The downside of rotation invariance in neural net training

author names withheld

Editor: Under Review for ALT 2024

Abstract

Any gradient descent trained neural net with a fully-connected input layer is rotation invariant when initialized with a rotation invariant distribution. This means that if the input instances are rotated then the trained weight vectors at the input layer counter rotate and the effect is vacuous. We show that learning with a rotation invariant algorithm is fundamentally limited in that such an algorithm cannot sample efficiently learn sparse linear functions. In contrast, there are simple non-rotation invariant updates for a single linear neurons that learn sparse linear functions exponentially faster. Also gradient descent on a 2-layer linear network where each hidden node is connected to a single input can achieve the same feat.

In the lower bounds, we give the algorithm a rotated version of the input instances and then prove that the Bayes optimal algorithm for this setup has a certain lower bound for linear sparse targets that can be easily be avoided by simple non-rotation invariant algorithms. We believe that this general proof technique will be useful for proving lower bounds for other families of algorithms that admit other classes of invariances.

Keywords: rotation invariance, gradient descent, feed forward neural networks, lower bounds, multiplicative updates, sparsity.

1. Introduction

In (Warmuth et al., 2021) a lower bound was proven for any algorithm that predicts with a rotation invariant function. This lower bound on the expected square loss of a random example is essentially $1 - k/d$ when learning a sparse linear function, where d is the input dimension and $1 \leq k \leq d$ is the number of training examples seen. The lower bound is surprising because it holds for predicting with any gradient descent trained neural net with a fully-connected input layer whose initial distribution is rotation invariant. In particular the lower bound holds for any structure of the hidden layers and transfer functions at the hidden nodes. However the lower bound is of limited interest because it becomes vacuous when the number of training examples k is at least the input dimension d. This is of course the case in most applications.

In this paper we prove lower bounds for the same class of algorithms for the more relevant over constraint case. The target class used is now sparse linear *plus Gaussian noise*. We clearly show that the rotation invariance of the algorithm is responsible for the lower bound:

1. Our lower bound technique creates a Bayesian setup where the learning is presented with a randomly rotated version of the input instances and we prove a lower bound on the Bayes optimal algorithm for this case.

2. We show that there are trivial non-rotation invariant algorithm that can learn noisy sparse linear much more efficiently: Multiplicative updates on a linear neuron or gradient descent on a two-layer linear net in which every hidden node is connected to exactly one input node (The gradient flow case gradient descent on such networks is equivalent to multiplicative updates on a linear neuron (Amid and Warmuth, 2020)).

Thus the lower bound can be circumvented by pivoting away from gradient descent to a multiplicative update on just a linear neuron or by changing the structure of the network so that gradient descent can access individual features. Both methods are of course not rotation invariant. Our proof technique is interesting in its own right and is different from the technique used for the under-constrained case. So far we have no unified proof method for both the under and over constraint cases.

ADDITIONAL RELATED WORK:

There is a long history for contrasting the generalization ability of additive versus multiplicative updates (See e.g. Kivinen et al. (1997); Kivinen and Warmuth (1997)). Additive (or gradient descent) updates subract multiples of the gradient from the weight vectors and multiplicative updates multiply the weights by factors that have a multiple of the same gradients in the exponent. Multiplicative updates perform dramatically better when the target is sparse. Surprisingly there is a connection between both update families discovered recently: Roughly when the weights are products of parameters, then the algorithms is biased towards sparsity (Gunasekar et al., 2017; Kerekes et al., 2021). In the gradient flow case, multiplicative updates on a linear neuron have been shown to be equivalent to gradient descent on a two-layer net in which every hidden node is connect to exactly one input. One property of additive updates is that they are rotation invariant when the loss depends on dot products between the instances and weight vectors. The purpose of this research is to show that this invariance has a downside because such algorithms can only make use of dot products between instances and cannot access individual features which is necessary for learning sparse targets efficiently.

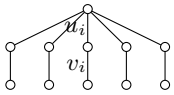


Figure 1: Spindly.

2. The lower bound method

2.1. Rotation invariance and problem setup

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

A *learning algorithm*, is a mapping, which given the training set $(\boldsymbol{X}, \boldsymbol{y})$, produces a real-valued *prediction function* $\mathbb{R}^d \ni \boldsymbol{x} \mapsto \widehat{y}(\boldsymbol{x}) \in \mathbb{R}$. To accentuate the fact that the prediction function depends on the training data, we will sometimes denote it by $\widehat{y}(\boldsymbol{x}|\boldsymbol{X}, \boldsymbol{y})$. With a slight abuse of the definition, we allow the function value of $\widehat{y}$ to be randomized (a random variable), based on some internal randomness of the algorithm (such as the initialization of weight vectors).

A learning algorithm $\widehat{y}(\cdot\,|\boldsymbol{X}, \boldsymbol{y})$ is called *rotation invariant* (Warmuth and Vishwanathan, 2005; Warmuth et al., 2021) if for any orthogonal matrix $\underset{d,d}{\boldsymbol{U}}$ and any input $\boldsymbol{x} \in \mathbb{R}^d$:

$$\widehat{y}(\boldsymbol{U}\boldsymbol{x}\,|\boldsymbol{X}\boldsymbol{U}^\top, \boldsymbol{y}) = \widehat{y}(\boldsymbol{x}\,|\boldsymbol{X}, \boldsymbol{y}) \tag{1}$$

In other words, the prediction $\widehat{y}(\boldsymbol{x}|\boldsymbol{X}, \boldsymbol{y})$ for any input $\boldsymbol{x}$ remains the same if we rotate both $\boldsymbol{x}$ and all examples from $\boldsymbol{X}$ by the same orthogonal matrix $\boldsymbol{U}$. If the algorithm is randomized, with random variable Z denoting the entire internal randomization of the algorithm,[1] then $\widehat{y}$ is a random

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

variable given by some function $\widehat{y}(\boldsymbol{x}_{\text{te}}|\boldsymbol{X},\boldsymbol{y}) = f_{\boldsymbol{x}_{\text{te}},\boldsymbol{X},\boldsymbol{y}}(Z)$, and the equality sign in equation (1) of the above definition of rotation invariance should be interpreted as "identically distributed".

Our lower bounds in sections 2.2-2.3 will 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 nodes in the input layer are initialized randomly with a rotation invariant distribution (e.g. i.i.d. Gaussians), which is trained by gradient descent, is rotation invariant, and is thus subject to our lower bound. That is, learning with any function of the following form is rotation invariant: $f(\boldsymbol{w}_1 \cdot \boldsymbol{x}, \boldsymbol{w}_2 \cdot \boldsymbol{x}, \ldots, \boldsymbol{w}_h \cdot \boldsymbol{x}, \boldsymbol{\theta})$, where the $\boldsymbol{w}_i$ are initialized by a rotationally symmetric distribution, are updated with gradient descent and the additional weights $\boldsymbol{\theta}$ are updated in any manner that depends on the input only via $\boldsymbol{w}_i \cdot \boldsymbol{x}$ (i.e., via the computation in the first layer). The reason is that the gradient $\nabla_{\boldsymbol{w}_i} f$ is equal to the instance $\boldsymbol{x}$ times a scalar that depends on $\boldsymbol{x}$ only via $\boldsymbol{w}_i \cdot \boldsymbol{x}$. Therefore it is easy to show, by induction on t, that rotating all instances by some orthogonal matrix $\boldsymbol{U}$ results in the same rotation of $\boldsymbol{w}_{i,t}$ for all i and t, and therefore that the rotation has no effect on $\boldsymbol{x} \cdot \boldsymbol{w}_i$. In contrast learning with $f(u_1 v_1 x_1, u_2 v_2 x_2, \ldots, u_d v_d x_d)$ (with parameters $\boldsymbol{u}, \boldsymbol{v}$) is not rotation invariant.

2.2. 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 2.3 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) a rotationally symmetric input distribution $p_{\text{in}}(\tilde{\boldsymbol{X}})$ with the input matrix $\underset{n+1,d}{\tilde{\boldsymbol{X}}} = [\boldsymbol{X}, \boldsymbol{x}_{\text{te}}^\top]$ consisting of the training matrix $\boldsymbol{X}$ of size n and the test example $\boldsymbol{x}_{\text{te}}$ (b) an observation model $q(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{X}})$ which gives the joint conditional distribution over n training outcomes and a test outcome, $\underset{n+1}{\tilde{\boldsymbol{y}}} = [\boldsymbol{y}, y_{\text{te}}]$, and (c) a loss function $\mathcal{L}(\widehat{y}, y)$. The task of any algorithm will be to produce predictions $\widehat{y}(\boldsymbol{x}_{\text{te}}|\boldsymbol{X},\boldsymbol{y})$ to minimize the loss on the test outcomes, $\mathcal{L}(\widehat{y}, y_{\text{te}})$. 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 $\underset{d,d}{\boldsymbol{U}}$, define the rotated observation model

$$q_{\boldsymbol{U}}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{X}}) = q(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{X}}\boldsymbol{U}^\top)$$

Now define a new learning problem by first sampling $\boldsymbol{U}$ uniformly (under the Haar measure p_{H}) and then generating observations according to $q_{\boldsymbol{U}}$. This is equivalent to a symmetrized observation model $\mathring{q}$ that is a mixture over all $q_{\boldsymbol{U}}$:

$$\mathring{q}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{X}}) = \int q_{\boldsymbol{U}}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{X}}) \mathrm{d}p_{\text{H}}(\boldsymbol{U})$$

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

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

Thus $\mathring{q}$ 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

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

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

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

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

This loss is bounded by

$$L_{\widehat{y}}(q) \geq L_{\mathcal{B}}(\mathring{q})$$

Proof For any orthogonal $\boldsymbol{U}$, the algorithm's expected loss is

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

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

As we show in the remainder of 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 $\boldsymbol{U}$ with elements of $\mathcal{T}$ and requiring p_{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, natural gradient on network in Fig. 1 is rotation invariant (See discussion in (Kerekes et al., 2021)). Thus the below lower bounds for sparse linear apply to the natural gradient algorithm, whereas vanilla gradient descent on this network approximates EGU$^\pm$ and we will show it breaks the lower bound.

2.3. Lower bound for least-squares regression

We demonstrate how the bound implied by 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 3.

In the specific problem class we consider, the number of training examples is $n = md$ for some integer m, and $\boldsymbol{X}$ consists of m stacked copies of a matrix $\boldsymbol{H} = \sqrt{d}\underset{d,d}{\boldsymbol{V}}$ (i.e. $\boldsymbol{X} = [\underbrace{\boldsymbol{H}; \ldots; \boldsymbol{H}}_{\times m}]$), where $\boldsymbol{V}$ is a random orthogonal matrix (distributed according to the Haar measure). The test input is one of the rows of $\boldsymbol{H}$, $\boldsymbol{x}_{\mathrm{te}} = \boldsymbol{h}_k$, with index k drawn uniformly at random. As the input distribution is based on drawing a random orthogonal matrix $\boldsymbol{V}$, its rotational symmetry is straightforward to verify. We assume that the labels are the first feature of $\tilde{\boldsymbol{X}}$ plus Gaussian noise, that is

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

which can be be equivalently written as

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{e}_1 + \boldsymbol{\xi}, & \boldsymbol{\xi} &\sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_{md}) \\ y_{\mathrm{te}} &= \boldsymbol{h}_k^\top\boldsymbol{e}_1 + \xi_{\mathrm{te}}, & \xi_{\mathrm{te}} &\sim N(0, \sigma^2). \end{aligned}$$

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 ξ_{te}. The choice of $\boldsymbol{e}_1$ as opposed to any other $\boldsymbol{e}_i$ is made w.l.o.g. Indeed, Theorem 1 implies that a rotationally invariant algorithm will have the same loss for $\boldsymbol{X}\boldsymbol{e}_1$ as it will for $\boldsymbol{X}\boldsymbol{w}$ for any other unit vector $\boldsymbol{w}$.

The accuracy of prediction $\widehat{y} = \widehat{y}(\cdot|\boldsymbol{X}, \boldsymbol{y})$ on the test set $(\boldsymbol{h}_k, y_{\mathrm{te}})$ is measured by the *squared loss* $\mathcal{L}(\widehat{y}, y_{\mathrm{te}}) = (\widehat{y} - y_{\mathrm{te}})^2$. For any choice of a random row index $k \in \{1, \ldots, d\}$, denote the corresponding prediction as $\widehat{y}_k = \widehat{y}(\boldsymbol{h}_k|\boldsymbol{X}, \boldsymbol{y})$, and let $\widehat{\boldsymbol{y}} = (\widehat{y}_1, \ldots, \widehat{y}_d)$ denote the vector of all such predictions. Similarly, let $\boldsymbol{y}_{\mathrm{te}}$ denote the vector of test labels for all choices of k, i.e., $y_{te_k} = \boldsymbol{h}_k^\top\boldsymbol{e}_1 + \xi_{\mathrm{te}}$, which can be jointly written as $y_{\mathrm{te}} = \boldsymbol{H}\boldsymbol{e}_1 + \xi_{\mathrm{te}}\boldsymbol{1}$ with $\boldsymbol{1} = (1, \ldots, 1)$. The expected value of the loss over the random choice of $k \in \{1, \ldots, d\}$ and over the independent test label noise is given by:

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

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

$$e(\widehat{\boldsymbol{y}}) = \mathbb{E}_{k,\xi_{\mathrm{te}}}[\mathcal{L}(\widehat{y}, y_{\mathrm{te}})] - \mathbb{E}_{k,\xi_{\mathrm{te}}}[\mathcal{L}(\widehat{y}^\star, y_{\mathrm{te}})] = \frac{1}{d}\|\widehat{\boldsymbol{y}} - \boldsymbol{H}\boldsymbol{e}_1\|^2$$

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

$$e(\widehat{\boldsymbol{w}}) = \frac{1}{d}\|\boldsymbol{H}\widehat{\boldsymbol{w}} - \boldsymbol{H}\boldsymbol{e}_1\|^2 = \frac{1}{d}(\widehat{\boldsymbol{w}} - \boldsymbol{e}_1)^\top \underbrace{\boldsymbol{H}^\top \boldsymbol{H}}_{d\boldsymbol{I}}(\widehat{\boldsymbol{w}} - \boldsymbol{e}_1) = \|\widehat{\boldsymbol{w}} - \boldsymbol{e}_1\|^2 \tag{3}$$

The proof of the lower bound for this problem closely follows Section 2.2 and uses Theorem 1.

1. We start with a Bayesian setting with the rotated observation model $q(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{X}}\boldsymbol{U}^\top)$ for a random orthogonal matrix $\boldsymbol{U}$. However, in the considered linear regression setup, this is equivalent to simply rotating the target weight vector by $\boldsymbol{U}^\top$, because $\tilde{\boldsymbol{X}}\boldsymbol{U}^\top\boldsymbol{e}_1 = \tilde{\boldsymbol{X}}(\boldsymbol{U}^\top\boldsymbol{e}_1)$. This means that we can equivalently consider a linear model $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{X}}\boldsymbol{w} + \tilde{\boldsymbol{\xi}}$, where $\boldsymbol{w}$ is generated randomly from a prior distribution uniform over a unit sphere $\mathcal{S}^{d-1} = \{\boldsymbol{w} \in \mathbb{R}^d \colon \|\boldsymbol{w}\| = 1\}$. Given the square loss function and linear observation model, the optimal Bayes predictor is based on the posterior mean, $\mathbb{E}[\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}]$.

2. Even though the posterior mean does not have a simple analytic form for prior distribution over a unit sphere, we use the results from (Marchand, 1993; Dickel, 2016) to show that the Ridge Regression (RR) predictor (which is the Bayes predictor for the Gaussian prior) with appropriately chosen regularization constant has the expected error by 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.

3. We show that the error of the RR predictor is at least $\frac{\sigma^2}{\sigma^2+m}$. This means that 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.

4. 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 $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{X}}\boldsymbol{e}_1 + \tilde{\boldsymbol{\xi}}$.

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

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

Note the lower bound does not hold for any fixed choice of $\boldsymbol{X}$ (of full rank) such as stacked version of the d dimensional Hadamard matrix (which is a fixed rotation of $\sqrt{d}\boldsymbol{I}$). Hadamard matrices were used extensively in previous lower bound proofs for sparse problems (Kivinen et al., 1997; Warmuth and Vishwanathan, 2005). For any fixed full-rank $\boldsymbol{X}$, there exists a row vector $\boldsymbol{v}$ s.t. $\boldsymbol{v}\boldsymbol{X} = \boldsymbol{e}_1^\top$. Now the linear algorithm $\widehat{y}(\boldsymbol{x}|\boldsymbol{X}, \boldsymbol{y}) = \boldsymbol{v}\boldsymbol{X}\,\boldsymbol{x}$ achieves minimal loss σ^2 while being trivially rotationally invariant because $\boldsymbol{v}\boldsymbol{X}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{x} = \boldsymbol{v}\boldsymbol{X}\,\boldsymbol{x}$.

Proof of Theorem 2: We consider the aforementioned rotated observational model, which can be defined as follows. Let $\boldsymbol{w} \in \mathbb{R}^d$ be a weight vector drawn uniformly from a unit sphere $\mathcal{S}^{d-1} = \{\boldsymbol{w} \in \mathbb{R}^d \colon \|\boldsymbol{w}\| = 1\}$. The algorithm is given data set $(\boldsymbol{X}, \boldsymbol{y})$ with $\boldsymbol{X} = [\boldsymbol{H}; \ldots; \boldsymbol{H}]$ being m copies of $\boldsymbol{H} = \sqrt{d}\boldsymbol{V}$, and $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{w} + \boldsymbol{\xi}$, where $\boldsymbol{\xi} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_{dm})$ is a vector of Gaussian i.i.d.

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

$$e(\widehat{\boldsymbol{w}}|\boldsymbol{w}) = \|\widehat{\boldsymbol{w}} - \boldsymbol{w}\|^2.$$

It is well-known (see, e.g., Berger (1985)) that the expected squared error, $\mathbb{E}_{\boldsymbol{w},\boldsymbol{\xi}}\left[e(\widehat{\boldsymbol{w}}|\boldsymbol{w})\right]$ (with expectation with respect to the prior and the label noise) is minimized by the *posterior mean* $\widehat{\boldsymbol{w}}^\star = \mathbb{E}_{\boldsymbol{w}|\boldsymbol{y}}[\boldsymbol{w}]$, that is the mean value of $\boldsymbol{w}$ with respect to the posterior distribution $q(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$. 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; Dickel, 2016). Let $\widehat{\boldsymbol{w}}_{RR}$ be the ridge regression estimator:

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

which is the posterior mean itself (and thus optimal) when the prior over $\boldsymbol{w}$ is Gaussian with zero mean and covariance $\frac{1}{d}\boldsymbol{I}$ (the covariance is multiplied by factor d^{-1} to have $\mathbb{E}[\|\boldsymbol{w}\|^2] = \mathbb{E}[\boldsymbol{w}\boldsymbol{w}^\top] = \mathrm{tr}(\frac{1}{d}\boldsymbol{I}) = 1$ 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 $\widehat{\boldsymbol{w}}^\star$:

Lemma 3

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

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

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

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

Theorem 4 *(Theorem 2 by Dickel (2016), Theorem 3.1 by Marchand (1993)) Let $n = md$ and let $s_1 \geq \ldots \geq s_d$ denote the eigenvalues of $n^{-1}\boldsymbol{X}^\top\boldsymbol{X}$. Then*

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

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

$$\mathbb{E}_{\boldsymbol{w},\boldsymbol{\xi}}\left[e(\widehat{\boldsymbol{w}}_{RR}|\boldsymbol{w})\right] \leq \mathbb{E}_{\boldsymbol{w},\boldsymbol{\xi}}\left[e(\widehat{\boldsymbol{w}}^\star|\boldsymbol{w})\right] + \frac{\sigma^2}{n + d\sigma^2} = \mathbb{E}_{\boldsymbol{w},\boldsymbol{\xi}}\left[e(\widehat{\boldsymbol{w}}^\star|\boldsymbol{w})\right] + \frac{1}{d}\frac{\sigma^2}{m + \sigma^2}. \qquad \blacksquare$$

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

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

and thus

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

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

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

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

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

to get:

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

Using this together with Lemma 3 gives the lower bound on the Bayes optimal predictor in the rotated observational model.

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

We now use the fact that that $\boldsymbol{H} = \sqrt{d}\boldsymbol{V}$ with orthogonal matrix $\boldsymbol{V}$ 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 $\boldsymbol{w} = \boldsymbol{e}_1$. ■

Note that the proof would significantly simplify if we assumed from the start that the target weight vector $\boldsymbol{w}$ is generated from a Gaussian distribution $N(\boldsymbol{0}, \frac{1}{d}\boldsymbol{I}_d)$ rather than from a unit sphere $\mathcal{S}^{d-1}$ (both priors give unit squared norm of $\boldsymbol{w}$ 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); Dickel (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 $\boldsymbol{w}$.

3. 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 do this with an approximated version of the unnormalized multiplicative update EGU$^{\pm}$. We use the same setup as in the lower bound, i.e. $\boldsymbol{X}$ consists of m stacked copies of a matrix $\boldsymbol{H} = \sqrt{d}\boldsymbol{V}$, when $\boldsymbol{V}$ is a rotation matrix and $\boldsymbol{y}$ is sparse linear (the first component) plus Gaussian noise with variance σ^2. The only difference is that for the lower bound, $\boldsymbol{V}$ was randomly chosen, but the upper bounds hold for *any* rotation matrix $\boldsymbol{V}$. The upper bound on the error that we obtain[2] is essentially $O(\sigma^2\frac{\log d}{md})$.

We begin with a bound for the normalized version of the multiplicative update called EG$^{\pm}$. The proof relies on the fact that, with high probability, the smallest coordinate of the gradient is the first one, and with large learning rate, EG$^{\pm}$ becomes argmax over the negative gradient coordinates, which is $\boldsymbol{e}_1$. EG$^{\pm}$ makes use of the fact that the norm of the linear target $\boldsymbol{e}_1$ is 1 and this additional knowledge allows the speedup.[3] We then prove our main upper bound for the unnormalized EGU$^{\pm}$.

3.1. Upper bound for the Exponentiated Gradient update

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

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

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

$$L(\boldsymbol{w}) = \frac{1}{n}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}_t\|^2.$$

Theorem 5 *The expected error of the batch EG$^{\pm}$ algorithm after the first iteration is bounded by*

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

Proof sketch: The gradient of the loss can be written down as $2(\boldsymbol{w} - \boldsymbol{e}_1 - \boldsymbol{\zeta})$, 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 down to $2de^{-\eta}$. If the high probability even does not hold, we bound the error by its maximum value 4. ■

2. Note that at this point, we do not consider general input matrices $\boldsymbol{X}$ in the upper bounds. The arbitrary covariance structure makes the analysis much more complicated and the general case is left for future research.
3. We can also provide an upper bound on the error of the online version of EG$^{\pm}$ for an arbitrary input matrix $\boldsymbol{X}$ 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.

3.2. Upper bound for the Approximated Unnormalized Exponentiated Gradient update

We now drop the normalization of EG$^{\pm}$ and use the approximation $\exp(x) \approx 1 + x$.[4] The resulting approximation of the unnormalized EGU$^{\pm}$ algorithm updates its weights as follows:

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

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

Theorem 6 *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 by $T = 4\sqrt{d}$ steps, achieves error bounded by:*

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

Proof sketch: 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* $\eta\sqrt{\boldsymbol{w}_t^2 + 4\beta^2}$. Since $\beta = 1/(2d)$, and $\boldsymbol{w}_1 = \mathbf{0}$, these learning rates are initially small and of order $O(1/d)$. We then show by a careful analysis of the update, that for all coordinates except the first one, the weights $w_{t,i}$ $(i > 1)$ remain small and so do the associated learning rates. Therefore, after T steps, the algorithm does not move significantly away from zero on these coordinates. In turn, 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 o increasing $w_{t,1}$. Eventually $w_{T+1,1}$ gets very close to 1 after T steps of the algorithm. ■

4. Experimental visualization

The main focus of this paper is the lower bound for rotation invariant algorithms and how it can be circumvented. Nevertheless some experimental visualization of the behavior of the algorithms is helpful. Figure 2 shows the behavior of two rotation-invariant algorithms (GD and ridge regression) and three non-invariant ones (EGU$^{\pm}$, the Approximated EGU$^{\pm}$ analyzed in Theorem 6, and gradient descent on the Spindly network of Figure 1) on the regression problem from Sections 2.3 and 3. 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 6, showing that the bound we were able to prove is weak (conservative). Note that all algorithms here require early stopping to minimize their error, although the dips in the loss curves for GD and ridge regression are too small to be visible.

4. 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).

In Theorem 5 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.

Finally, the bounds of Sections 2.3 and 3 and the curves of Figure 2 assume the instance matrix $\boldsymbol{X}$ comprises m copies of a scaled d dimensional rotation matrix. This unusual setup was needed for technical reasons. However if we instead use a Gaussian instance matrix $\underset{md,d}{\boldsymbol{X}}$ together with sparse linear targets plus Gaussian noise, then the curves are nearly identical to those in Figure 2 (not shown).

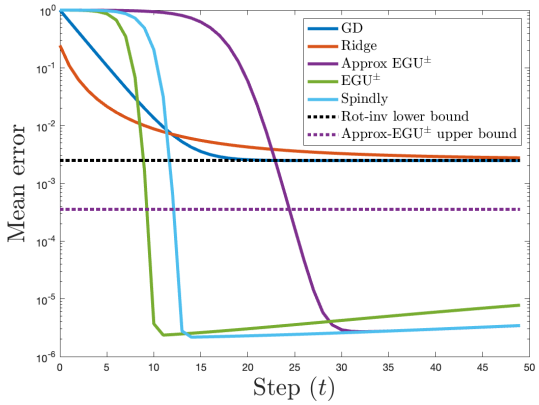


Figure 2: Excess test loss averaged over 100 runs of the linear regression problem in sections 2.3-3, using $d = 1024$, $m = 100$, $\sigma = 1/2$. Learning rates are $\eta_{GD} = 0.1$, $\eta_{\text{Approx EG}^{\pm}} = 1/2$ (per Theorem 6, also $\beta = 1/2d$), $\eta_{\text{EGU}^{\pm}} = 1$, $\eta_{\text{Spindly}} = 1/4$. Ridge regression is plotted with regularization parameter $\lambda = 10^5/t$. 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 6 (using $\delta = .001$).

5. Noise experiments on Fashion MNIST

We perform experiments on the Fashion MNIST (Xiao et al., 2017) dataset using a multilayer feed-forward 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 behave similarly, although the fully-connected network achieves higher test accuracy (85% for fully-connected vs. 81% for spindly). Next, we double the number of features of the examples by augmenting each example with uniformly sampled noise in the range $[-1, 1]$. With noisy augmentation, the spindly network achieves 80% test accuracy while the fully-connected network gets only to 69%. 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 shows that Gradient Descent has 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 the appendix.

6. Conclusion and open problems

We gave a lower bound for rotation invariant neural nets. Our work suggests the following approach: The structure of the network plus the weight update imply a certain set of invariances and the invariances lead to lower bounds for the Bayes optimal algorithm of the model constructed from the invariances. For example a certain two-sided invariance characterizes the matrix version of multiplicative updates (Warmuth et al., 2014). Is it thus possible to prove lower bounds for these algorithms for dense linear targets that are beaten by vanilla gradient descent? A more immediate question is what are the invariances when the fully-connected bottom layer of a gradient descent trained neural net uses convolution. Can such networks learn sparse linear problems? Also transformers (Vaswani et al., 2017) should be investigated with this method. Transformers clearly have an ability to access individual tokens. The question is which structural feature of transformers enables them to do that and learn sparse linear problems. Partial results along this line recently appeared in Abernethy et al. (2023).

Another interesting question is when transforming the instances (i.e. a kernelization) "helps" rotation invariant algorithms. Note that linear neurons that are fed transformed instances when updated with gradient descent again have the $1 - k/d$ lower bound in the noise-free underconstraint case (Warmuth and Vishwanathan, 2005). Proving similar lower bounds for the noisy over constraint case as done in this paper would be a first step.

Finally, the updates discussed in this paper focus on mirror descent and its approximations. However there is large body of work based on L_1 regularization that is also biased towards sparsity (See e.g. Tibshirani (2015); Axiotis and Yasuda (2023); Hoff (2017)). How these updates are related to mirror descent is an interesting research topic.

Recall that multiplicative update are mirror descent using the $\log$ link functions. By streching the link in the EG$^{\pm}$ version it was shown that EG$^{\pm}$ has gradient descent as a special case (Ghai et al., 2020). Actually we observe that this stretching method can be used to enhance any mirror descent update with a hyper parameter that can realize gradient descent as a special case. However no practical algorithm have been developed based on this observation.

Finally, the "full versions" of many commonly 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. The inductive biases of all these algorithms need to be analyzed. For example, in our preliminary work (not shown) we see that diagonalized AdaGrad is biased away from sparsity in the weight vectors. Using sparsity as a yard stick, we show that dramatic differences can occur that provably cannot be overcome by enhancing gradient descent with any number of hidden layers. Therefore a thorough investigation of mirror descent for training neural nets should also achieve large differences that have not been explored.

References

Ruslan Abdulkadirov, Pavel Lyakhov, and Nikolay Nagornov. Survey of optimization algorithms in modern neural networks, 04 2023.

Jakob Abernethy, Alekh Agarwal, Teodor V. Marinov, and Manfred K. Warmuth. A mechanism for sample-efficient in-context learning for sparse retrieval tasks. *arXiv preprint arXiv:2305.17040*, 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.

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.

Kyriakos Axiotis and Taisuke Yasuda. Performance of ℓ_1 regularization for sparse convex optimization, 2023.

James O. Berger. *Statistical decision theory and Bayesian analysis*. Springer, 1985.

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.

Lee H. Dickel. Ridge regression and asymptotic minimax estimation over spheres of growing dimension. *Bernoulli*, 22(1):1–37, 2016.

Udaya Ghai, Elad Hazan, and Yoram Singer. Exponentiated gradient meets gradient descent. In *Conference on Learning Theory (COLT)*, volume 117, pages 1–23. PMLR, 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.

Peter D. Hoff. Lasso, fractional norm and structured sparse estimation using a hadamard product parametrization. *Computational Statistics & Data Analysis*, 115:186–198, 2017.

Anna Kerekes, Anna Mészáros, and Ferenc Huszár. Depth without the magic: Inductive bias of natural gradient descent. *ArXiv*, abs/2111.11542, 2021.

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.

Eric Marchand. Estimation of a multivariate mean with constraints on the norm. *The Canadian Journal of Statistics*, 21(4):359–366, 1993.

R. J. Tibshirani. A general framework for fast stagewise algorithms. *Journal of Machine Learning Research*, 16(Dec):2543–2588, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

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 Shuisheng Zhou. Kernelization of matrix updates. *Journal of Theoretical Computer Science*, 558:159–178, 2014.

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 Theorem 5

Here we prove that the batch version of EG$^{\pm}$ achieves small error already after one trail, when the learning rate is set to a sufficiently large value (Theorem 5)

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

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

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

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

We compute the expression for the gradient:

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

where

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

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

$$\nabla L(\boldsymbol{w}) = 2(\boldsymbol{w} - \boldsymbol{e}_1 - \boldsymbol{\zeta}), \tag{5}$$

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

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

We also bound the error of the algorithm from above:

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

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

Consider the weights of the batch EG$^\pm$ algorithm after just a *single* trial, that is $\boldsymbol{v}_2^+$ and $\boldsymbol{v}_2^-$ Using (5) and noting that $\boldsymbol{w}_1 = \mathbf{0}$, 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:

$$v_{2,1}^+ = \frac{e^{2\eta(1+\zeta_1)}}{Z_2}, \qquad v_{2,1}^- = \frac{e^{-2\eta(1+\zeta_1)}}{Z_2}, \qquad Z_2 = \sum_{i=1}^{d} e^{2\eta(\delta_{1i}+\zeta_i)} + e^{-2\eta(\delta_{1i}+\zeta_i)}, \tag{6}$$

We will now lower-bound $v_{2,1}^+$, and later use a relation which directly follows from (6):

$$v_{2,1}^- = e^{-4\eta(1+\zeta_1)}v_{2,1}^+. \tag{7}$$

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

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

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

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

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

Note that for any $i \ge 2$,

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

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 (6) ($\zeta_i = -1/4$ would result in the same value of these weights). This gives:

$$\begin{aligned} v^+_{2,1} &\geq \frac{e^{2\eta(1+\zeta_1)}}{e^{2\eta(1+\zeta_1)} + e^{-2\eta(1+\zeta_1)} + (d-1)(e^{\eta/2} + e^{-\eta/2})} \\ &= \frac{1}{1 + e^{-4\eta(1+\zeta_1)} + e^{-2\eta(1+\zeta_1)}(d-1)(e^{\eta/2} + e^{-\eta/2})} \\ &\geq \frac{1}{1 + e^{-4\eta(1-1/4)} + e^{-2\eta(1-1/4)}(d-1)(e^{\eta/2} + e^{-\eta/2})} \\ &\geq \frac{1}{1 + e^{-3\eta} + e^{-3/2\eta}2(d-1)e^{\eta/2}} \\ &\geq \frac{1}{1 + e^{-3\eta} + 2(d-1)e^{-\eta}} \geq \frac{1}{1 + (2d-1)e^{-\eta}} \end{aligned} \tag{8}$$

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 (7). 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:

$$e(\boldsymbol{w}_2) \leq 2(1 - v^+_{2,1} + v^-_{2,1}) \leq (2d-1)e^{-\eta} + e^{-2\eta} \leq 2de^{-\eta}$$

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

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

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

Appendix B. Derivation of the Approximated (EGU$^{\pm}$) algorithm

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, $\boldsymbol{v}^+_t$ and $\boldsymbol{v}^-_t$, and the prediction vector is given by $\boldsymbol{w}_t = \boldsymbol{v}^+_t - \boldsymbol{v}^-_t$. Let β denote the initial value of weights, that is $\boldsymbol{v}^+_1 = \boldsymbol{v}^-_1 = \beta\mathbf{1}$. The weights are updated according to

$$\boldsymbol{v}^{\pm}_{t+1} = \boldsymbol{v}^{\pm}_t \odot e^{\mp\eta\nabla L(\boldsymbol{w}_t)} = \beta e^{\mp\eta\sum_{j=1}^{t}\nabla L(\boldsymbol{w}_j)}. \tag{9}$$

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

$$\boldsymbol{v}_t^+ = \frac{\boldsymbol{w}_t + \sqrt{\boldsymbol{w}_t^2 + 4\beta^2}}{2}, \qquad \boldsymbol{v}_t^- = \frac{-\boldsymbol{w}_t + \sqrt{\boldsymbol{w}_t^2 + 4\beta^2}}{2}. \tag{10}$$

Expanding the EGU$^\pm$ update (9) in the learning rate we get

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

Dropping the $O(\eta^2)$ term and using (10) gives

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

so that the update becomes (4):

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

Appendix C. Proof of Theorem 6

Using (5), $\nabla L(\boldsymbol{w}_t) = 2(\boldsymbol{w}_t - \boldsymbol{e}_1 + \boldsymbol{\zeta})$, the update (4) becomes:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - 2\eta \sqrt{\boldsymbol{w}_t^2 + 4\beta^2}\, (\boldsymbol{w}_t - \boldsymbol{e}_1 + \boldsymbol{\zeta}) \tag{11}$$

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 δ, we can solve for γ:

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

This means the with probability at least $1 - \delta$, $|\zeta_i| \leq \gamma$ for all $i = 1, \ldots, d$. 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 (11) in terms of $\boldsymbol{s}_t = \boldsymbol{w}_t - \boldsymbol{e}_1 - \boldsymbol{\zeta}$

$$\boldsymbol{s}_{t+1} = (1 - 2\eta \sqrt{(\boldsymbol{s}_t + \boldsymbol{e}_1 + \boldsymbol{\zeta})^2 + 4\beta^2}) \boldsymbol{s}_t \tag{12}$$

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 $2\eta\sqrt{(s_{t,i}+\zeta_i)^2 + 4c^2} < 1$; this condition is ensured by noticing that

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

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:

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

so that

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

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

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

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

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

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

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

This expression is increasing in ζ_i^2 so we can upper-bound it by

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

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

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

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

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

Thus, the total error from 'zero-signal' coordinates is

$$\sum_{i=2}^{d} w_{T+1,i}^2 \le \frac{8\sigma^2\ln\frac{2d}{\delta}}{md} \tag{13}$$

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

Analysis for the 'signal' coordinate. Using (12), 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 of $\eta = 1/4$, because

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

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

$$s_{t+1} = (1 - 2\eta\sqrt{(r - s_t)^2 + 4\beta^2}s_t, \qquad s_1 = r, \tag{14}$$

Already after the first iteration,

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

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 (14) and upper bound

$$s_{t+1} \le (1 - 2\eta(r - s_t))s_t \tag{15}$$

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

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

which give:

$$\frac{s_t}{r - s_t} = Ce^{-\eta rt} \quad \Longrightarrow \quad s_t = \frac{r}{1 + Ce^{2\eta rt}}.$$

Inspired by this we will bound $\frac{s_t}{r-s_t}$. From (15) we get:

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

so that:

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

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

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

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 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 $1 - \frac{1}{\sqrt{8d}} \overset{d\geq 4}{\geq} 1 - \frac{1}{4\sqrt{2}}$, we get $1 + 2\eta r >= \frac{3}{2} - \frac{1}{8\sqrt{2}} > e^{1/3}$ (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

$$(1 - w_{T+1,1})^2 = (s_{T+1,i} + \zeta_i)^2 \leq 2s_{T+1,i}^2 + 2\zeta_i^2 \leq 9e^{-\frac{8}{3}\sqrt{d} + 2\ln d} + \frac{2\sigma^2 \ln \frac{2d}{\delta}}{md}. \tag{16}$$

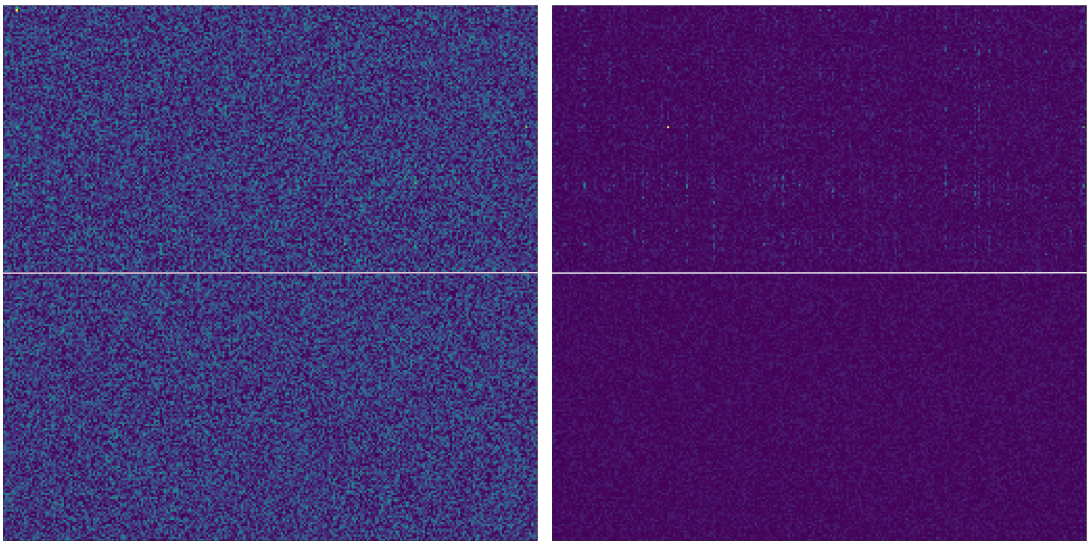
Bound the error of Spindly algorithm The final error of the algorithm is obtained by summing (13) and (16):

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

Appendix D. Fashion MNIST experiment details

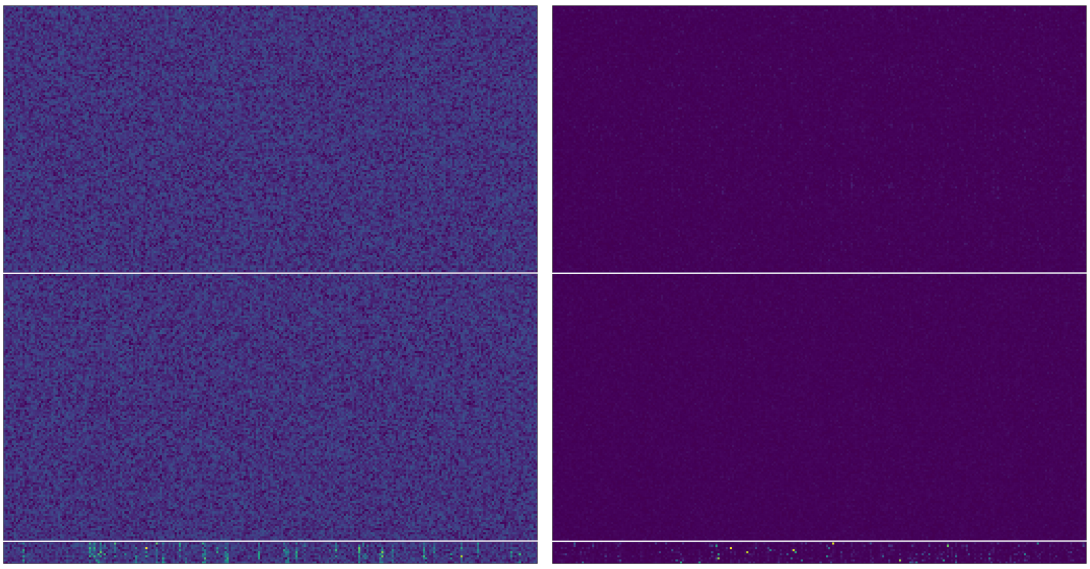
In our experiments, we use a constant learning rate (which we tune for each case). We use the full batch of 60000 training examples and train each network for 1000 epochs. We first provide some visualization of the weights for the noisy case where each example is augmented with unifrom noise. Figure 3 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 4 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 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.



(*a*) Fully-connected (*b*) Spindly

Figure 3: 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.



(*a*) Fully-connected (*b*) Spindly

Figure 4: 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.