we can bound:

$$s_{t+1,i} \geq (1 - 2\eta\sqrt{(s_{T+1,i} + \zeta_i)^2 + 4\beta^2})s_{t,i},$$

so that

$$s_{T+1,i} \geq (1 - 2\eta\sqrt{(s_{T+1,i} + \zeta_i)^2 + 4\beta^2})^T s_{1,i} \geq (1 - 2\eta T\sqrt{(s_{T+1,i} + \zeta_i)^2 + 4\beta^2})\zeta_i,$$

where we used the Bernoulli inequality  $(1 + x)^n \geq 1 + xn$ . Returning to the original variable  $w_{t,i} = s_{t,i} + \zeta_i$  gives

$$w_{T+1,i} \geq 2\eta T\zeta_i\sqrt{w_{T+1,i}^2 + 4\beta^2}.$$

Since we also know that  $w_{T+1,i} < 0$  (because  $s_{t,i} = w_{t,i} - \zeta_i$  was decreasing in  $t$  with  $s_{1,i} = -\zeta_i$  and  $w_{1,i} = 0$ ), we have

$$w_{T+1,i}^2 \leq 4\eta^2 T^2 \zeta_i^2 (w_{T+1,i}^2 + 4\beta^2),$$

which can be solved for  $w_{T+1,i}^2$ :

$$w_{T+1,i}^2 \leq \frac{16\beta^2\eta^2 T^2 \zeta_i^2}{1 - 4\eta^2 T^2 \zeta_i^2} = \frac{1}{d^2} \frac{4\eta^2 T^2 \gamma^2}{1 - 4\eta^2 T^2 \gamma^2}$$

This expression is increasing in  $\zeta_i^2$  so we can upper-bound it by

$$w_{T+1,i}^2 \leq \frac{1}{d^2} \frac{4\eta^2 T^2 \gamma^2}{1 - 4\eta^2 T^2 \gamma^2}$$

Since  $\eta = 1/4$ ,  $T = 4\sqrt{d}$  and  $\gamma \leq \frac{1}{\sqrt{8d}}$ , the denominator is bounded from below

$$1 - 4\eta^2 T^2 \gamma^2 \geq 1 - \frac{1}{2} \geq \frac{1}{2},$$

so that, using  $\gamma = \sigma \sqrt{\frac{\ln \frac{2d}{\delta}}{md}}$ ,

$$w_{T+1,i}^2 \leq \frac{1}{d^2} \frac{8d \cdot \sigma^2 \ln \frac{2d}{\delta}}{md} = \frac{8\sigma^2 \ln \frac{2d}{\delta}}{md^2}.$$

Thus, the total error from ‘zero-signal’ coordinates is

$$\sum_{i=2}^d w_{T+1,i}^2 \leq \frac{8\sigma^2 \ln \frac{2d}{\delta}}{md} \quad (25)$$

We also need to show that the same amount of error comes from the first coordinate.

**Analysis for the ‘signal’ coordinate.** Using (24), we have for  $s_{t,1} = 1 - \zeta_i - w_{t,1}$ :

$$s_{t+1,1} = (1 - 2\eta \sqrt{(1 - s_{t,1} - \zeta_i)^2 + 4\beta^2}) s_{t,1},$$

with  $s_{1,1} = 1 - \zeta_i$ . As before, we note that  $s_{t,i}$  is decreasing in  $t$ , as long as  $2\eta \sqrt{(1 - s_{t,1} - \zeta_i)^2 + 4\beta^2} < 1$ . However, this condition is satisfied for our choice of  $\eta = 1/4$ , because

$$\begin{aligned} 2\eta \sqrt{(1 - s_{t,1} - \zeta_i)^2 + 4\beta^2} &\leq 2\eta \sqrt{(1 - \zeta_i)^2 + 4\beta^2} \leq \frac{1}{2} \sqrt{1 + 2|\zeta_i| + \zeta_i^2 + d^{-2}} \\ &\leq \frac{1}{2} \sqrt{1 + \frac{1}{\sqrt{2d}} + \frac{9}{8d^2}} \stackrel{d \geq 4}{\leq} \frac{1}{2} \sqrt{1 + \frac{1}{8} + \frac{9}{128}} < 1. \end{aligned}$$

Now, we need to carefully analyze the update. Initially  $s_{t,1}$  decreases slowly, as the square root term is essentially of order  $1/d$ . At some point, however,  $s_{t,1}$  falls below a certain constant (say,  $s_{t,1} = 1/2$ ), and the square root term is of order  $O(1)$ , and the convergence becomes exponential.

First, to simplify analysis we simply denote  $s_{t,i}$  by  $s_t$ ; moreover, define  $r = 1 - \zeta_i$ , so that the update becomes

$$s_{t+1} = (1 - 2\eta \sqrt{(r - s_t)^2 + 4\beta^2}) s_t, \quad s_1 = r, \quad (26)$$

Already after the first iteration,

$$s_2 = (1 - 1/2d^{-1}) s_1 = \frac{2d-1}{2d} r,$$

so that  $(r - s_2)^2 = 1/(4d^2)$  becomes comparable with  $4\beta^2 = 1/d^2$  term. So we can drop the  $4\beta^2$  term from the square root in (26) and upper bound

$$s_{t+1} \leq (1 - 2\eta(r - s_t)) s_t \quad (27)$$

To get some insight into this expression we solve the corresponding differential equation:

$$\dot{s} = -2\eta(r - s)s,$$

which give:

$$\frac{s_t}{r - s_t} = C e^{-\eta r t} \implies s_t = \frac{r}{1 + C e^{2\eta r t}}.$$



Inspired by this we will bound  $\frac{s_t}{r-s_t}$ . From (27) we get:

$$r - s_{t+1} \geq r - (1 - 2\eta(r - s_t)s_t) = (1 + 2\eta s_t)(r - s_t),$$

so that:

$$\frac{s_{t+1}}{r - s_{t+1}} \leq \underbrace{\left( \frac{1 - 2\eta(r - s_t)}{1 + 2\eta s_t} \right)}_{=: A_t} \left( \frac{s_t}{r - s_t} \right).$$

We will now bound  $A_t$  independent of  $s_t$ . To this end, note that  $A_t$  is maximized when  $s_t = r$ . Indeed,

$$A_t = \frac{1 + 2\eta s_t - 2\eta r}{1 + 2\eta s_t} = 1 - \frac{2\eta r}{1 + 2\eta s_t} \leq 1 - \frac{2\eta r}{1 + 2\eta r} = \frac{1}{1 + 2\eta r}.$$

This way, we get an upper bound:

$$\frac{s_{T+1}}{r - s_{T+1}} \leq (1 + 2\eta r)^{-(T-1)} \frac{s_2}{r - s_2} = (2d - 1)(1 + 2\eta r)^{-(T-1)},$$

or by solving for  $s_{T+1}$ ,

$$s_{T+1} \leq \frac{r}{1 + (2d - 1)^{-1}(1 + 2\eta r)^{T-1}} \leq r(2d - 1)(1 + 2\eta r)^{-(T-1)}.$$

The expression above is decreasing in  $r$  for  $T \geq 4$  (can be verified by computing the derivative), so we will upper-bound it by lower-bounding  $r$ , that is  $r = 1 - \zeta_i \geq 1 - \frac{1}{\sqrt{8d}}$ . Taking  $\eta = 1/4$  and using  $1 - \frac{1}{\sqrt{8d}} \stackrel{d \geq 4}{\geq} 1 - \frac{1}{4\sqrt{2}}$ , we get  $1 + 2\eta r \geq \frac{3}{2} - \frac{1}{8\sqrt{2}} > e^{1/3}$  (checked numerically). Therefore,

$$s_{T+1} \leq (2d - 1)e^{-(T-1)/3} \leq e^{1/3 + \ln 2} e^{-T/3 + \ln d} \leq 3e^{-T/3 + \ln d}.$$

Using the fact that  $T = 4\sqrt{d}$ , we get

$$s_{T+1} \leq 3e^{-\frac{4}{3}\sqrt{d} + \ln d}$$

To bound  $(1 - w_{T+1,1})^2$  we use

$$(1 - w_{T+1,1})^2 = (s_{T+1,1} + \zeta_1)^2 \leq 2s_{T+1,1}^2 + 2\zeta_1^2 \leq 9e^{-\frac{8}{3}\sqrt{d} + 2\ln d} + \frac{2\sigma^2 \ln \frac{2d}{\delta}}{md}. \quad (28)$$

**Bound the error of Approximated EGU<sup>±</sup> algorithm** The final error of the algorithm is obtained by summing (25) and (28):

$$\begin{aligned} \|\mathbf{w}_{T+1} - \mathbf{e}_1\|^2 &\leq 9e^{-\frac{8}{3}\sqrt{d} + 2\ln d} + \frac{2\sigma^2 \ln \frac{2d}{\delta}}{md} + \frac{8\sigma^2 \ln \frac{2d}{\delta}}{md} \\ &= \frac{10\sigma^2 \ln \frac{2d}{\delta}}{md} + 9e^{-\frac{8}{3}\sqrt{d} + 2\ln d}. \end{aligned}$$

■

## Appendix F. Upper bound for the spindly network

In this section, we upper-bound the error of the spindly network defined in Figure 1, showing essentially the same bound (up to constants) as for the Approximated EGU $^\pm$ . We note that essentially the same algorithm has been analyzed by Vaskevicius et al. (2019), giving the same bound  $O(\frac{\sigma^2 \log d}{md})$  under the restricted isometry property (RIP) assumption.

**Theorem 9** Assume  $d \geq 4$  is such that  $\sqrt{d}$  is an integer. Consider the spindly network given in Figure 1 trained with gradient descent, with weights initialized as  $\mathbf{u} = \sqrt{2/d}\mathbf{1}$  and  $\mathbf{v} = \mathbf{0}$ , and the learning rate set to  $\eta = 1/4$ . Let  $m \geq 8\sigma^2 \ln \frac{2d}{\delta} = \Omega(\sigma^2 \log(d/\delta))$ . With probability at least  $1 - \delta$ , the algorithm run for  $T = 4\sqrt{d}$  steps achieves error bounded by:

$$e(\mathbf{w}_{T+1}) \leq \frac{33\sigma^2 \ln \frac{2d}{\delta}}{4md} + 16e^{-\frac{8}{3}\sqrt{d}+2\ln d} = O\left(\frac{\sigma^2 \log d}{md} + e^{-\frac{8}{3}\sqrt{d}}\right)$$

**Proof** The spindly network predicts with  $\mathbf{w}_t$  given by  $\mathbf{w}_t = \mathbf{u}_t \odot \mathbf{v}_t$ , and both  $\mathbf{u}_t$  and  $\mathbf{v}_t$  are updated with the gradient descent algorithm:

$$\mathbf{u}_{t+1} = \mathbf{u}_t - \eta \nabla_{\mathbf{u}_t} L(\mathbf{w}_t), \quad \mathbf{v}_{t+1} = \mathbf{v}_t - \eta \nabla_{\mathbf{v}_t} L(\mathbf{w}_t)$$

Using (17) and the chain rule,  $\nabla_{\mathbf{u}_t} L(\mathbf{w}_t) = 2(\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta}) \odot \mathbf{v}_t$  and  $\nabla_{\mathbf{v}_t} L(\mathbf{w}_t) = 2(\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta}) \odot \mathbf{u}_t$ , so that

$$\mathbf{u}_{t+1} = \mathbf{u}_t - 2\eta(\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta}) \odot \mathbf{v}_t, \quad \mathbf{v}_{t+1} = \mathbf{v}_t - 2\eta(\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta}) \odot \mathbf{u}_t \quad (29)$$

Let us introduce two vectors,  $\mathbf{v}_t^+$  and  $\mathbf{v}_t^-$ , given by:

$$\mathbf{v}_t^+ = \frac{1}{4}(\mathbf{u}_t + \mathbf{v}_t)^2, \quad \mathbf{v}_t^- = \frac{1}{4}(\mathbf{u}_t - \mathbf{v}_t)^2,$$

(square applied coordinatewise) and note that

$$\mathbf{w}_t = \mathbf{u}_t \cdot \mathbf{v}_t = \mathbf{v}_t^+ - \mathbf{v}_t^-.$$

Subtracting and adding equations in (29), followed by squaring both sides gives

$$\begin{aligned} \mathbf{v}_{t+1}^+ &= (1 - 2\eta(\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta}))^2 \odot \mathbf{v}_t^+ \\ \mathbf{v}_{t+1}^- &= (1 + 2\eta(\mathbf{w}_t - \mathbf{e}_1 - \boldsymbol{\zeta}))^2 \odot \mathbf{v}_t^- \end{aligned} \quad (30)$$

We initialize the algorithm as follows:

$$\mathbf{u}_1 = \sqrt{\frac{2}{d}}, \quad \mathbf{v}_1 = \mathbf{0},$$

so that  $\mathbf{v}_1^+ = \mathbf{v}_1^- = \frac{1}{2d}\mathbf{1}$  and  $\mathbf{w}_1 = \mathbf{0}$ , similarly as in the Approximated EGU $^\pm$ . We set  $\eta = \frac{1}{8}$ ,  $T = 4\sqrt{d}$ , and use the same assumption as in the previous section, that is  $m \geq 8\sigma^2 \ln \frac{2d}{\delta}$ , which implies that with probability at least  $1 - \delta$ ,  $|\zeta_i| \leq \gamma$  for all  $i = 1, \dots, d$ , where

$$\gamma = \sigma \sqrt{\frac{\ln \frac{2d}{\delta}}{md}} \leq \frac{1}{\sqrt{8d}}. \quad (31)$$

As before, we denote the high probability event above as  $E$ , and we condition everything what follows on the fact that  $E$  happened. We will also assume that  $d \geq 9$ .

**The analysis for ‘zero signal’ direction.** As before, we can analyze each coordinate separately. We start with any coordinate  $i \geq 2$ , for which the update (30) becomes

$$v_{t+1,i}^\pm = (1 \mp 2\eta(w_{t,i} - \zeta_i))^2 v_{t,i}^\pm \quad (32)$$

W.l.o.g. assume  $\zeta_i > 0$  (the analysis for  $\zeta_i < 0$  is analogous). We will first prove by induction on  $t$  that  $0 \leq w_{t,i} \leq \zeta_i$  for all  $t = 1, \dots, T+1$ . Since  $w_{1,i} = 0$  and  $\zeta_i > 0$ , it clearly holds for  $t = 1$ . Now, assume that it holds for iterations  $1, \dots, t$  and we prove that it also holds for  $t+1$ . We have:

$$\begin{aligned} w_{t+1,i} &= v_{t+1,i}^+ - v_{t+1,i}^- \\ &= (1 - 2\eta(w_{t,i} - \zeta_i))^2 v_{t,i}^+ - (1 + 2\eta(w_{t,i} - \zeta_i))^2 v_{t,i}^- \\ &= (1 + 4\eta^2(w_{t,i} - \zeta_i)^2) w_{t,i} - 4\eta(w_{t,i} - \zeta_i)(v_{t,i}^+ + v_{t,i}^-) \geq 0, \end{aligned} \quad (33)$$

because  $w_{t,i} - \zeta_i < 0$  and  $w_{t,i} \geq 0$  from the induction assumption, while  $v_{t,i}^+, v_{t,i}^- \geq 0$  from their definitions.

Now, since  $-\zeta_i \leq w_{q,i} - \zeta_i \leq 0$  for  $q = 1, \dots, t$  from the inductive assumption,  $\eta = \frac{1}{8}$  and  $|\zeta_i| \leq \gamma \leq \frac{1}{\sqrt{8d}}$  from (31), we have  $0 \geq 2\eta(w_{q,i} - \zeta_i) \geq -\frac{1}{4\sqrt{8d}}$ , so that

$$(1 + 2\eta(w_{q,i} - \zeta_i))^2 < 1, \quad (1 - 2\eta(w_{q,i} - \zeta_i))^2 > 1, \quad q = 1, \dots, t.$$

Thus, we see from (32) that  $v_{q,i}^+$  is monotonically increasing in  $q$ , and  $v_{q,i}^-$  is monotonically decreasing in  $q$ . This means that  $w_{q,i} = v_{q,i}^+ - v_{q,i}^-$  is monotonically increasing in  $q$ . Therefore,

$$(1 + 2\eta(w_{q,i} - \zeta_i))^2 \geq (1 + 2\eta(w_{1,i} - \zeta_i))^2 = (1 - 2\eta\zeta_i)^2 \quad \text{for all } q = 1, \dots, t,$$

and, similarly,

$$(1 - 2\eta(w_{q,i} - \zeta_i))^2 \leq (1 + 2\eta(w_{1,i} - \zeta_i))^2 = (1 + 2\eta\zeta_i)^2 \quad \text{for all } q = 1, \dots, t.$$

This let us upper-bound  $v_{t+1,i}^+$  as

$$v_{t+1,i}^+ = \prod_{q=1}^t (1 - 2\eta(w_{q,i} - \zeta_i))^2 v_{1,i}^+ \leq (1 + 2\eta\zeta_i)^{2t} v_{1,i}^+ = (1 + 2\eta\zeta_i)^{2t} \frac{1}{2d},$$

and similarly lower-bound  $v_{t+1,i}^-$  as

$$v_{t+1,i}^- = \prod_{q=1}^t (1 + 2\eta(w_{q,i} - \zeta_i))^2 v_{1,i}^- \geq (1 - 2\eta\zeta_i)^{2t} v_{1,i}^- = (1 - 2\eta\zeta_i)^{2t} \frac{1}{2d}.$$

We have

$$(1 + 2\eta\zeta_i)^{2t} = e^{2t \ln(1 + 2\eta\zeta_i)} \leq e^{4t\eta\zeta_i},$$

where we used  $\ln(1 + x) \leq x$ . Moreover, by Bernoulli’s inequality,

$$(1 - 2\eta\zeta_i)^{2t} \geq 1 - 4t\eta\zeta_i.$$

This gives:

$$w_{t+1,i} = v_{t+1,i}^+ - v_{t+1,i}^- = \frac{1}{2d} ((1 + 2\eta\zeta_i)^{2t} - (1 - 2\eta\zeta_i)^{2t}) \leq \frac{1}{2d} (e^{4t\eta\zeta_i} - 1 + 4t\eta\zeta_i).$$

Now, note that

$$4t\eta\zeta_i \leq \frac{T}{2}\gamma \leq \frac{4\sqrt{d}}{2\sqrt{8d}} \leq \frac{1}{\sqrt{2}},$$

Using the convexity of  $e^x$ , we have for  $x \in [0, a]$  that

$$e^x = e^{(1-\frac{x}{a}) \cdot 0 + \frac{x}{a} \cdot a} \leq \left(1 - \frac{x}{a}\right) e^0 + \frac{x}{a} e^a = 1 + x \frac{e^a - 1}{a}.$$

Taking  $x = 4t\eta\zeta_i$  and  $a = \frac{1}{\sqrt{2}}$ , we have

$$e^{4t\eta\zeta_i} \leq 1 + 4t\eta\zeta_i \sqrt{2} (e^{1/\sqrt{2}} - 1) \leq 1 + 6t\eta\zeta_i.$$

This allows us to bound

$$w_{t+1,i} \leq \frac{1}{2d} (e^{4t\eta\zeta_i} - 1 + 4t\eta\zeta_i) \leq \frac{1}{2d} 10t\eta\zeta_i \leq \frac{5T\eta\zeta_i}{d} = \frac{5}{2\sqrt{d}} \zeta_i \leq \frac{5}{6} \zeta_i \leq \zeta_i, \quad (34)$$

where we used  $d \geq 9$ . This finishes the inductive proof that  $0 \leq w_{t,i} \leq \zeta_i$  for all  $t = 1, \dots, T+1$ . However, applying (34) to  $t = T$  gives

$$w_{T+1,i} \leq \frac{5}{2\sqrt{d}} \zeta_i,$$

so that using (31)

$$w_{T+1,i}^2 \leq \frac{25}{4d} \zeta_i^2 \leq \frac{25}{4d} \gamma^2 = \frac{25\sigma^2 \ln \frac{2d}{\delta}}{4md^2}.$$

Thus, the total error from ‘zero-signal’ coordinates is

$$\sum_{i=2}^d w_{T+1,i}^2 \leq \frac{25\sigma^2 \ln \frac{2d}{\delta}}{4md}. \quad (35)$$

**Analysis for the ‘signal’ coordinate.** For  $i = 1$ , the update becomes:

$$v_{t+1,1}^\pm = (1 \mp 2\eta(w_{t,1} - 1 - \zeta_1))^2 v_{t,1}^\pm.$$

First, we will show by induction that  $0 \leq w_{t,1} \leq 1 + \zeta_1$  for all  $t = 1, \dots, T+1$ . This is clearly true for  $t = 1$  as  $w_{t,1} = 0$  and  $|\zeta_1| \leq \frac{1}{\sqrt{8d}}$ . Assume now this is true for  $q = 1, \dots, t$ , and we prove it is also true for  $t+1$ . We have

$$\begin{aligned} w_{t+1,1} &= v_{t+1,1}^+ - v_{t+1,1}^- \\ &= (1 - 2\eta(w_{t,1} - 1 - \zeta_1))^2 v_{t,1}^+ - (1 + 2\eta(w_{t,1} - 1 - \zeta_1))^2 v_{t,1}^- \\ &= (1 + 4\eta^2(w_{t,1} - 1 - \zeta_1)^2) w_{t,1} - 4\eta(w_{t,1} - 1 - \zeta_1)(v_{t,1}^+ + v_{t,1}^-) \\ &= (1 + 4\eta^2(w_{t,1} - 1 - \zeta_1)^2) w_{t,1} - 4\eta(w_{t,1} - 1 - \zeta_1)(w_{t,1} + 2v_{t,1}^-) \\ &= (1 - 2\eta(w_{t,1} - 1 - \zeta_1))^2 w_{t,1} + 8\eta(1 + \zeta_1 - w_{t,1}) v_{t,1}^-. \end{aligned} \quad (36)$$

It follows from the induction assumption that both terms in the last line are nonnegative, thus  $w_{t+1,1} \geq 0$ . To show that  $w_{t+1,1} \leq 1 + \zeta_1$ , note that by inductive assumption  $1 + 2\eta(w_{t,i} - 1 - \zeta_1) \geq 1 - 2\eta(1 - \zeta_1) \geq 1 - 2\eta(1 + \gamma) = \frac{3}{4} - \frac{1}{4}\gamma \geq \frac{3}{4} - \frac{1}{4\sqrt{8d}} > 0$ , and also  $1 + 2\eta(w_{t,i} - 1 - \zeta_1) \leq 1$ . This means that  $v_{q,1}^-$  is nonincreasing in  $q$ , and thus  $v_{t,1}^- \leq v_{1,1}^- = \frac{1}{2d} \leq \frac{1}{18}$  (due to  $d \geq 9$ ). Plugging this into (36), we can bound

$$\begin{aligned} w_{t+1,1} &\leq (1 - 2\eta(w_{t,1} - 1 - \zeta_1))^2 w_{t,1} + 8\eta(1 + \zeta_1 - w_{t,1}) \frac{1}{18} \\ &= (1 - 2\eta(w_{t,1} - 1 - \zeta_1))^2 w_{t,1} + \frac{4}{9}\eta(1 + \zeta_1 - w_{t,1}). \end{aligned} \quad (37)$$

We now maximize the right-hand side of (37) with respect  $w_{t,1} \in [0, 1 + \zeta_1]$ . To this end, define function

$$f(x) = (1 + (a - x)/4)^2 x + (a - x)/18,$$

with  $a = 1 + \zeta_1$ . We get

$$f'(x) = -\frac{x}{2}(1 + (a - x)/4) + (1 + (a - x)/4)^2 - \frac{1}{18},$$

which is a convex quadratic function of  $x$ , so that  $f(x)$  achieves its maximum in the left root of  $f'(x)$ . Solving  $f'(x) = 0$  is equivalent to

$$\frac{3x^2}{16} - x \left(1 + \frac{a}{4}\right) + \frac{17}{18} + \frac{a}{2} + \frac{a^2}{16} = 0,$$

which left root is given by

$$x_\ell = \frac{1}{3} \left( 2a + 8 - \sqrt{a^2 + 8a + \frac{51}{3}} \right).$$

Note that  $a = 1 + \zeta_1 \geq 1 - \gamma > 1 - \frac{1}{\sqrt{8d}} \geq 1 - \frac{1}{3\sqrt{8}} \geq 0.88 := a_0$ . One can verify that  $x_\ell$  is increasing in  $a$  (e.g. by inspecting the sign of the derivative of  $x_\ell$  with respect to  $a$ ), which means that

$$x_\ell \geq \frac{1}{3} \left( 2a_0 + 8 - \sqrt{a_0^2 + 8a_0 + \frac{51}{3}} \right) \geq 1.59.$$

This value is to the right of range  $[0, a]$ , because  $a \leq 1 + \gamma > 1 + \frac{1}{3\sqrt{8}} \leq 1.12$ . This means that the maximum of  $f(x)$  in the range  $[0, a]$  is achieved for  $x = 0$ . This corresponds to  $w_{t,1} = 1 + \zeta_i$  in (37), which gives:

$$w_{t+1,1} \leq 1 + \zeta_1,$$

which was to be shown by induction.

We now lower bound  $w_{t+1,1}$ . Using (36) and the proven fact that  $1 + \zeta_i - w_{t,1} \geq 0$  for all  $t$ , we have

$$w_{t+1,1} \geq (1 - 2\eta(w_{t,1} - 1 - \zeta_1))^2 w_{t,1} = (1 + 2\eta(r - w_{t,1}))^2 w_{t,1},$$

where we simplified the notation with  $r = 1 + \zeta_1$ . We further bound

$$(1 + 2\eta(r - w_{t,1}))^2 = (1 + 4\eta(r - w_{t,1}) + 4\eta^2(r - w_{t,1})^2) \geq 1 + 4\eta(r - w_{t,1}),$$

so that

$$w_{t+1,1} \geq (1 + 4\eta(r - w_{t,1}))w_{t,1}.$$

Now consider expression  $Q_{t+1} = \frac{r}{w_{t+1,1}} - 1$ . Clearly,  $Q_{t+1}$  is decreasing  $w_{t+1,1}$  so we have

$$\begin{aligned} Q_{t+1} &\leq \frac{r}{(1 + 4\eta(r - w_{t,1}))w_{t,1}} - 1 = \frac{Q_t + 1}{1 + 4\eta(r - w_{t,1})} - 1 \\ &= \frac{Q_t - 4\eta(r - w_{t,1})}{1 + 4\eta(r - w_{t,1})} = \frac{Q_t - 4\eta w_{t,1} Q_t}{1 + 4\eta(r - w_{t,1})} = Q_t \frac{1 - 4\eta w_{t,1}}{1 + 4\eta(r - w_{t,1})}. \end{aligned}$$

Now, note that

$$\frac{1 - 4\eta w_{t,1}}{1 + 4\eta(r - w_{t,1})} = 1 - \frac{4\eta r}{1 + 4\eta(r - w_{t,1})} \leq 1 - \frac{4\eta r}{1 + 4\eta r} = \frac{1}{1 + 4\eta r},$$

where we used  $w_{t,1} \geq 0$ . Thus we get

$$Q_{t+1} \leq \frac{1}{1 + 4\eta r} Q_t$$

for all  $t = 1, \dots, T$ , which implies

$$Q_{T+1} \leq \frac{1}{(1 + 4\eta r)^{T-1}} Q_2$$

(we cannot start from  $Q_1$  as it is undefined). To obtain  $Q_2$ , we note that

$$w_{2,1} = (1 + 2\eta r)^2 v_1^+ - (1 - 2\eta r)^2 v_1^- = \frac{1}{2d} ((1 + 2\eta r)^2 - (1 - 2\eta r)^2) = \frac{4\eta r}{d}.$$

Therefore,

$$Q_2 = \frac{d}{4\eta} - 1 = 2d - 1 \leq 2d,$$

and so

$$Q_{T+1} = \frac{r - w_{T+1,1}}{w_{T+1,1}} \leq \frac{2d}{(1 + r/2)^{T-1}},$$

or, equivalently,

$$r - w_{T+1,1} \leq r \frac{\frac{2d}{(1+r/2)^{T-1}}}{1 + \frac{2d}{(1+r/2)^{T-1}}} \leq r \frac{2d}{(1 + r/2)^{T-1}}.$$

We can bound  $r = 1 + \zeta_1 \geq 1 - \gamma \geq 1 - \frac{1}{\sqrt{8d}} \geq 1 - \frac{1}{3\sqrt{8}} \geq 0.88$  and similarly  $r \leq 1 + \frac{1}{3\sqrt{8}} \leq 1.12$ .

This gives  $1 + r/2 \geq 1.44 \geq e^{1/3}$ , so that

$$r - w_{T+1,1} \leq 2.24de^{-1/3(T-1)} = 2.24e^{1/3}e^{-4/3\sqrt{d} + \ln d} \leq 4e^{-4/3\sqrt{d} + \ln d}$$

To bound  $(1 - w_{T+1,1})^2$  we use

$$(1 - w_{T+1,1})^2 = (r - w_{T+1,1} - \zeta_1)^2 \leq 2(r - w_{T+1,1})^2 + 2\zeta_1^2 \leq 16e^{-8/3\sqrt{d} + 2\ln d} + \frac{2\sigma^2 \ln \frac{2d}{\delta}}{md}, \quad (38)$$

where in the last line we used (31).



**Bound the error of the Spindly network.** The final error of the algorithm is obtained by summing (35) and (38):

$$\begin{aligned}\|\mathbf{w}_{T+1} - \mathbf{e}_1\|^2 &\leq 16e^{-8/3\sqrt{d}+2\ln d} + \frac{2\sigma^2 \ln \frac{2d}{\delta}}{md} + \frac{25\sigma^2 \ln \frac{2d}{\delta}}{4md} \\ &= \frac{33\sigma^2 \ln \frac{2d}{\delta}}{4md} + 16e^{-\frac{8}{3}\sqrt{d}+2\ln d}.\end{aligned}$$

■

## Appendix G. Upper bound for the priming method

The priming method operates as follows (Warmuth and Amid, 2023): First, it computes the least squares estimator  $\hat{\mathbf{w}}_{LS} = (\mathbf{X}^\top \mathbf{X})\mathbf{X}^\top \mathbf{y}$ . Then, it scales each column of  $\mathbf{X}$  (each feature) by the corresponding coordinate of  $\hat{\mathbf{w}}_{LS}$ , resulting in a new matrix  $\tilde{\mathbf{X}} = \mathbf{X} \text{diag}(\hat{\mathbf{w}}_{LS})$ . Next, it calculates the Ridge Regression solution  $\tilde{\mathbf{w}}_{RR}$  using the new inputs  $\tilde{\mathbf{X}}$  and an appropriate regularization constant  $\lambda$ . The final priming predictor  $\hat{\mathbf{w}}'$  is obtained by multiplying it by the coordinates of  $\hat{\mathbf{w}}_{LS}$ , that is  $\hat{\mathbf{w}}' = \text{diag}(\hat{\mathbf{w}}_{LS})\tilde{\mathbf{w}}_{RR}$ . In the proof we rewrite the priming predictor  $\hat{\mathbf{w}}'$  as a regularized least-squares solution, with the square regularizer based on a matrix  $\lambda \text{diag}(\hat{\mathbf{w}}_{LS})^{-2}$ . Note that the regularization strength is amplified along directions where the coordinates of  $\hat{\mathbf{w}}_{LS}$  are small in magnitude, effectively biasing the algorithm towards sparse solutions. The proof carefully bounding the expected error of such a predictor.

**Theorem 10** *Consider the priming method equipped with  $\lambda = \sigma^2 \sqrt{d}$ . The expected error can be bounded by:*

$$e(\hat{\mathbf{w}}') \leq \frac{17\sigma^2}{md} + \frac{32\sigma^4}{m^2d} + \frac{4\sigma e^{-md/(8\sigma^2)}}{\sqrt{2\pi md}}.$$

Note that this upper bound is by a factor of  $O(\log d)$  better (assuming  $\sigma^2 = O(1)$ ) than the upper bound for Approximated EGU $^\pm$  (and thus by a factor of  $O(d)$  better than the error of any rotation invariant algorithm). However we don't know whether such an improved upper bound is also possible for Approximated EGU $^\pm$ .

**Proof** We start with rewriting the priming method into a form which is easier to analyze. Let  $\hat{\mathbf{w}}_{LS} = (\mathbf{X}^\top \mathbf{X})\mathbf{X}^\top \mathbf{y}$  be the least-squares solution, which induces a diagonal weight matrix  $\mathbf{W} = \text{diag}(\hat{\mathbf{w}}_{LS})$ . The new (rescaled) input matrix is then given by  $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{W}$ . The ridge regression solution on the new inputs with regularization constant  $\lambda$  is then  $\tilde{\mathbf{w}} = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} + \lambda \mathbf{I}_d)^{-1} \tilde{\mathbf{X}}^\top \mathbf{y}$ . Finally the output of the algorithm is  $\hat{\mathbf{w}}' = \mathbf{W}\tilde{\mathbf{w}}$ . We thus have

$$\begin{aligned}\hat{\mathbf{w}}' &= \mathbf{W}(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} + \lambda \mathbf{I}_d)^{-1} \tilde{\mathbf{X}}^\top \mathbf{y} \\ &= \mathbf{W}(\mathbf{W}\mathbf{X}^\top \mathbf{X}\mathbf{W} + \lambda \mathbf{I}_d)^{-1} \mathbf{W}\mathbf{X}\mathbf{y} \\ &= \left( \mathbf{W}^{-1}(\mathbf{W}\mathbf{X}^\top \mathbf{X}\mathbf{W} + \lambda \mathbf{I}_d)\mathbf{W}^{-1} \right)^{-1} \mathbf{X}\mathbf{y} \\ &= \left( \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{W}^{-2} \right)^{-1} \mathbf{X}\mathbf{y}\end{aligned}$$

(in any of the elements of  $\widehat{\mathbf{w}}_{LS}$  is zero, take the limit of the expression above). Thus,  $\widehat{\mathbf{w}}'$  is a regularized least-square solution with the quadratic regularization matrix  $\lambda \mathbf{W}^{-2}$ :

$$\widehat{\mathbf{w}}' = \underset{\widehat{\mathbf{w}}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}\|^2 + \frac{\lambda}{2} \widehat{\mathbf{w}}^\top \mathbf{W}^{-2} \widehat{\mathbf{w}} \right\}.$$

Note that the regularization strength is amplified along directions where the coordinates of  $\widehat{\mathbf{w}}_{LS}$  are small in magnitude, effectively biasing the algorithm towards sparse solutions.

We now specialize the priming predictor to our setup. We have

$$\mathbf{X}^\top \mathbf{X} = n \mathbf{I}_d, \quad \text{and} \quad \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{X} \mathbf{e}_1 + \mathbf{X}^\top \boldsymbol{\xi} = n \mathbf{e}_1 + \sqrt{n} \boldsymbol{\zeta},$$

where  $\boldsymbol{\zeta} = n^{-1/2} \mathbf{X}^\top \boldsymbol{\xi}$ . Being a linear function of  $\boldsymbol{\xi}$ ,  $\boldsymbol{\zeta}$  has a normal distribution with parameters that can be obtained from:

$$\mathbb{E}[\boldsymbol{\zeta}] = n^{-1/2} \mathbf{X}^\top \mathbb{E}[\boldsymbol{\xi}] = \mathbf{0}, \quad \mathbb{E}[\boldsymbol{\zeta} \boldsymbol{\zeta}^\top] = n^{-1} \mathbf{X}^\top \mathbb{E}[\boldsymbol{\xi} \boldsymbol{\xi}^\top] \mathbf{X} = n^{-1} \sigma^2 \mathbf{X}^\top \mathbf{X} = \sigma^2 \mathbf{I}_d,$$

that is  $\boldsymbol{\zeta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$ . This gives us the expression for  $\widehat{\mathbf{w}}_{LS}$ :

$$\widehat{\mathbf{w}}_{LS} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = n^{-1} (n \mathbf{e}_1 + \sqrt{n} \boldsymbol{\zeta}) = \mathbf{e}_1 + n^{-1/2} \boldsymbol{\zeta},$$

as well as for  $\widehat{\mathbf{w}}'$ :

$$\widehat{\mathbf{w}}' = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{W}^{-2})^{-1} \mathbf{X}^\top \mathbf{y} = (n \mathbf{I}_d + \lambda \mathbf{W}^{-2})^{-1} (n \mathbf{e}_1 + \sqrt{n} \boldsymbol{\zeta})$$

We will analyze  $\widehat{\mathbf{w}}'$  separately for each coordinate  $i = 1, \dots, d$ . To simplify the notation, let  $w_i$  be the  $i$ -th coordinate of  $\widehat{\mathbf{w}}'$ , and let  $v_i$  be the  $i$ -th coordinate of  $\widehat{\mathbf{w}}_{LS}$ . The error of  $\widehat{\mathbf{w}}'$  is given by:

$$e(\widehat{\mathbf{w}}') = \|\widehat{\mathbf{w}}' - \mathbf{e}_1\|^2 = (w_1 - 1)^2 + \sum_{i=2}^d w_i^2.$$

As  $\mathbf{W} = \operatorname{diag}(\widehat{\mathbf{w}}_{LS})$  is diagonal, we have for any  $i > 1$

$$w_i = \frac{\sqrt{n} \zeta_i}{n + \lambda v_i^{-2}} = \frac{\sqrt{n} v_i^2 \zeta_i}{n v_i^2 + \lambda} = \frac{1}{\sqrt{n}} \frac{\zeta_i^3}{\zeta_i^2 + \lambda},$$

where we used  $v_i = n^{-1/2} \zeta_i$  for  $i > 1$ . Thus the error on the  $i$ -th coordinate is

$$w_i^2 = \frac{1}{n} \frac{\zeta_i^6}{(\zeta_i^2 + \lambda)^2} \leq \frac{1}{n} \frac{\zeta_i^6}{\lambda^2}.$$

Taking expectation over  $\zeta_i \sim \mathcal{N}(0, \sigma^2)$  and using  $\mathbb{E}[\zeta_i^6] = 15\sigma^6$ ,

$$\mathbb{E}[w_i^2] \leq \frac{15\sigma^6}{n\lambda^2}.$$

We now switch to coordinate  $i = 1$ . We have  $v_1 = 1 + n^{-1/2} \zeta_1 = n^{-1} (n + \sqrt{n} \zeta_1)$  so that

$$w_1 = \frac{n + \sqrt{n} \zeta_1}{n + \lambda v_1^2} = \frac{v_1^2 (n + \sqrt{n} \zeta_1)}{v_1^2 n + \lambda} = \frac{n^{-2} (n + \sqrt{n} \zeta_1)^3}{n^{-1} (n + \sqrt{n} \zeta_1)^2 + \lambda} = \frac{(n + \sqrt{n} \zeta_1)^3}{n(n + \sqrt{n} \zeta_1)^2 + n^2 \lambda}.$$

The error on the first coordinate is thus

$$\begin{aligned}(w_1 - 1)^2 &= \left( \frac{(n + \sqrt{n}\zeta_1)^3 - n(n + \sqrt{n}\zeta_1)^2 - n^2\lambda}{n(n + \sqrt{n}\zeta_1)^2 + n^2\lambda} \right)^2 \\ &= \left( \frac{\sqrt{n}\zeta_1(n + \sqrt{n}\zeta_1)^2 - n^2\lambda}{n(n + \sqrt{n}\zeta_1)^2 + n^2\lambda} \right)^2.\end{aligned}$$

Using  $(a + b) \leq 2a^2 + 2b^2$  in the numerator and  $(a + b) \geq a^2 + b^2$  for  $a, b \geq 0$  in the denominator, we bound

$$\begin{aligned}(w_1 - 1)^2 &\leq 2 \frac{n\zeta_1^2(n + \sqrt{n}\zeta_1)^4 + n^4\lambda^2}{(n(n + \sqrt{n}\zeta_1)^2 + n^2\lambda)^2} \leq \frac{2n\zeta_1^2(n + \sqrt{n}\zeta_1)^4}{n^2(n + \sqrt{n}\zeta_1)^4} + \frac{2n^4\lambda^2}{n^2(n + \sqrt{n}\zeta_1)^4 + n^4\lambda^2} \\ &= \frac{2\zeta_1^2}{n} + \frac{2\lambda^2}{(\sqrt{n} + \zeta_1)^4 + \lambda^2}\end{aligned}$$

We now take expectation with respect  $\zeta_1$  and get:

$$\mathbb{E}[(w_1 - 1)^2] = \frac{2\sigma^2}{n} + \mathbb{E} \left[ 2\lambda^2((\sqrt{n} + \zeta_1)^4 + \lambda^2)^{-1} \right].$$

The second term requires more work. Using  $f(\zeta_1) = 2\lambda^2((\sqrt{n} + \zeta_1)^4 + \lambda^2)^{-1}$ :

$$\begin{aligned}\mathbb{E}[f(\zeta_1)] &= \mathbb{E}[f(\zeta_1)|\zeta_1 \leq c] P(\zeta_1 \leq c) + \mathbb{E}[f(\zeta_1)|\zeta_1 \geq c] P(\zeta_1 \geq c) \\ &\leq P(\zeta_1 \leq c) \max_{\zeta_1 \leq c} \{f(\zeta_1)\} + \max_{\zeta_1 \geq c} \{f(\zeta_1)\}.\end{aligned}$$

We take  $c = -\sqrt{n}/4$  and use a bound  $P(Z < -t) \leq \frac{\exp\{-t^2/2\}}{\sqrt{2\pi}t}$  to get

$$P(\zeta_1 \leq -\sqrt{n}/4) = P(Z \leq -\sqrt{n}/(4\sigma)) \leq \frac{2\sigma e^{-n/(32\sigma^2)}}{\sqrt{2\pi}n}.$$

Moreover,

$$\max_{\zeta_1 \leq c} \{f(\zeta_1)\}^{\zeta_1 = -\sqrt{n}} = 2, \quad \max_{\zeta_1 \geq c} \{f(\zeta_1)\}^{\zeta_1 = c} = \frac{2\lambda^2}{\left(\frac{3}{4}\right)^4 n^2 + \lambda^2} \leq \frac{2\lambda^2}{\left(\frac{3}{4}\right)^4 n^2} \leq \frac{7\lambda^2}{n^4}$$

Thus,

$$\mathbb{E}[(w_1 - 1)^2] \leq \frac{2\sigma^2}{n} + \frac{7\lambda^2}{n^2} + \frac{4\sigma e^{-n/(32\sigma^2)}}{\sqrt{2\pi}n}.$$

Taking it all together, we have

$$\mathbb{E}[\|\hat{\mathbf{w}}' - \mathbf{e}_1\|^2] \leq (d-1) \frac{15\sigma^6}{n\lambda^2} + \frac{2\sigma^2}{n} + \frac{7\lambda^2}{n^2} + \frac{4\sigma e^{-n/(32\sigma^2)}}{\sqrt{2\pi}n}.$$

Without optimizing too much, we simply take  $\lambda^2 = \sqrt{dn}\sigma^3 = d\sqrt{m}\sigma^3$  and get

$$\begin{aligned}\mathbb{E}[e(\hat{\mathbf{w}}')] &= \mathbb{E}[\|\hat{\mathbf{w}}' - \mathbf{e}_1\|^2] \leq \frac{2\sigma^2}{n} + \frac{22\sigma^3\sqrt{d}}{n^{3/2}} + \frac{4\sigma e^{-n/(32\sigma^2)}}{\sqrt{2\pi}n} \\ &= \frac{2\sigma^2}{md} + \frac{22\sigma^4}{m^{3/2}d} + \frac{4\sigma e^{-md/(32\sigma^2)}}{\sqrt{2\pi}md}.\end{aligned}$$

■

## Appendix H. Theorem 5 and EGU equivalences

### Proof (Theorem 5)

For reparameterization we have

$$\begin{aligned}\dot{\mathbf{w}} &= \frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \dot{\hat{\mathbf{w}}} \\ &= -\frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \nabla_{\hat{\mathbf{w}}} L \\ &= -\frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \left( \frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \right)^\top \nabla_{\mathbf{w}} L\end{aligned}$$

so the preconditioner is  $\frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \left( \frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \right)^\top$ .

For MD we have

$$\begin{aligned}\dot{\mathbf{w}} &= \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}} \dot{\tilde{\mathbf{w}}} \\ &= -\frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}} \nabla_{\mathbf{w}} L\end{aligned}$$

so the preconditioner is  $\frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}}$ .

For Riemannian GD the update

$$\dot{\mathbf{w}} = -\Gamma_{\mathbf{w}} \nabla_{\mathbf{w}} L$$

immediately implies the preconditioner is  $\Gamma_{\mathbf{w}}^{-1}$ . ■

We now analyze the implications of the theorem for EGU. EGU is defined by the mirror map (applied componentwise):

$$f(\mathbf{w}) = \log \mathbf{w}$$

This implies

$$\frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}} = \text{Diag}(\mathbf{w}) \tag{39}$$

Now consider the reparameterization (applied componentwise):

$$\hat{\mathbf{w}} = 2\sqrt{\mathbf{w}}$$

This implies

$$\frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} = \frac{1}{2} \text{Diag}(\hat{\mathbf{w}})$$

and therefore

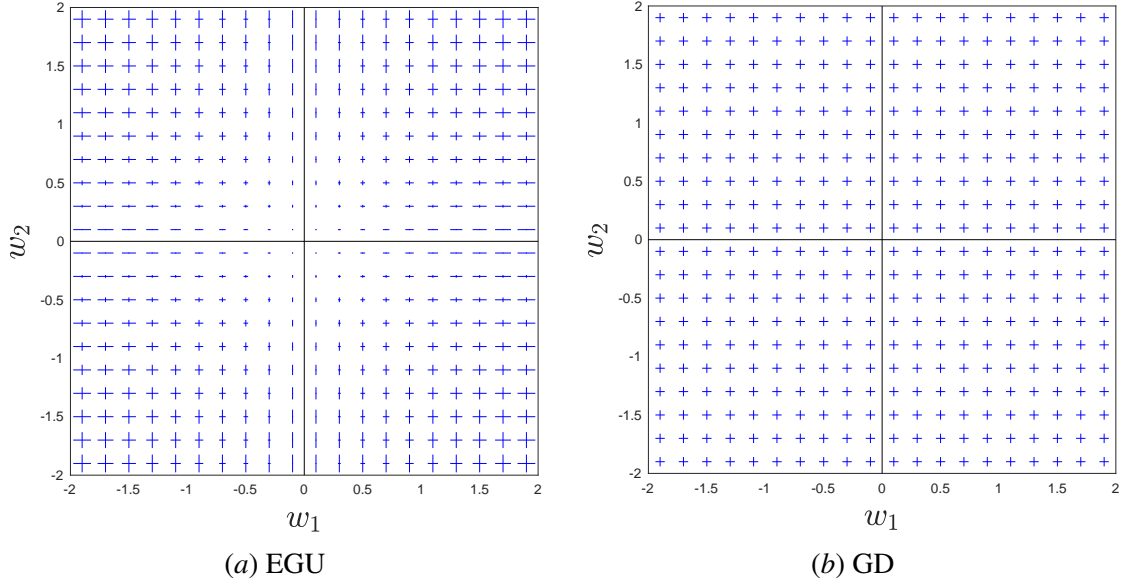
$$\frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \left( \frac{\partial \mathbf{w}}{\partial \hat{\mathbf{w}}} \right)^\top = \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{w}}}$$

as in Theorem 5. This means continuous-time EGU is equivalent to gradient flow on the spindly network when the two weights are set to be equal, i.e.  $w_i = u_i^2$  instead of  $w_i = u_i v_i$  (note that if  $\mathbf{u}$  and  $\mathbf{v}$  are initialized equally then they will remain equal) (Warmuth et al., 2021).

Using (39), Theorem 5 also implies EGU is equivalent to Riemannian GD with metric

$$\Gamma_{\mathbf{w}} = \text{Diag}(\mathbf{w})^{-1} \tag{40}$$

We visualize this metric in Figure 3 for a 2d parameter space. This geometry that is implicit in EGU helps to explain how the algorithm encourages weight trajectories to stay near sparse solutions.



**Figure 3:** (a): Metric in (40) for the Riemannian GD interpretation of EGU. (b): For comparison, the Euclidean metric implicit in GD, which is uniform and in particular rotationally symmetric. Blue lines indicate intervals of constant distance according to the respective metric.

## Appendix I. Trajectory derivations

We analyze several algorithms on the regression problem of Sections 1.3 and 2. The mean loss over the training set is

$$L(\mathbf{w}) = \frac{1}{md} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

Using  $\mathbf{X}^\top \mathbf{X} = md\mathbf{I}$ , the gradient is

$$\begin{aligned} \nabla_{\mathbf{w}} L &= 2(\mathbf{w} - \mathbf{w}^{\text{LS}}) \\ \mathbf{w}^{\text{LS}} &= \frac{1}{md} \mathbf{X}^\top \mathbf{y} \end{aligned}$$

where  $\mathbf{w}^{\text{LS}}$  is the linear least-squares solution to which all considered algorithms converge.

The continuous EGU update can be written as

$$\dot{w}_i = -w_i \nabla_{w_i} L = 2w_i(w_i^{\text{LS}} - w_i)$$

The trajectory in (10) can be directly verified to satisfy this PDE, with the initial condition  $w_i(0)$  satisfied by setting

$$c_i = \tanh^{-1} \left( \frac{2w_i(0)}{w_i^{\text{LS}}} - 1 \right)$$

The continuous EGU $\pm$  update can be written as (Amid and Warmuth, 2020)

$$\dot{w}_i = -\sqrt{w_i^2 + 1} \nabla_{w_i} L = 2\sqrt{w_i^2 + 1} (w_i^{\text{LS}} - w_i)$$

The trajectory in (11) can be directly verified to satisfy this PDE, with the initial condition  $w_i(0)$  satisfied by setting

$$\begin{aligned}\tau_i &= \sinh^{-1} \left( \frac{1 + w_i^{\text{LS}} w_i(0)}{w_i(0) - w_i^{\text{LS}}} \right) + 2t\xi \sqrt{(w_i^{\text{LS}})^2 + 1} \\ \xi &= \text{sign} (w_i(0) - w_i^{\text{LS}})\end{aligned}$$

Primed gradient flow amounts to GD under the reparameterization  $\hat{\mathbf{w}} = \text{Diag}(\mathbf{w}^{\text{LS}})^{-1} \mathbf{w}$  and so by Theorem 5 the update can be written as

$$\dot{w}_i = - (w_i^{\text{LS}})^2 \nabla_{w_i} L = 2 (w_i^{\text{LS}})^2 (w_i^{\text{LS}} - w_i)$$

The trajectory in (12) can be directly verified to satisfy this PDE and the initial condition  $w_i(0)$ .

**Adagrad:** Continuous-time Adagrad can be written as

$$\begin{aligned}\dot{w}_i &= -G_i^{-1/2} \nabla_{w_i} L = 2G_i^{-1/2} (w_i^{\text{LS}} - w_i) \\ \dot{G}_i &= \beta (\nabla_{w_i} L)^2 = 4\beta (w_i^{\text{LS}} - w_i)^2\end{aligned}$$

with preconditioner learning rate  $\beta$  and  $G_i(0) = \varepsilon$  a stability parameter.

To solve these coupled PDEs we begin by defining

$$\delta_i = w_i - w_i^{\text{LS}}$$

which leads to

$$\begin{aligned}\dot{\delta}_i &= -2G_i^{-1/2} \delta_i \\ \dot{G}_i &= 4\beta \delta_i^2\end{aligned} \tag{41}$$

Combining these two equations yields

$$\begin{aligned}\ddot{G}_i &= 8\beta \delta_i \dot{\delta}_i \\ &= -4G_i^{-1/2} 4\beta \delta_i^2 \\ &= -4G_i^{-1/2} \dot{G}_i\end{aligned}$$

which has solution

$$\begin{aligned}G_i &= \frac{16}{k_i^2} \left( W \left( -e^{-k_i(t+\ell_i)} \right) + 1 \right)^2 \\ \dot{G}_i &= -\frac{32}{k_i} W \left( -e^{-k_i(t+\ell_i)} \right)\end{aligned} \tag{42}$$

Using  $G_i(0) = \varepsilon$  and  $\dot{G}_i(0) = 4\beta (w_i^{\text{LS}} - w_i(0))^2$  allows to solve for the constants  $k_i$  and  $\ell_i$ :

$$\begin{aligned}k_i &= \frac{8}{\beta (w_i^{\text{LS}} - w_i(0))^2 + 2\sqrt{\varepsilon}} \\ \ell_i &= \frac{1}{k_i} - \frac{1}{k_i} \log \left( 1 - \frac{k_i \sqrt{\varepsilon}}{4} \right) - \frac{\sqrt{\varepsilon}}{4}\end{aligned}$$

Substituting (41) and (42) gives the corresponding expression for  $w_i$ , matching (13):

$$\begin{aligned} w_i &= w_i^{\text{LS}} + \delta_i \\ &= w_i^{\text{LS}} + \text{sign}(w_i(0) - w_i^{\text{LS}}) \frac{\sqrt{\dot{G}_i}}{2\sqrt{\beta}} \\ &= w_i^{\text{LS}} - \text{sign}(w_i^{\text{LS}} - w_i(0)) \sqrt{-\frac{8}{\beta k_i} W(-e^{-k_i(t+\ell_i)})} \end{aligned}$$

**Incremental priming and Burg MD:** We also consider an incremental version of priming where the learned weights are continuously transferred into the priming vector rather than only once at the end of a pre-training phase. Specifically, we begin with the predictive model

$$\hat{y}_t(\mathbf{x}) = (\mathbf{x} \odot \mathbf{p}_t)^\top \mathbf{w}_t$$

where  $\mathbf{p}_t$  is the priming vector,  $\mathbf{w}_t$  is the weight vector, and  $t$  indexes iterations of the learning algorithm. The idea is to maintain  $\mathbf{w}_t = \mathbf{1}$ , transferring each update of  $\mathbf{w}$  immediately into  $\mathbf{p}$ . Specifically, at each time step we make a provisional GD update

$$\begin{aligned} \tilde{\mathbf{w}}_{t+1} &= \mathbf{w}_t - \eta \nabla_{\mathbf{w}_t} L(\hat{\mathbf{y}}_t(\mathbf{X}), \mathbf{y}) \\ &= \mathbf{1} + \eta \text{Diag}(\mathbf{p}_t) \mathbf{X}^\top (\mathbf{y} - \mathbf{X} \mathbf{p}_t) \end{aligned}$$

where the second line uses the inductive assumption  $\mathbf{w}_t = \mathbf{1}$ . This assumption holds because we immediately transfer the provisional update into  $\mathbf{p}$ :

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t \odot \tilde{\mathbf{w}}_{t+1} \\ \mathbf{w}_{t+1} &= \mathbf{1} \end{aligned}$$

This transfer leaves predictions at step  $t+1$  unchanged because  $(\mathbf{x} \odot \mathbf{p}_t)^\top \tilde{\mathbf{w}}_{t+1} = (\mathbf{x} \odot \mathbf{p}_{t+1})^\top \mathbf{w}_{t+1}$  for all  $\mathbf{x}$ , but it leads the learning on step  $t+1$  to contribute immediately to priming on future steps.

We now simplify the model by dropping the inconsequential  $\mathbf{w}$  and directly computing the update for  $\mathbf{p}$ :

$$\begin{aligned} \hat{y}_t(\mathbf{x}) &= \mathbf{x}^\top \mathbf{p}_t \\ \mathbf{p}_{t+1} &= \mathbf{p}_t \odot (1 + \eta \text{Diag}(\mathbf{p}_t) \mathbf{X}^\top (\mathbf{y} - \mathbf{X} \mathbf{p}_t)) \\ &= \mathbf{p}_t - \eta \text{Diag}(\mathbf{p}_t)^2 \nabla_{\mathbf{p}_t} L(\hat{\mathbf{y}}_t(\mathbf{X}), \mathbf{y}) \end{aligned}$$

Treating  $\mathbf{p}$  as the weight vector, we have a preconditioned GD algorithm which, by Theorem 5, is equivalent in the continuous-time case to MD with mirror map  $f(\mathbf{p}) = -\mathbf{p}^{-1}$  (componentwise). This corresponds to Burg MD (Jain et al., 2007; Amid and Warmuth, 2020). (In the discrete time case, incremental priming corresponds to the “dual update” of Burg MD (Warmuth and Jagota, 1998),  $\mathbf{p}_{t+1} = \mathbf{p}_t - \eta \nabla_{f(\mathbf{p}_t)} L$  instead of  $f(\mathbf{p}_{t+1}) = f(\mathbf{p}_t) - \eta \nabla_{\mathbf{p}_t} L$ .)

Changing notation from  $\mathbf{p}$  back to  $\mathbf{w}$ , we can write the update as

$$\dot{w}_i = -w_i^2 \nabla_{w_i} L = 2w_i^2 (w_i^{\text{LS}} - w_i)$$

This has the solution

$$w_i(t) = \frac{w_i^{\text{LS}}}{W\left(\exp\left(-2\left(w_i^{\text{LS}}\right)^2(t - b_i)\right)\right) + 1}$$

$$b_i = \frac{1}{2\left(w_i^{\text{LS}}\right)^2} \log W^{-1}\left(\frac{w_i^{\text{LS}}}{w_i(0)} - 1\right)$$

where  $W$  is Lambert’s  $W$  function as above. The solution can be verified by substituting it back into the PDE.

## Appendix J. Anisotropic covariance

Theorem 2 and the analytic trajectory solutions in Section 3.2 assume spherical input covariance, meaning  $X^\top X$  is proportional to the identity. When it is not, rotationally invariant algorithms can produce trajectories that curve toward sparse solutions under certain circumstances (Figure 4a). Nevertheless, Theorem 1 implies a rotationally invariant algorithm cannot perform better on sparse over non-sparse problems with a rotationally symmetric input distribution, i.e. when averaged over all rotations of  $X^\top X$ .

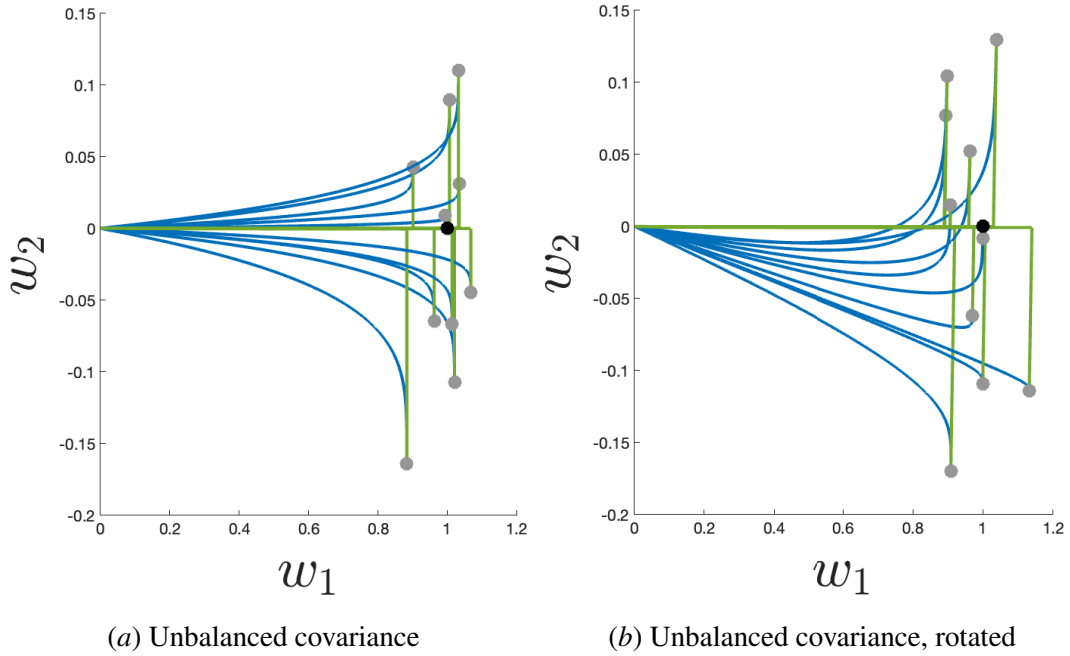
For a rotationally invariant algorithm on a linear problem, rotating the problem rotates the entire weight trajectory:  $X \rightarrow XU^\top$  implies  $w(t) \rightarrow Uw(t)$ . This allows us to understand the algorithm’s behavior with a fixed sparse target  $v$  and rotated input  $XU^\top$  by examining its behavior with a rotated target  $Uv$  and fixed input  $X$ . We illustrate this in Figure 4 for a 2-dimensional problem with  $X = H \text{Diag}(2, 1)$  so that  $X^\top X$  has condition number 4. On the left, the first principal component of  $X^\top X$  is aligned with the sparse target. GD and EGU $\pm$  both produce trajectories that bend toward the target, but for different reasons: GD’s sparsity bias depends on  $X$  while EGU $\pm$ ’s does not. This is seen in the right figure where the input is rotated. Rotation invariance of GD implies its trajectories also rotate, so that it no longer learns efficiently. Thus GD has no sparsity advantage when averaging over all rotated inputs. In contrast, EGU $\pm$  shows a sparsity bias in all cases. Because it is not rotationally invariant, rotating the problem does not rotate its trajectories (they are altered but to a much lesser degree).

## Appendix K. Fashion MNIST experiment details

In our experiments, we use a constant learning rate (which we tune for each case). We use the full batch of 60000 training examples and train each network for 5000 epochs. We first provide some visualization of the weights for the noisy case where each example is augmented with uniform noise. Figure 5 shows a subset of the weights for each network where the top slice corresponds to the image feature weights and the bottom slice corresponds to the noise feature weights. For the spindly network, the average maximum absolute value of the effective weights (i.e., the product of the two weights within each spindle) for each input neuron is 0.0182 for the image weights and 0.0025 for the noise features. The difference is less drastic for the fully-connected network, where the values are 0.0627 and 0.0568, respectively.

Next, we show the results when adding extra one-hot embeddings of the labels as features. Figure 6 shows a subset of the image and label weights for each network, along with the weights corresponding to the labels at the bottom. The spindly network assigns relatively larger weights to





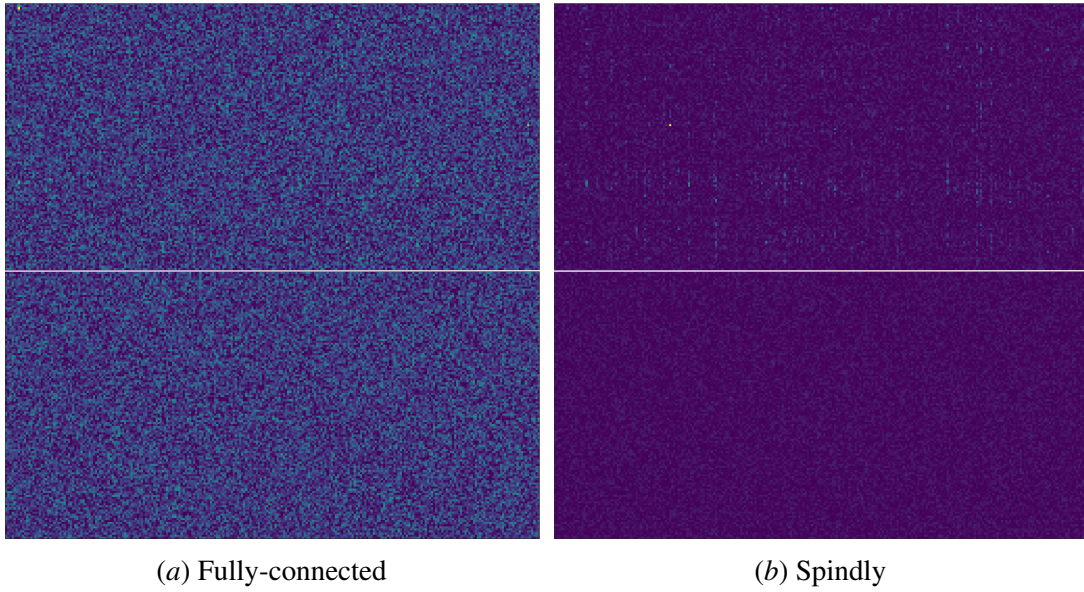
**Figure 4:** GD produces curved trajectories (blue) when the covariance  $\mathbf{X}^\top \mathbf{X}$  is nonspherical. This can speed learning then the first principal component is aligned with the target as in (a). However, GD and other rotationally invariant algorithms cannot produce a systematic bias toward sparsity. When the input distribution is rotated as in (b), the trajectories rotate as well such that they no longer learn the sparse target efficiently. In contrast, non-rotation invariant algorithms such as EGU± (green) can learn sparse targets efficiently under any rotation of the input.

the label features. The average maximum absolute value of the weights for each neuron is 0.7834 for the label weights, whereas the image and noise weights have values 0.0057 and 0.0025, respectively. Again, the difference between the label weights and the rest is less prominent for the fully-connected network: The average maximum absolute values are 0.4213 for label weights and 0.0604 and 0.0584 for the image and noise weights, respectively.

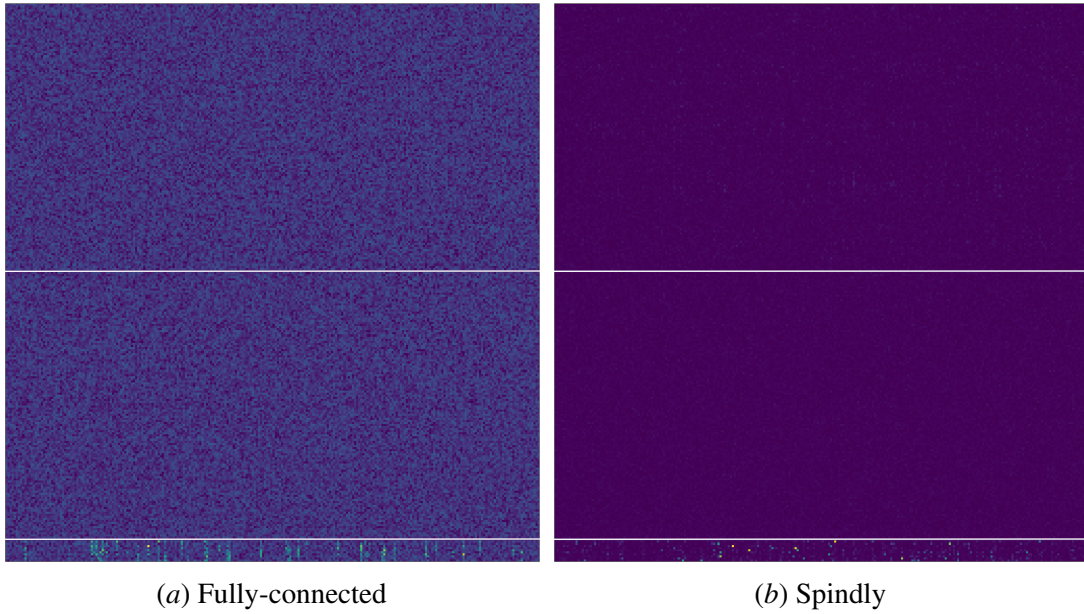
In summary, the network with the fully connected input layer is more easily confused by noisy features. When noisy and informative features are added then the network with a spindlified input layer finds a model that almost entirely relies on the informative features whereas the net with the fully connected input layer still makes use of original and noisy features.

**More network details:** Three layer fully connected net with RELU transfer functions, 10 outputs, square loss, trained with batch gradient descent for 5000 epochs, batch size 60000.

Number of weights in the net with the fully connected input layer: input size 784, 256 nodes on the first two hidden layers, 10 outputs, for a total of 268.000 weights. Adding a copy of noisy input features almost doubles the number of weights. The number of informative features is  $10 * 256 = 2560$ . It is important to note that the number of examples is much larger than the input dimension but much smaller than the total number of weights.



**Figure 5:** The weights of the first layer when trained with images augmented with noise. The top slice corresponds to the image feature weights and the bottom slice corresponds to the noise feature weights.



**Figure 6:** The weights of the first layer when trained with images augmented with noise and one-hot representation of the labels. The top slice corresponds to the image feature weights, the middle slice corresponds to the noise feature weights, and the bottom corresponds to the label weights.