DEPTH WITHOUT THE MAGIC: INDUCTIVE BIAS OF NATURAL GRADIENT DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

In gradient descent, changing how we parametrize the model can lead to drastically different optimization trajectories, giving rise a surprising range of meaningful inductive biases: identifying sparse classifiers or reconstructing low-rank matrices without explicit regularization. This implicit regularization has been hypothesised to be a contributing factor to good generalization in deep learning. However, natural gradient descent is approximately invariant to reparameterization, it always follows the same trajectory and finds the same optimum. The question naturally arises: What happens if we eliminate the role of parameterization, which solution will be found, what new properties occur? We characterize the behaviour of natural gradient flow in deep linear networks for separable classification under logistic loss and deep matrix factorization. Some of our findings extend to nonlinear neural networks with sufficient but finite over-parametrization. We demonstrate that there exist learning problems where natural gradient descent fails to generalize, while gradient descent with the right architecture peforms well.

1 INTRODUCTION

There is plenty of empirical evidence that the choice of network architecture is an important determinant of the success of deep learning (He et al., 2015; Vaswani et al., 2017). The empirical observations are now supported by theoretical work into the role that parameter-to-hypothesis mapping plays in determining inductive biases of gradient-based learning. Unregularized gradient descent can efficiently find low-rank solutions in matrix completion problems (Arora et al., 2019), sparse solutions in separable classification (Gunasekar et al., 2018) or compressed sensing (Vaškevičius et al., 2019). Valle-Pérez et al. (2018) studied deep neural networks and found evidence that the parameter-hypothesis mapping[1] is biased towards simpler functions as measured by Kolmogorov complexity. Taken together, these observations and findings have lead the community to hypothesize that

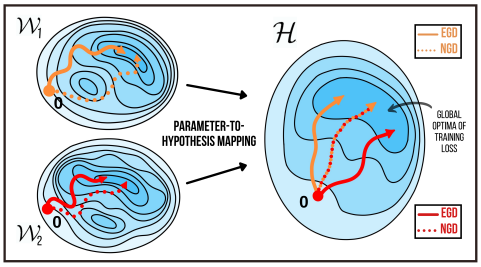

Figure 1: Illustration of parametrization-dependence of EGD and independence of NGD. Consider two parameter spaces ($\mathcal{W}_1$, $\mathcal{W}_2$) and two optimization trajectories in each: one EGD, one NGD. If we map these into the hypothesis space ($\mathcal{H}$) then EGD finds different optima, but NGD finds the same.

The parameter-to-hypothesis mapping influences the inductive biases of gradient-base learning and may play an important role in generalization.

In parallel to improving architectures, considerable research was done to improve optimization algorithms for deep learning, with a focus on faster convergence and robustness to hyperparameters. Among the most advanced optimization methods are natural gradient descent (NGD) techniques. An intuitive motivation for NGD is that it improves convergence by implicitly lifting the problem from parameter-space, where the loss is non-convex and poorly behaved to the Riemannian manifold

[1]The mapping between the parameter space and the set of hypotheses as seen on Figure 1

of hypotheses, where the loss is better behaved. From the perspective of inductive biases, the most interesting aspect of NGD is its approximate invariance to reparametrization.

Natural gradient descent eliminates the effect of parameter-to-hypothesis mapping.

These two observations invite questions about the nature of inductive biases in NGD as well as the role of parametrization-dependence in generalization. The first, practical, implication is as follows: if the parameter-to-hypothesis mapping really does play an important role in generalization, then eliminating its influence on the optimization path may be undesirable, and consequently the pursuit of implementing exact NGD in deep architectures may be counterproductive. Secondly, studying the behaviour of NGD in various models and tasks may give us new insights about the importance of parametrization, and could perhaps offer a way to experimentally or theoretically test hypotheses.

In this paper we study the inductive bias of natural gradient descent in deep linear models. These models are particularly suited for our analysis because (a) efficient algorithms exist to calculate exact natural gradients which is otherwise computationally intractable and (b) the inductive biases of Euclidean gradient descent (EGD) in these models have been thoroughly studied and understood.

We make the following contributions:

- In linear classification, we show that NGF is invariant under invertible transformations of data (Theorems 1&2) and as a consequence it cannot recover the ℓ_p large margin solutions that EGD tends to converge to.
- We further show that when the number of parameters exceeds the number of datapoints, NGF interpolates training labels in a way similar to ordinary least squares or ridgeless regression (Theorems 3&4).
- We demonstrate experimentally that there exist learning problems where NGD can not reach good generalization performance, while EGD with the right architecture can succeed.
- To perform experiments, we extended the work of Bernacchia et al. (2018) to derive efficient and numerically stable algorithms for calculating exact natural gradients in diagonal networks (Gunasekar et al., 2018) and deep matrix factorization (Arora et al., 2019).

Before stating our main theoretical and experimental results we review some relevant background on parametrization-dependent implicit regularization and natural gradients.

2 Background

2.1 Separable Classification with Deep Linear Models

In this article we consider binary classification datasets $\{(\mathbf{x}_n, y_n), n = 1, \dots, N\}$ separable by a homogeneous linear classifier with a positive margin (i. e. $\exists \boldsymbol{\beta}^*$ s.t. $y_n \mathbf{x}_n^\top \boldsymbol{\beta}^* \geq 1 \; \forall n$). (We use the notation $X = (\mathbf{x}_1 \cdots \mathbf{x}_N)^\top$). In such situation $\boldsymbol{\beta}^*$ is not unique and there may be many separating hyperplanes which all achieve 0 training loss - it is up to the inductive biases of the learning algorithm to select one. Soudry et al. (2017) studied the dynamics of unregularized Euclidean gradient descent on logistic loss and found that the iterate $\boldsymbol{\beta}(t)$ converges to the well-known ℓ_2 large margin classifier in direction, that is

$$\lim_{t\to\infty} \frac{\boldsymbol{\beta}(t)}{|\boldsymbol{\beta}(t)|} = \frac{\boldsymbol{\beta}^*_{\ell_2}}{|\boldsymbol{\beta}^*_{\ell_2}|} \text{ where } \boldsymbol{\beta}^*_{\ell_2} = \underset{\boldsymbol{\beta}\in\mathbb{R}^D}{\arg\min} ||\boldsymbol{\beta}||_2 \text{ s.t. } y_n \mathbf{x}_n^\top \boldsymbol{\beta} \geq 1 \quad \forall n.$$

Importantly, Gunasekar et al. (2018) later showed that this behaviour changes if the gradient descent is performed on a different parametrization. In this paper we will focus on L-layer linear diagonal networks (Gunasekar et al., 2018), where $\boldsymbol{\beta} = \mathbf{w_1} \odot \mathbf{w_2} \odot \ldots \odot \mathbf{w_L}$, using $\odot$ to denote elementwise product. When we adjust parameters $\mathbf{w_1}, \dots, \mathbf{w_L}$ through Euclidean gradient descent, $\boldsymbol{\beta}(t)$ converges to the $\ell_{\frac{2}{L}}$ large margin separator defined as

$$\lim_{t\to\infty} \frac{\boldsymbol{\beta}(t)}{|\boldsymbol{\beta}(t)|} = \frac{\boldsymbol{\beta}^*_{diag}}{|\boldsymbol{\beta}^*_{diag}|} \text{ where } \boldsymbol{\beta}^*_{diag} = \underset{\boldsymbol{\beta}\in\mathbb{R}^D}{\arg\min} ||\boldsymbol{\beta}||_{\frac{2}{L}} \text{ s.t. } y_n \mathbf{x}_n^\top \boldsymbol{\beta} \geq 1 \quad \forall n.$$

A remarkable consequence of this is that unregularized gradient descent can find sparse classifiers, without any form of explicit regularization. In fact, this inductive bias is even more sparsity-seeking than the typically used ℓ_1 regularization (see e. g. Koh et al., 2007; Tibshirani, 1996). Figure 2 illustrates this behaviour in a 2D example.

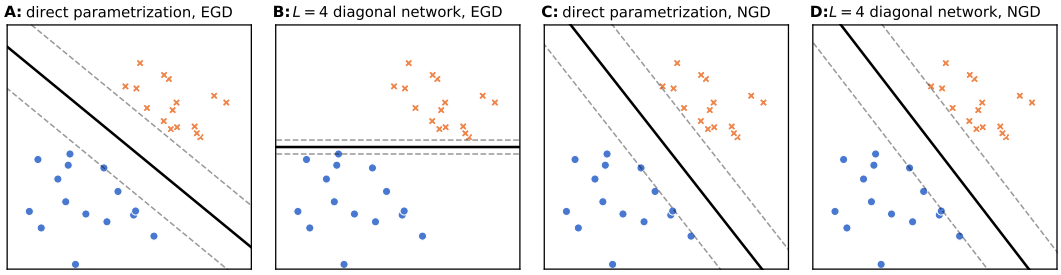

Figure 2: Implicit regularization of EGD and NGD on logistic loss in separable classification. EGD reaches different optima depending on parametrization: fully connected networks reach ℓ_2 large margin (**A**), while L-layer linear diagonal networks reach the $\ell_{\frac{2}{L}}$-large margin solution which favours sparsity (**B**), while L-layer linear diagonal networks reach the $\ell_{\frac{2}{L}}$-large m. NGD converges to the same optimum irrespective of the parametrization (**C, D**).

2.2 Matrix Completion via Deep Matrix Factorization

The task of matrix completion involves recovering an unknown matrix $\boldsymbol{\beta}^* \in \mathbb{R}^{D\times D}$ from a randomly chosen subset of observed entries[2]. The problem is clearly underdefined: there are infinitely many matrices that match the observed entries. It is common to make additional assumptions about $\boldsymbol{\beta}^*$, most commonly that that it has low rank, under which it becomes identifiable.

One approach to matrix completion under the low-rank assumption is based on explicit regularization (e.g. nuclear norm) which leads to a convex optimization problem. Another common approach is matrix factorization using an underparametrized representation $\boldsymbol{\beta} = UV$ where the sizes of $U \in \mathbb{R}^{D\times R}$ and $V \in \mathbb{R}^{R\times D}$ are restricted to ensure $\boldsymbol{\beta}$'s rank is at most R. Learning then proceeds by minimizing the non-convex mean-squared reconstruction error in U, V via gradient descent.

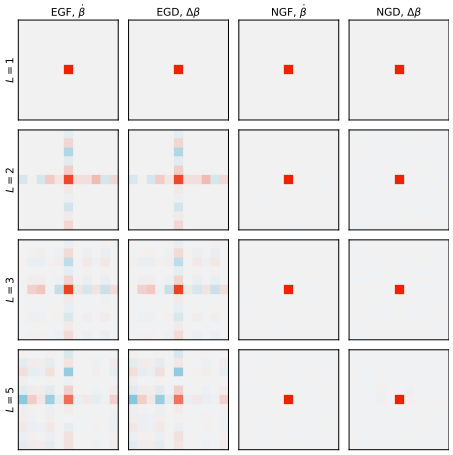

Figure 3: Illustration of the descent direction in EGF, EGD, NGF and NGD (*left to right*) in different depth matrix factorization models (*top to bottom*). The algorithms minimize the squared error on a single observation at the middle of the matrix. Each panel shows how entries of the full 11×11 matrix change from a random initial state. When $L \geq 2$, Euclidean gradient methods also move entries where there is no observation - enabling implicit regularization towards low-rank solutions. By contrast, and due to invariance, natural gradient methods move only the single entry to match the observation.

Remarkably, Gunasekar et al. (2017) showed that the gradient-based matrix factorization method tends to converge to low-rank solutions even in the overparametrized setting, i.e. when $\boldsymbol{\beta} = W_1 W_2$ where W_1 and W_2 are full square matrices, without any explicit regularization. This was later extended by Arora et al. (2019), who studied the deep matrix product parametrization of the form $\boldsymbol{\beta} = W_1 W_2 \cdots W_L$. Arora et al. (2019) ran experiments for different matrix completion tasks varying initialization, depth and number of observations and compared them to minimum nuclear norm solution. When the number of observed entries is large gradient descent in deep matrix factorization models tended to the minimum nuclear norm solution, but for fewer observed entries, which is the interesting case, this was not the case. Gradient descent preferred solutions with lower effective rank at the expense of higher nuclear norm. From the evolution of the singular values of $\boldsymbol{\beta}$ they also concluded that the implicit regularization is towards low rank that becomes stronger as depth grows.

[2]to simplify presentation we assume the matrices are square, but our arguments hold more generally.

2.3 NATURAL GRADIENT DESCENT

In the next section we briefly introduce some notation and key properties of natural gradient descent (NGD, Amari, 1997; Pascanu & Bengio, 2013). Intuitively, one can think of NGD as a gradient descent method, but not in the Euclidean space (with the Euclidean metric) of parameters, but instead on the Riemannian manifold of probabilistic models the parameters define (equipped with a different metric). More specifically, let's say that the parameter of interest is θ, where θ defines a probabilistic model $p(y|\mathbf{x}, \theta)$. We assume that we wish to minimize the log loss under this model, i. e. $l(\theta, \mathbf{x}, y) = -\log p(y|\mathbf{x}, \theta)$ and $\mathcal{L}(\theta) = \sum_{n=1}^{N} l(\theta, \mathbf{x}_n, y_n)$. Then, NGD is usually defined as

$$\theta(t+1) = \theta(t) - \eta F^{-1}(\theta)\nabla_\theta \mathcal{L}(\theta), \tag{1}$$

where η is the step size and

$$F(\theta) = \mathbb{E}_X[\mathbb{E}_{Y|X;\theta}[\nabla_\theta \mathcal{L}(\theta)\nabla_\theta^\top \mathcal{L}(\theta)]] \tag{2}$$

is the average Fisher information matrix. In the above definition, $\mathbb{E}_{Y|X;\theta}$ is taken over the distribution specified by θ, but distribution with respect to which the expectation $\mathbb{E}_X$ is calculated can be arbitrarily chosen. In this article we use the empirical distribution of training data, though other choices are possible Pascanu & Bengio (2013). We will also consider natural gradient flow (NGF) the continuous limit of NGD, analogously defined as

$$\dot{\theta} = -F^{-1}(\theta)\nabla_\theta \mathcal{L}(\theta). \tag{3}$$

We also note, that $F(\theta)$ is not generally invertible, and indeed it will not be in some of the cases we will consider. Therefore, it is more correct to define NGF as any trajectory θ_t which satisfies

$$F(\theta)\dot{\theta} = -\nabla_\theta \mathcal{L}(\theta). \tag{4}$$

The natural gradient direction is thus only unique within the eigenspace of $F(\theta)$. Of all natural gradient directions, one common choice is to use the Moore-Penrose pseudoinverse of F:

$$\dot{\theta} = -F^{+}(\theta)\nabla_\theta \mathcal{L}(\theta). \tag{5}$$

We have seen how in EGD, different parametrization of the same problem leads to drastically different trajectories and optima. However, NGD with infinitesimally small learning rate (i. e. NGF) always follows the same trajectory in model-space and this finds the same optimum, irrespective of how it is parametrized, provided that the parametrization is smooth and locally invertible. Below we formally state this property Amari (1997), alongside a short proof for illustration.

Statement (Invariance of NGF under reparametrization)**.** *Let $\mathbf{w}$ and θ be two parameter vectors related by the mapping $\theta = \mathcal{P}(\mathbf{w})$ and consider natural gradient flow in $\mathbf{w}$. Assume that (1) the Jacobian $J = \frac{\partial \theta_t}{\partial \mathbf{w}_t}$ and (2) $F(\theta_t)$ are both full rank for all t. If $\mathbf{w}_t$ follows natural gradient flow starting from $\mathbf{w}_0$ then $\theta_t = \mathcal{P}(\mathbf{w}_t)$ follows NGF, i. e. it solves $\dot{\theta}_t = -F(\theta_t)^{+}\nabla_{\theta_t}\mathcal{L}(X, \theta_t)$.*

Proof. We use that $F(\mathbf{w}) = J^\top F(\theta) J$ which follows from the definition of F:

$$F(\mathbf{w}) = \mathbb{E}_X[\nabla_{\mathbf{w}}\mathcal{L}(X, \mathbf{w})\nabla_{\mathbf{w}}^\top \mathcal{L}(X, \mathbf{w})] = \mathbb{E}_X[J^\top \nabla_\theta \mathcal{L}(X, \theta)\nabla_\theta^\top \mathcal{L}(X, \theta) J] = J^\top F(\theta) J$$

The invariance statement follows:

$$\begin{aligned}(\dot{\mathcal{P}(\mathbf{w_t})}) = J\dot{\mathbf{w_t}} = -JF(\mathbf{w}_t)^{+}\nabla_{\mathbf{w}_t}\mathcal{L}(X, \mathbf{w}_t) &= -JJ^{+}F(\theta_t)^{+}(J^\top)^{+}J^\top \nabla_{\theta_t}\mathcal{L}(X, \theta_t) = \\ &= -F(\theta_t)^{+}\nabla_{\theta_t}\mathcal{L}(X, \theta_t).\end{aligned}$$

3 NATURAL GRADIENTS UNDER LOGISTIC LOSS ON SEPARABLE DATA

We have seen in Section 2.1 that when trained on separable data with the logistic loss EGD tends to converge to large margin classifiers. To illustrate how NGD differs, we first prove an invariance property which, as we will see, rules out large margin behaviour. We state this property separately when $N < D$ and when $N \geq D$ in the theorems that follow.

Theorem 1. *Let's assume, that $N < D$, X is full rank and A is an invertible $D \times D$ matrix. Let $\boldsymbol{\beta}_t = \boldsymbol{\beta}_t(X, \mathbf{y})$ be the trajectory of NGF and $\boldsymbol{\beta}'_t = \boldsymbol{\beta}_t(XA^\top, \mathbf{y})$ (the trajectory of NGF on data $XA^\top$). Then $X\boldsymbol{\beta} = XA^\top\boldsymbol{\beta}'$.*

Proof sketch. We use the notation $\mathbf{s} = X\boldsymbol{\beta}$ and $\mathbf{s}' = XA^\top\boldsymbol{\beta}$ and prove that $\mathbf{s}_t = \mathbf{s}'_t$. The full proof can be found in Appendix C.1.

Theorem 2. *Let $\boldsymbol{\beta}_t(X, \mathbf{y})$ be the trajectory of NGF and let A be a $D \times D$ invertible transformation. If $N \geq D$, X has full rank and we consider NGF on the transformed data $XA^\top$, then $A^\top\boldsymbol{\beta}_t(XA^\top, \mathbf{y}) = \boldsymbol{\beta}_t(X, \mathbf{y})$.*

Remark. When $N \geq D$ and X is full rank, the size of $F(\boldsymbol{\beta})$ is $D \times D$ and its rank is D, therefore the Fisher information matrix of $\boldsymbol{\beta}$ is invertible.

Proof sketch. First let's say $\boldsymbol{\beta}'_t = \boldsymbol{\beta}_t(XA^\top, \mathbf{y})$ and $\boldsymbol{v}^\top = \boldsymbol{\beta}'^\top A$. Then we prove the following:

$$\nabla_{\boldsymbol{\beta}'}\mathcal{L}(y_n\boldsymbol{\beta}'^\top A\mathbf{x}_n) = A\nabla_{\boldsymbol{v}}\mathcal{L}(y_n\boldsymbol{v}^\top\mathbf{x}_n) \tag{6}$$

$$F(\boldsymbol{\beta}') = AF(\boldsymbol{v})A^\top. \tag{7}$$

Hence we get:

$$\dot{\boldsymbol{\beta}}' = F(\boldsymbol{\beta}')^{-1}\nabla_{\boldsymbol{\beta}'}\mathcal{L}(\boldsymbol{\beta}') \quad \text{and} \quad \dot{\boldsymbol{v}} = F(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}). \tag{8}$$

So if $\boldsymbol{v}$ and $\boldsymbol{\beta}'$ have the same initialization, then $\boldsymbol{v}_t = \boldsymbol{\beta}'_t$. Full proof can be found in Appendix C.2.

Conclusion. *Let $u_t(X, \mathbf{y})$ denote the trajectory of $X\boldsymbol{\beta}_t$, which is the linear function $\boldsymbol{\beta}^\top\mathbf{x}$ evaluated at each of the datapoints $\mathbf{x}_n$. Then $u_t(XA^\top, \mathbf{y}) = u_t(X, \mathbf{y})$.*

Proof.

$$u_t(XA^\top, \mathbf{y}) = XA^\top\boldsymbol{\beta}_t(XA^\top, \mathbf{y}) = X\boldsymbol{\beta}_t(X, \mathbf{y}) = u_t(X, \mathbf{y})$$

□

One special case of this invariance property is invariance to scaling the dimensions of input data (when A is diagonal). Imagine we scale any dimension by a constant a, NGF counteracts it by scaling the corresponding coordinate of β by a^{-1}. We see now why this rules out characterising implicit regularization of NGD as minimizing non-data-dependent norms of $\boldsymbol{\beta}$. In particular, it rules out the ℓ_p large-margin behaviour we have seen in EGD.

Remark. Let A be a $D \times D$ invertible transformation and let $\boldsymbol{\beta}^*(X, \mathbf{y})$ be the ℓ_2 large margin solution, *i.e.* $\boldsymbol{\beta}^*(X, \mathbf{y}) = \operatorname{argmin} \|\boldsymbol{\beta}\|_2$ subject to $y_n\boldsymbol{\beta}^\top\mathbf{x}_n \geq 1\ \forall n$. Then the ℓ_2 large margin classifier does not have the invariance property, namely there exists a dataset (X,y) and a transformation A such that $A^\top\boldsymbol{\beta}^*_t(XA^\top, \mathbf{y}) \neq \boldsymbol{\beta}^*_t(X, \mathbf{y})$. We include a proof by counterexample in Appendix D.

Having ruled out norm-based implicit regularization, it's natural to consider other statistical methods that exhibit invariance under invertible data transformations. One candidate is ridge-less regression or ordinary least squares (OLS), whose parameter is given by the formula $\boldsymbol{\beta}_{\text{OLS}} = (X^\top X)^{-1}X^\top y$. As it turns out, the connection between NGD in linear regression and the OLS estimate run deeper than sharing this invariance property.

Theorem 3. *If $N < D$ and X is full rank, if parameters $\boldsymbol{\beta}_t$ of a linear model follow natural gradient flow under logistic loss, the logits $\mathbf{s}_t = X\boldsymbol{\beta}_t$ follow an asymptotically linear trajectory with direction vector $\mathbf{y}$.*

Remark. The Fisher information matrix w.r.t. $\boldsymbol{\beta}$ is $F(\boldsymbol{\beta}) = X^\top D(\boldsymbol{\beta})X$, where $D(\boldsymbol{\beta})$ is diagonal with positive elements on the diagonal. We see, that $\text{rank}(F) = \text{rank}(X) < N$, so the NGF is not well-defined. When $\text{rank}(X) = D$, $\boldsymbol{\beta}$ has $N - D$ degrees of freedom and we did describe $\boldsymbol{\beta}$ on D dimensions.

This Theorem follows from the more general Theorem 4 which we will state later.

Informally, when we have more parameters than datapoints, NGD discovers a solution that interpolates the training labels $\mathbf{y}$ (encoded as -1s and $+1$s perfectly) just like ordinary least squares does in this case. Furthermore, if one uses the Moore-Penrose pseudoinverse to calculate the descent direction, i. e. Eqn. (5), then $\boldsymbol{\beta}_t$ converges in direction to the OLS parameter.

Ordinary least-squares interpolation works very differently from large-margin methods for classification. In the language of large margin methods, $\mathbf{s}_t$ converging to $\mathbf{y}$ in direction means that all datapoints become *support vectors*. This can also happen in ℓ_2 large margin methods, where it is

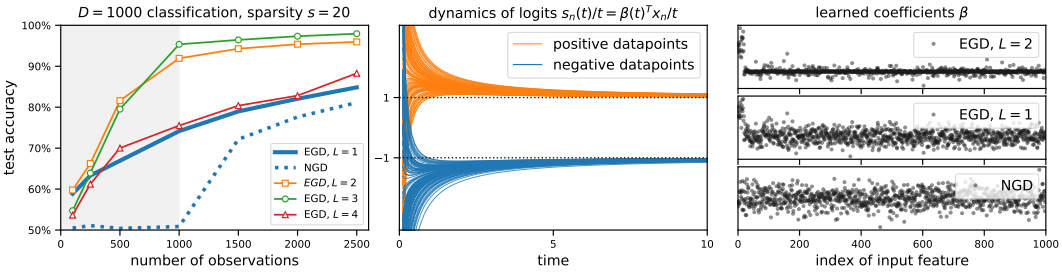

Figure 4: NGD and EGD in a 1000 dimensional sparse classification task, where the ground truth classifier has 20 non-zero components. *Left:* Test accuracy of EGD depends on parametrizaion. When there are there are fewer datapoints than dimensions, EGD with 2 or 3-layer diagonal parametrization can reach up to 90% accuracy. By contrast, NGD performs at chance level, and is worse than EGD even when $N \geq D$. *Middle:* Under NGD, when $N < D$, logits of the model grow linearly, proportional to the binary label. *Right:* Coefficient vector $\boldsymbol{\beta}$ learnt by EGD in different architectures and NGD when $N = 2500$: In the 2-layer diagonal network, corresponding to ℓ_1 implicit regularisation, β becomes sparse. In the 1-layer model, the solution is substantially less sparse, but the overall structure is learnt. NGD fails to learn the sparse structure.

called support vector proliferation, but only when $D > N \log N$ (Hsu et al., 2020). Consequences of this phenomenon to generalization are discussed in Hsu et al. (2020).

Theorem 3 provided useful insight into the inductive bias of NGD in linear models but it turns out it is relatively straightforward to generalize this result to Theorem 4 which holds for non-linear overparametrized models as well.

Theorem 4. *Let* $\mathbf{w} \in \mathbb{R}^K, K \geq N$ *be the parameters of a classifier with logits* $\mathbf{s} = s(X; \mathbf{w}) \in \mathbb{R}^N$. *If* $\mathbf{w}_t$ *follows natural gradient flow on the logistic loss with labels* $\mathbf{y}$ *and the Jacobi matrix* $J_t = \frac{\partial \mathbf{s}_t}{\partial \mathbf{w}_t}$ *is of full rank, then* $\mathbf{s}_t$ *grows asymptotically linearly with direction vector* $\mathbf{y}$.

Remark. If our network is linear $J = X$, so Theorem 3 is a special case of Theorem 4 indeed.

Proof sketch. The main idea is that, since J is full rank, by parametrization invariance of NGD the trajectory of $\mathbf{s}$ is determined by the trajectory of the corresponding $\boldsymbol{\beta}$.

$$\dot{\mathbf{s}} = -F^{-1}(\mathbf{s}) \nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s}) \tag{9}$$

Then we can calculate $F(s)$ which turns out to be diagonal, so we have N independent differential equations. We solve them to get the result. The details of the proof can be found in the Appendix C.

3.1 Experiments

In order to validate and illustrate our findings we have run two main simulations, with results presented in Figures 2 and 4. In both experiments we considered the direct parametrization $\boldsymbol{\beta} = \mathbf{w}$ and the diagonal parametrization $\boldsymbol{\beta} = \mathbf{w}_1 \odot \cdots \odot \mathbf{w}_L$ (Gunasekar et al., 2018) for different depth L. In order to run these experiments we needed to implement an efficient algorithm for computing natural gradients in these models: naively calculating and then inverting the Fisher information matrix is computationally inefficient and numerically unstable. We therefore developed an algorithm that exploits the structure in the Fisher information matrix, extending the work of Bernacchia et al. (2018) for diagonal networks. The details of our algorithms can be found in Appendix B.2.

In Experiment 1 we illustrated EGD and NGD in a 2D toy classification dataset. Positive and negative classes were generated such that they are separable by the the axis-aligned separator, but there exists a non-axis-aligned separator with a higher margin. Based on the findings of Gunasekar et al. (2018) we expected EGD to find the large margin solution when L is low, and the axis-aligned solution when L is sufficiently large. The results in panels a and b of Figure 2 confirm these predictions. Figure 2c-d illustrate the parametrization-independence of NGD: it converges to the same solution irrespective of parametrization. The solution is different from both the EGD solutions.

In Experiment 2 we focused on generalization performance. We generated a 1000-dimensional dataset with standard Gaussian X, and a sparse ground-truth separator whose first 20 components

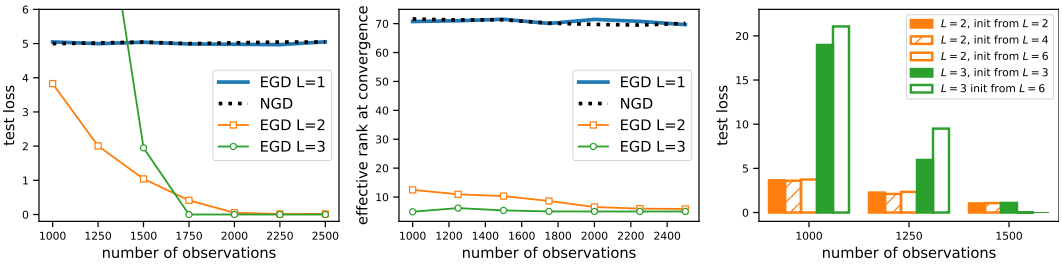

Figure 5: Performance of unregularized EGD and NGD in rank-5 matrix completion tasks using different architectures. *Left and Middle:* Using deep matrix product parametrizations with $L \geq 2$ layers, EGD can reach low training error and identify low-rank solutions even when the number of observations is small. By contrast, NGD in the same problem works similarly to EGD in the naive parametrization and fails to generalize completely. *Right:* 2 (orange) and 3 (green) layer models were initialized by collapsing randomly initialized deeper models to test the effect of initialization separately from the effect of EGD dynamics. Initialization plays a negligible role in the inductive bias of EGD in deep matrix factorization.

were set to 1, the rest were 0. Methods with explicit or implicit regularization towards sparse solutions should enjoy good generalization even when $N < D$. Confirming our expectations, we observed that EGD in diagonal parametrizations ($L = 2$, $L = 3$) performed best on this task. The deeper diagonal model ($L = 4$) was on par with the shallow solution, we expect that our 2 million EGD steps were simply not long enough for the implicit regularization to kick in (Moroshko et al., 2020). The NGD solution on the other hand completely fails to generalize when $N < D$ and does relatively poorly even as $N > D$. The middle panel of Figure 4 validates the predictions of Theorem 4: logits from the model converge to $t\mathbf{y}$. Finally, the right-hand panels of Figure 4 show that NGD was unable to identify the sparse structure, which the diagonal model infers best, and even the shallow model approximately finds.

4 Matrix Completion

As we have seen in Section 2.2, EGD in the deep matrix product parametrization $\boldsymbol{\beta} = W_1 \cdots W_L$ converges to low-rank solutions. However, when $L = 1$, i.e. when we run EGD directly on $\boldsymbol{\beta}$, the solution we find is trivial: entries of $\boldsymbol{\beta}$ where we have observation will converge to the observed value, while other entries won't move. Due to parameter-invariance, NGD cannot differentiate between parametrizations of different depth, it is natural to expect that it will fail the same way as EGD does when $L = 1$. Let's look at NGD in matrix completion.

In matrix completion we minimize the squared reconstruction error, which corresponds to the log loss in an isotropic Gaussian observation model with $\boldsymbol{\beta}$ as mean. In a Gaussian model, the Fisher Information Matrix of $\boldsymbol{\beta}$ becomes $F(\boldsymbol{\beta}) = \frac{1}{\sigma_n^2} I$, where σ_n^2 is the observation noise. The observation noise σ_n^2 is assumed a constant, and is inconsequential here as it cancels with the $\frac{1}{\sigma_n^2}$ term in the log loss. Consequently, without loss of generality, we can consider $F(\boldsymbol{\beta})$ the identity and from this it follows that NGF in any parametrization is equivalent to EGF in the direct parametrization ($\boldsymbol{\beta} = \mathbf{w}$). This implies that NGF will completely fail to generalize, i. e. make an accurate prediction of any unobserved entry of the matrix.

Figure 3 illustrates the key property of the dynamics which allows EGD to generalize in deeper parametrizations. Each panel shows values of the neural tangent kernel (NTK) (Jacot et al., 2018), its equivalent object for NGF called the natural NTK (Rudner et al., 2019), or their discretized versions. For matrix factorization the NTK $\boldsymbol{K}(\theta)$ is a $(D \times D) \times (D \times D)$ tensor which depends on the parameters θ where $k_i, j, k, l(\theta)$ measures how much the entry $\beta_{i,j}$ moves in reaction to a negative loss gradient w.r.t. $\beta_{k,l}$. In these visualizations, we set $D = 11$, and we plot the heatmap of $k_{i,j,5,5}$. We can see that when we parametrize $\boldsymbol{\beta}$ directly, the NTK is simply the identity, only the entry $\beta_{5,5}$ moves. However, when $L = 2$, EGD can now respond to the gradient signal at $\beta_{5,5}$ by moving entries in the fifth row of W_1 or in the fifth column of W_2. This, in turn, might result in moving $\beta_{i,5}$ or $\beta_{5,i}$ as well. This explains the cross pattern seen in Figure 3 first panel in the second

row. This non-identity NTK is what allows generalization to happen as 'information flows' from observations too unobserved entries of $\boldsymbol{\beta}$. However, in NGF, the natural NTK remains the identity irrespective of parametrization. This is true even in the approximately invariant NGD.

For our Matrix Factorization experiments we had to develope a scalable and numerically stable algorithm for computing the natural gradient. We did this by extending the algorithm of Bernacchia et al. (2018) to matrix factorization. Exploiting the structure of the Jacobian in the deep matrix product parametrization ($\boldsymbol{\beta} = W_1 \cdots W_L$) we calculate the natural gradient w.r.t. W_l as $\tilde{\nabla}_{W_l}\mathcal{L} = \frac{1}{L} B_l^{\top +} \tilde{\nabla}_{\boldsymbol{\beta}} \mathcal{L} A_l^{+}$, where $A_l = \prod_{i=1}^{l-1} W_i$ and $B_l = \prod_{i=l+1}^{L} W_i$. We note that A_i and B_i are matrices that are readily computed during the forward and backward pass of reverse-mode automatic differentiation of the loss. The details of the derivation can be found in Appendix B.4.

Using this algorithm, in Figure 5 we experimentally verify that NGD finds a trivial optimum in deep matrix product parametrizations of varying depth. We follow the experimental setup of Arora et al. (2019) and reproduce their results for EGD. We performed an extensive grid search of hyper-parameters and found no setting where NGD would achieve non-trivial performance.

5 Summary and Discussion

Inductive biases of gradient-based learning are driven to a large extent by the way we parametrize our hypothesis. Natural gradient descent (NGD), on the other hand, ignores the parametrization and implicitly optimizes over the manifold of hypothesis. This invited the question whether NGD exhibits any of the useful implicit regularization that EGD has been shown to have. We characterized the behaviour of NGD over logistic loss, and found that in the overparametrized regime, NGD converges to the ordinary least squares interpolant of training labels. This is in contrast with the large-margin-type behaviour EGD exhibits. In experiments we found that in the models we studied, NGD fails to generalize as well as EGD with the right parametrization.

5.1 Other related work

Approximate NGD algorithms: Since exact NGD is computationally prohibitive, a great deal of research has been devoted to developing approximate NGD algorithms for deep leaning: K-FAC Grosse & Martens (2016); Martens & Grosse (2015) exploits the approximately Kronecker structure of the Fisher information matrix, while, while Bernacchia et al. (2018) start from exact gradient descent in linear neural networks and then apply the formula verbatim to the non-linear case. Another line of work aims at improving the invariance properties of NGD algorithms bringing them closer to ideal of NGF (Song et al., 2018; Luk & Grosse, 2018). Our motivation differs in that are not focused on designing better NGD algorithms, instead we raise the question whether closer approximation of NGF is desirable in the first place. In order to perform experiments that validate our findings we develop efficient exact natural gradient descent algorithms in overparametrized linear models extending the work of Bernacchia et al. (2018).

Convergence Rates for NGD: The main reason for using NGD in deep learning is the intuitive notion it might speed up convergence by virtue of being invariant to parametrization (Amari, 1997; Pascanu & Bengio, 2013; Martens, 2014). This intuition is backed up by theory: ichi Amari (1998) proved fast convergence on a quadratic loss; Bernacchia et al. (2018) proved fast convergence for deep linear models under quadratic loss; more recently, Zhang et al. (2021) gave a proof of fast convergence which holds for a broad class of overparametrized networks and also extends to K-FAC; Rudner et al. (2019) analysed NGD in the neural tangent kernel (NTK) regime. Our work differs in that our primary interest is not whether NGD converges fast, but to better understand and illustrate possible trade-offs between fast convergence and generalization.

Generalization of NGD: Wilson et al. (2017) were the first to propose that faster convergence may come at the cost of diminished generalization performance in deep learning. Much like our work, Wilson et al. (2017) provided illustrative examples where different methods reach qualitatively different solutions. They focused on adaptive learning rate algorithms like Adam, but due to the connections between Adam and the empirical Fisher information, one might speculate that their findings would extend to NGD as well. Our work extends on this question by studying NGD, and by focusing on deep linear models we are able to provide theorems to support our observations. Zhang et al. (2021) argued against the notion that NGD may not generalize well, and supported their

argument with a generalization bound which holds for both NGD and EGD. However, generalization bounds often fail to predict the empirically observed performance of deep learning (Jiang et al., 2019, see e. g.). We therefore argue that a more qualitative understanding of the role of parametrization-independence is needed to paint a fuller picture of generalization in deep learning.

5.2 Q&A

Q: How about stochastic gradient descent? Following Gunasekar et al. (2017; 2018); Arora et al. (2019) we analysed only full-batch gradient descent. This allowed us to prove properties of gradient flow, i. e. in the limit of infinitesimally small learning rates, which is not a meaningful limit in SGD. This line of work demonstrates that useful inductive biases exist in gradient-based learning even in the absence of gradient noise. Indeed, recent empirical evidence suggests that stochasticity may not be necessary for good generalization in deep networks (see e. g. Geiping et al., 2021). In practice, we expect the question of generalization to be a complex one, with multiple factors like stochasticity or parametrization-dependence playing a role. We propose that analysing NGD is a useful tool in understanding this complex interplay, as it acts as a form of ablation by eliminating parametrization-dependence.

Q: Does this mean NGD does not generalize well? Not necessarily. We show that there are cases where it does not, but it is entirely possible that there are other situations where the inductive biases of NGD are more helpful than those of EGD in a specific parametrization, especially when large amounts of data are available.

Q: Does initialization play a role? Changing the parametrization effects gradient-based algorithms in at least two ways: through training dynamics and through initialization. As weights are often initialized from a distribution that's aligned with the coordinate-system, e. g. isotropic Gaussians, these imply a parameter-dependent initial distribution in hypothesis-space. For example, if matrices $W_1, \ldots, W_L$ are drawn from an isotropic Gaussian, their product will be effectively low-rank with a probability that increases as L increases. Thus, it is conceivable that initialization plays a role in implicit regularization towards low-rank. To test this, we ran additional experiments where we controlled the effect of initialization. We trained deep matrix factorization models where we initialized each component matrix W_1 to be a product of independent Gaussian matrices. Thus, we were able to create models that behave like a $L = 6$ layer model at initialization but $L = 2$ layer model during training. We found that the effect of initialization was negligible on generalization performance (Figure 5.c).

Q: What if you calculate Fisher information on test data? Pascanu & Bengio (2013) noted that in deep learning, averaging the Fisher information over test data, rather than training data seemingly improves performance. In our theorems and experiments we always averaged over the training data. It is an interesting question whether averaging on test data would change the inductive biases of NGD especially towards better generalization. Intuitively, we don't expect this to be the case, especially in the overparametrized $D < N$ case. It helps if we think of NGD as a trust region method. The role of preconditioning with the Fisher information matrix is to reduce movement along directions where small changes to parameters result in large changes to the hypothesis, and vice versa. Averaging the Fisher information matrix over test data might therefore result in test predictions remaining close to what they were at initialization, which would be undesirable. Nevertheless, this is a direction that invites further analysis and experiments.

Q: What about other forms of natural gradients? In addition to the *Fisher-Rao* natural gradients that we consider here, there are other forms of natural gradients, such as those based on the Wasserstein metric Li & Montufar (2018); Arbel et al. (2019). When considering this broader family of natural gradient descent, it is natural to ask if the choice of metric may give rise to different inductive biases in NGD similarly to how different parametrizations effect EGD differently. We think this is a fertile area for future research.

Reproducibility Statement

Python code to reproduce our results (including all Figures except Figure 1) can be found in the following (anonymized) git repository which contains unit tests and documentation:
`https://anonymous.4open.science/r/deeplinear-2F10`

REFERENCES

Shun-Ichi Amari. Neural learning in structured parameter spaces - natural riemannian gradient. In M. C. Mozer, M. Jordan, and T. Petsche (eds.), *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1997. URL https://proceedings.neurips.cc/paper/1996/file/39e4973ba3321b80f37d9b55f63ed8b8-Paper.pdf.

Michael Arbel, Arthur Gretton, Wuchen Li, and Guido Montufar. Kernelized wasserstein natural gradient. 10 2019. URL https://arxiv.org/abs/1910.09652v4.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 5 2019. URL http://arxiv.org/abs/1905.13655.

Alberto Bernacchia, Máté Lengyel, and Guillaume Hennequin. Exact natural gradient in deep linear networks and its application to the nonlinear case. *Advances in Neural Information Processing Systems*, 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/7f018eb7b301a66658931cb8a93fd6e8-Abstract.html.

Donald W. Fausett and Charles T. Fulton. Large Least Squares Problems Involving Kronecker Products. *SIAM Journal on Matrix Analysis and Applications*, 15(1), 1994. ISSN 0895-4798. doi: 10.1137/s0895479891222106.

Jonas Geiping, Micah Goldblum, Phillip E. Pope, Michael Moeller, and Tom Goldstein. Stochastic training is not necessary for generalization. 9 2021. URL https://arxiv.org/abs/2109.14119v1.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. *33rd International Conference on Machine Learning, ICML 2016*, 2:851–874, 2 2016. URL https://arxiv.org/abs/1602.01407v2.

Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization. *Advances in Neural Information Processing Systems*, 2017-December:6152–6160, 5 2017. URL https://arxiv.org/abs/1705.09280v1.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks. *Advances in Neural Information Processing Systems*, 2018-December:9461–9471, 6 2018. URL http://arxiv.org/abs/1806.00468.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 12 2015. URL https://arxiv.org/abs/1512.03385v1.

Daniel Hsu, Vidya Muthukumar, and Ji Xu. On the proliferation of support vectors in high dimensions. 9 2020. URL https://arxiv.org/abs/2009.10670v1.

Shun ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 2 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL http://direct.mit.edu/neco/article-pdf/10/2/251/813415/089976698300017746.pdf.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 2018-December:8571–8580, 6 2018. URL https://arxiv.org/abs/1806.07572v4.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2019.

Kwangmoo Koh, Seung-Jean Kim, Stephen Boyd, and Yi Lin. An interior-point method for large-scale 1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

Wuchen Li and Guido Montufar. Natural gradient via optimal transport. *Information Geometry*, 1: 181–214, 3 2018. URL https://arxiv.org/abs/1803.07033v5.

Kevin Luk and Roger Grosse. A coordinate-free construction of scalable natural gradient. 2018. URL https://arxiv.org/abs/1808.10340v1.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21:1–76, 12 2014. URL https://arxiv.org/abs/1412.1193v11.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. *32nd International Conference on Machine Learning, ICML 2015*, 3:2398–2407, 3 2015. URL https://arxiv.org/abs/1503.05671v7.

Edward Moroshko, Suriya Gunasekar, Blake Woodworth, Jason D. Lee, Nathan Srebro, and Daniel Soudry. Implicit bias in deep linear classification: Initialization scale vs training accuracy. *Advances in Neural Information Processing Systems*, 2020-December, 7 2020. URL https://arxiv.org/abs/2007.06738v1.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 1 2013. URL https://arxiv.org/abs/1301.3584v7.

Tim GJ Rudner, Florian Wenzel, Yee Whye Teh, and Yarin Gal. The natural neural tangent kernel: Neural network training dynamics under natural gradient descent. In *4th workshop on Bayesian Deep Learning (NeurIPS 2019)*, 2019.

Yang Song, Jiaming Song, and Stefano Ermon. Accelerating natural gradient with higher-order invariance. *35th International Conference on Machine Learning, ICML 2018*, 11:7491–7514, 3 2018. URL https://arxiv.org/abs/1803.01273v2.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 19:1–57, 10 2017. URL https://arxiv.org/abs/1710.10345v4.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58:267–288, 1 1996. ISSN 2517-6161. doi: 10.1111/J.2517-6161.1996.TB02080.X. URL https://onlinelibrary.wiley.com/doi/full/10.1111/j.2517-6161.1996.tb02080.x.

Guillermo Valle-Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *7th International Conference on Learning Representations, ICLR 2019*, 5 2018. URL https://arxiv.org/abs/1805.08522v5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December:5999–6009, 6 2017. URL https://arxiv.org/abs/1706.03762v5.

Tomas Vaškevičius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. *Advances in Neural Information Processing Systems*, 32, 9 2019. URL https://arxiv.org/abs/1909.05122v1.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in Neural Information Processing Systems*, 2017-December:4149–4159, 5 2017. URL https://arxiv.org/abs/1705.08292v2.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64:107–115, 3 2021. ISSN 15577317. doi: 10.1145/3446776. URL https://dl.acm.org/doi/abs/10.1145/3446776.

A USEFUL LEMMAS

We will need the following lemma in the proof of Theorem 1,4.

Lemma 1. *If we solve a separable classification problem with natural gradient flow with separator $\boldsymbol{\beta}$ and output $\mathbf{s}$, then the gradient and the Fisher information matrix are the following (in case of linear network this means $\mathbf{s} = X\boldsymbol{\beta}$):*

$$[\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})]_i = -y_i(1-\phi(y_i\mathbf{s}_i)) \tag{10}$$

$$[F(\mathbf{s})]_{i,j} = \delta_{i,j}\phi(\mathbf{s}_i)(1-\phi(\mathbf{s}_i) \tag{11}$$

Proof. First note that $\phi(u) = \frac{1}{1+e^{-u}}$ and $\phi(-u) = 1-\phi(u) = \frac{e^{-u}}{1+e^{-u}}$.

$$[\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})]_i = \frac{\partial \mathcal{L}}{\partial s_i} = \frac{\partial}{\partial s_i}\sum_{n=1}^{N}\log(1+e^{-y_n s_n}) = \frac{-y_i e^{-y_i s_i}}{1+e^{-y_i s_i}} = -y_i(1-\phi(y_i s_i)) \tag{12}$$

Using Equation (12) we get the following:

$$\begin{aligned}
[F(\mathbf{s})]_{i,j} &= [\mathbb{E}_{\mathbf{y}}[\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})\nabla_{\mathbf{s}}^{\top}\mathcal{L}(\mathbf{s})]]_{i,j} = [\mathbb{E}_{\mathbf{y}}[y_i(1-\phi(y_i s_i))y_j(1-\phi(y_j s_j))]]_{i,j} = \\
&= \begin{cases} \mathbb{E}_{\mathbf{y}}[(1-\phi(y_i s_i))^2] & if \quad i=j \\ \mathbb{E}_{y_i}[y_i(1-\phi(y_i s_i))]\mathbb{E}_{y_j}[y_j(1-\phi(y_j s_j))] & if \quad i\neq j \end{cases} = \\
&= \begin{cases} \phi(s_i)(1-\phi(s_i)) & if \quad i=j \\ \mathbb{E}_{y_i}[y_i(1-\phi(y_i))]\mathbb{E}_{y_j}[y_j(1-\phi(y_j))] & if \quad i\neq j \end{cases}
\end{aligned} \tag{13}$$

Now we get the following:

$$\mathbb{E}_{y_i}[y_i(1-\phi(y_i s_i))] = \phi(s_i)(1-\phi(s_i)) - (1-\phi(s_i))(1-\phi(-s_i)) = 0 \tag{14}$$

Hence we get:

$$[F(\mathbf{s})]_{i,j} = \delta_{i,j}\phi(s_i)(1-\phi(s_i)). \tag{15}$$

□

The next lemma is essential in all computation connected to matrix completion with matrix factorization.

Lemma 2. *If we assume that the product matrix $\boldsymbol{\beta}$ comes from a Gaussian distribution with fixed $\sigma_n I$ standard deviation and μ mean, then the Fisher information matrix of the product matrix in matrix factorization is $F(\boldsymbol{\beta}) = \frac{1}{\sigma_n^2}I$.*

Proof. Because of the assumption:

$$p(X|\theta) = \mathcal{N}(X|\mu, \sigma_n I), \tag{16}$$

where θ is the parameters of the model (μ, σ_n).

$$\nabla_\theta \log p(X|\theta) = \nabla_\theta \log\left(\frac{1}{\sigma_n\sqrt{2\pi}}e^{-\frac{(X-\mu)^2}{2\sigma_n^2}}\right) = \nabla_\mu\left(\log\left(\frac{1}{\sigma_n\sqrt{2\pi}}\right) - \frac{(X-\mu)^2}{2\sigma_n^2}\right) = \frac{(X-\mu)}{\sigma_n^2}, \tag{17}$$

therefore we can compute the Fisher as

$$F(\boldsymbol{\beta}) = \mathbb{E}_{X\sim p(X|\theta)}\left[\nabla_\theta \log p(X|\theta)\nabla_\theta^{\top}\log p(X|\theta)\right] = \mathbb{E}_{X\sim p(X|\theta)}\left[\frac{(X-\mu)(X-\mu)^{\top}}{\sigma_n^4}\right] = \frac{\sigma_n^2 I}{\sigma_n^4} = \frac{1}{\sigma_n^2}I \tag{18}$$

□

B EXACT NATURAL GRADIENTS IN LINEAR MODELS

B.1 SIMPLE LINEAR MODEL LOGISTIC LOSS

To obtain the natural gradient $\tilde{\nabla}_{\boldsymbol{\beta}}\mathcal{L}$ with respect to $\boldsymbol{\beta}$, we have to solve the following linear system:

$$F(\boldsymbol{\beta})\tilde{\nabla}_{\boldsymbol{\beta}}\mathcal{L} = \nabla_{\boldsymbol{\beta}}\mathcal{L}, \tag{19}$$

where $F(\boldsymbol{\beta})$ is the Fisher information matrix and $\nabla_{\boldsymbol{\beta}}\mathcal{L}$ is the (Euclidean) gradient. Under the logistic loss the Fisher information matrix becomes

$$F(\boldsymbol{\beta}) = X^\top \operatorname{diag}[\phi(X\boldsymbol{\beta}) \odot \phi(-X\boldsymbol{\beta})]X, \tag{20}$$

where ϕ is the logistic sigmoid which is applied elementwise to vector arguments and $\odot$ denotes elementwise product. The gradient of the logistic loss is as follows:

$$\nabla_{\boldsymbol{\beta}}\mathcal{L} = -(y \odot X)^\top \phi(-(y \odot X)\beta) \tag{21}$$

Mathematically, we could use these expressions and solve the linear system Equation (19), however, this would be potentially numerically unstable for reasons outlined below. Let's introduce the notation $\tilde{X} = y \odot X$ and $u = \tilde{X}\boldsymbol{\beta}$ to simplify the formulæ. Due to symmetry, in the Fisher information all occurrences of X can be replaced by $\tilde{X}$. This gives rise to the following expressions for the Fisher information matrix:

$$F(\boldsymbol{\beta}) = \tilde{X}^T \operatorname{diag}[\phi(u) \odot \phi(-u)]\tilde{X} \tag{22}$$

and the gradient:

$$\nabla_{\boldsymbol{\beta}}\mathcal{L} = -\tilde{X}\phi(-u). \tag{23}$$

As the classifier gets better, components of u increase and diverges to $+\infty$. As a consequence both $F(\boldsymbol{\beta})$ and $\nabla_{\boldsymbol{\beta}}\mathcal{L}$ are expected to become small, from the term $\phi(-u)$. This could lead to issues with numerical stability. To solve this, we rewrite both using following identity:

$$\phi(-u) = \frac{1}{1+e^u} = \frac{e^{-u}}{1+e^{-u}} = e^{-u}\phi(u) \tag{24}$$

obtaining:

$$F(\boldsymbol{\beta}) = e^{-u_{max}}\tilde{X}^T \operatorname{diag}[e^{-u+u_{max}}\phi^2(u)]\tilde{X} \tag{25}$$

$$\nabla_{\boldsymbol{\beta}}\mathcal{L} = -e^{-u_{max}}\tilde{X}e^{-u+u_{max}}\phi(u), \tag{26}$$

where u_{max} is the largest entry of u. We have thus isolated the term responsible for poor numerical performance into a multiplicative term $e^{-u_{max}}$ which we can simply leave out when solving the linear system. The remaining terms are well-behaved even as u increases, provided that the difference between elements of u is not too large.

B.2 DIAGONAL LINEAR NETWORK UNDER LOGISTIC LOSS

In a diagonal linear network we express $\boldsymbol{\beta} = \mathbf{w}_1 \odot \cdots \odot \mathbf{w}_L$. Here we will discuss how we compute the natural gradient with respect to $\mathbf{w_1}$.

We now solve the following (underdetermined) system of linear equations, which we write using using Einstein summation notation:

$$F(\boldsymbol{\beta})_{i,j}J_{j,l,k}\tilde{\nabla}_{\mathbf{w}_{l,k}}\mathcal{L} = \nabla_{\boldsymbol{\beta}_i}\mathcal{L}, \tag{27}$$

where $J_{j,l,k} = \frac{\partial \boldsymbol{\beta}_j}{\partial \mathbf{w}_{l,k}}$ is the Jacobian of the mapping from $\mathbf{w}$ to $\boldsymbol{\beta}$. In this specific parametrization, most entries of J is non-zero. Let's denote the product of the first $l-1$ weight vectors as $\mathbf{a}_l$ and the product of the last $L-l-1$ weight vectors as $\mathbf{b}_l$ so we can have:

$$\boldsymbol{\beta}_i = \underbrace{\mathbf{w}_{1,i}\cdots\mathbf{w}_{l-1,i}}_{\mathbf{a}_{l,i}}\mathbf{w}_{l,i}\underbrace{\mathbf{w}_{l+1,i}\cdots\mathbf{w}_{L,i}}_{\mathbf{b}_{l,i}} = \mathbf{a}_{l,i}\mathbf{w}_{l,i}\mathbf{b}_{l,i}. \tag{28}$$

Thus, the Jacobian becomes:

$$J_{i,l,k} = \begin{cases} \mathbf{a}_{l,i}\mathbf{b}_{l,i} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{29}$$

Substituting this back, we have to solve the following system of equations:

$$F(\boldsymbol{\beta})_{i,j} J_{j,l,k} \tilde{\nabla}_{w_{l,k}} \mathcal{L} = \nabla_{\boldsymbol{\beta}_i} \mathcal{L} \tag{30}$$

$$F(\boldsymbol{\beta})_{i,j} \mathbf{a}_{l,j} \mathbf{b}_{l,j} \tilde{\nabla}_{w_{l,j}} \mathcal{L} = \nabla_{\boldsymbol{\beta}_i} \mathcal{L}. \tag{31}$$

$$\tag{32}$$

To ensure numerical stability, we use the same trick as in B.2.

Since the above system of equations is underdetermined, we could choose different solutions. In our experiments we used the `pytorch.linalg.lstsq` least squares solver which finds the solution with the lowest ℓ_2 norm.

B.3 Separable classification

First note that in our model ($\forall n \in \{1, 2, \cdots N\}$, $\phi(s) = \frac{1}{1+e^{-s}}$)

$$\begin{aligned} p(y_n = 1 | \mathbf{x}_n, \boldsymbol{\beta}) &= \frac{1}{1 + e^{-y_n \mathbf{x}_n^\top \boldsymbol{\beta}}} = \phi(-y_n \mathbf{x}_n^\top \boldsymbol{\beta}) \\ p(y_n = -1 | \mathbf{x}_n, \boldsymbol{\beta}) &= 1 - \frac{1}{1 + e^{-y_n \mathbf{x}_n^\top \boldsymbol{\beta}}} = 1 - \phi(-y_n \mathbf{x}_n^\top \boldsymbol{\beta}) \end{aligned} \tag{33}$$

So the loss function is ($\forall n \in \{1, 2, \cdots N\}$)

$$\ell(y_n \mathbf{x}_n^\top \boldsymbol{\beta}) = \log(1 + e^{-y_n \mathbf{x}_n^\top \boldsymbol{\beta}}) \tag{34}$$

and

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{n=1}^{N} \log(1 + e^{-y_n \mathbf{x}_n^\top \boldsymbol{\beta}}) \tag{35}$$

Until know this did not depend on the parametrization. Now look at the parametrizations we used in our article.
If we use a **fully connected network** the gradient is the following:

$$\begin{aligned} \nabla_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) &= \sum_{n=1}^{N} \nabla_{\boldsymbol{\beta}} \log(1 + e^{-y_n \mathbf{x}_n^\top \boldsymbol{\beta}}) = \sum_{n=1}^{N} \frac{-y_n \mathbf{x}_n e^{-y_n \mathbf{x}_n^\top \boldsymbol{\beta}}}{1 + e^{-y_n \mathbf{x}_n^\top \boldsymbol{\beta}}} = \\ &= \sum_{n=1}^{N} \frac{-y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{x}_n^\top \boldsymbol{\beta}}} = \sum_{n=1}^{N} -y_n \mathbf{x}_n (1 - \phi(y_n \mathbf{x}_n^\top \boldsymbol{\beta})). \end{aligned} \tag{36}$$

The Fisher information matrix is the following:

$$\begin{aligned} F(\boldsymbol{\beta}) &= \mathbb{E}_X[\mathbb{E}_{Y|X}[\nabla_{\boldsymbol{\beta}} \ell(-y_n \mathbf{x}_n^\top \boldsymbol{\beta}) \nabla_{\boldsymbol{\beta}}^\top \ell(-y_n \mathbf{x}_n^\top \boldsymbol{\beta})]] = \\ &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{Y|X}[\mathbf{x}_n \mathbf{x}_n^\top (1 - \phi(y_n \mathbf{x}_n^\top \boldsymbol{\beta}))^2] = \\ &= \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top (\phi(\mathbf{x}_n^\top \boldsymbol{\beta})(1 - \phi(\mathbf{x}_n^\top \boldsymbol{\beta}))^2 + (1 - \phi(x_n^\top \boldsymbol{\beta}))(1 - \phi(-\mathbf{x}_n^\top \boldsymbol{\beta}))^2) = \\ &= \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top (\phi(\mathbf{x}_n^\top \boldsymbol{\beta})(1 - \phi(\mathbf{x}_n^\top \boldsymbol{\beta}))^2 + (1 - \phi(x_n^\top \boldsymbol{\beta}))\phi^2(\mathbf{x}_n^\top \boldsymbol{\beta})) = \\ &= \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top \phi(\mathbf{x}_n^\top \boldsymbol{\beta})(1 - \phi(\mathbf{x}_n^\top \boldsymbol{\beta})) \end{aligned} \tag{37}$$

If we use a **diagonal network** $\boldsymbol{\beta} = \mathbf{w}_1 \odot \mathbf{w}_2 \odot \cdots \odot \mathbf{w}_{L-1} \odot \mathbf{w}_L$, where $\mathbf{w} = \begin{pmatrix} \mathbf{w}_1^\top & \mathbf{w}_2^\top & \cdots & \mathbf{w}_L^\top \end{pmatrix}^\top$. The gradient is the following:

$$\nabla_{\boldsymbol{\beta}} \mathcal{L} = J^\top \nabla_{\mathbf{w}} \mathcal{L} \tag{38}$$

where J (the Jacobian) is the following

$$J = \begin{pmatrix} \frac{\partial \boldsymbol{\beta}}{\partial \mathbf{w}_1} & \cdots & \frac{\partial \boldsymbol{\beta}}{\partial \mathbf{w}_L} \end{pmatrix}. \tag{39}$$

where

$$\left[\frac{\partial \boldsymbol{\beta}}{\partial \mathbf{w}_n}\right]_{i,j} = \frac{\partial \boldsymbol{\beta}_i}{\partial [\mathbf{w}_n]_j} = \delta_{i,j} \prod_{k=1, k\neq i}^{N} [\mathbf{w}_k]_i \tag{40}$$

The Fisher information matrix is the following:

$$F(\mathbf{w}) = J^\top F(\boldsymbol{\beta}) J \tag{41}$$

B.4 Matrix factorization

Before we compute the natural gradient of matrix factorization let us introduce some notations: $\boldsymbol{\beta} = W_1 W_2 \dots W_L$, as before and

$$\theta = vec(\boldsymbol{\beta}), \tag{42}$$

$$\mathbf{w} = vec(W_1, W_2, \dots, W_L), \tag{43}$$

where *vec* vectorizes the matrices to obtain a column vector. θ is a reparametrization of $\mathbf{w}$, so $\theta = \mathcal{P}(\mathbf{w})$ and let $J = \frac{\partial \theta}{\partial \mathbf{w}}$. With this notation, let's compute the natural gradient with respect to the parametrization $\mathbf{w}$.

$$\tilde{\nabla}_{\mathbf{w}} \mathcal{L} = F(\mathbf{w})^{-1} \nabla_{\mathbf{w}} \mathcal{L} = (J^\top F(\theta)) J)^{-1} (J^\top \nabla_\theta \mathcal{L}) = J^{-1} F(\theta)^{-1} \nabla_\theta \mathcal{L} \tag{44}$$

We use the assumption that J is full rank and because of $F(\theta) = I$ is invertible $(J^\top F(\theta)) J)^{-1} = J^{-1} F(\theta)^{-1} J^{-\top}$. Thus, the natural gradient simplifies to

$$\tilde{\nabla}_{\mathbf{w}} \mathcal{L} = J^{-1} \nabla_\theta \mathcal{L} \tag{45}$$

and multiplying by J we obtain

$$J \tilde{\nabla}_{\mathbf{w}} \mathcal{L} = \nabla_\theta \mathcal{L}. \tag{46}$$

We can consider the Jacobian like L consecutive matrices

$$J = [J_1 J_2 \dots J_L] \tag{47}$$

where $J_i = \frac{\partial \theta}{\partial vec(W_i)}$, and note that $\nabla_\theta \mathcal{L} = vec(\nabla_{\boldsymbol{\beta}} \mathcal{L})$ and $\tilde{\nabla}_{\mathbf{w}} \mathcal{L} = vec(\tilde{\nabla}_{W_1, W_2, \dots W_L} \mathcal{L})$. Rewrite equation 46:

$$J vec(\tilde{\nabla}_{W_1, W_2, \dots W_L} \mathcal{L}) = vec(\nabla_{\boldsymbol{\beta}} \mathcal{L}). \tag{48}$$

If we solve the following equation for $i = 1, \dots L$, then the concatenation of vectors $vec(\tilde{\nabla}_{W_i} \mathcal{L})$ will solve equation 48 as well.

$$J_i vec(\tilde{\nabla}_{W_i} \mathcal{L}) = \frac{1}{L} vec(\nabla_{\boldsymbol{\beta}} \mathcal{L}) \tag{49}$$

Let $A_i = W_1 W_2 \dots W_{i-1}$ and $B_i = W_{i+1} W_{i+2} \dots W_L$ and using $\otimes$ notation for the Kronecker product and utilize the property $vec(ABC) = (C^\top \otimes A) vec(X)$ we get

$$J_i = \frac{\partial vec(\boldsymbol{\beta})}{\partial vec(W_i)} = \frac{\partial vec(A_i W_i B_i)}{\partial vec(W_i)} = \frac{\partial (B_i^\top \otimes A_i) vec(W_i)}{\partial vec(W_i)} = B_i^\top \otimes A_i, \tag{50}$$

thus we need to solve

$$(B_i^\top \otimes A_i) vec(\tilde{\nabla}_{W_i} \mathcal{L}) = \frac{1}{L} vec(\nabla_{\boldsymbol{\beta}} \mathcal{L}) \tag{51}$$

for $vec(\tilde{\nabla}_{W_i} \mathcal{L})$. One can do this by exploiting properties of the Kronecker product and using Moore-Penrose pseudo-inverses as follows:

$$vec(\tilde{\nabla}_{W_i} \mathcal{L}) = \frac{1}{L} (B_i^{\top +} \otimes A_i^+) vec(\nabla_{\boldsymbol{\beta}} \mathcal{L}) = \frac{1}{L} (B_i^{\top +} \nabla_{\boldsymbol{\beta}} \mathcal{L} A_i^+) \tag{52}$$

We note that when A_i and B_i are near full-rank, using the pseudoinverses may not be numerically stable. Fausett & Fulton (1994) instead proposed a solution based on QR decomposition, and even discussed an approach which extends to the rank deficient case. In practice we found that this was not necessary for ours experiments. As a result, in our implementation we use the formula $\frac{1}{L} B_i^{\top +} \nabla_{\boldsymbol{\beta}} \mathcal{L} A_i^+$ to update the factor matrices with the natural gradient.

C PROOF OF THEOREMS

C.1 PROOF OF THEOREM 1

Statement. *Let's assume, that $N < D$, X is full rank and A is an invertible $D \times D$ matrix. Let $\boldsymbol{\beta}_t = \boldsymbol{\beta}_t(X, \mathbf{y})$ be the trajectory of NGF and $\boldsymbol{\beta}'_t = \boldsymbol{\beta}_t(XA^\top, \mathbf{y})$ (the trajectory of NGF on data $XA^\top$). Then $X\boldsymbol{\beta} = XA^T\boldsymbol{\beta}'$.*

Proof. Let $\mathbf{s} = X\boldsymbol{\beta}$ and $\mathbf{s}' = XA^T\boldsymbol{\beta}'$. The gradient and the Fisher information matrix are the following (the calculation can be found in Lemma 1).

$$\begin{aligned} [\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})]_i &= -y_i(1 - \phi(y_i\mathbf{s}_i)) \\ [F(\mathbf{s})]_{i,j} &= \delta_{i,j}\phi(\mathbf{s}_i)(1 - \phi(\mathbf{s}_i)) \end{aligned} \tag{53}$$

The exact same can be said about s', so s and s' are the solutions of the same differential equations, so if we use the same initialization $s_t = s'_t$. □

C.2 PROOF OF THEOREM 2

Statement. *Let $\boldsymbol{\beta}_t(X, \mathbf{y})$ be the trajectory of NGF and let A be a $D \times D$ invertible transformation. If $N \geq D$, X has full rank and we consider NGF on the transformed data $XA^\top$, then $A^\top\boldsymbol{\beta}_t(XA^\top, \mathbf{y}) = \boldsymbol{\beta}_t(X, \mathbf{y})$.*

Proof. Let $\boldsymbol{\beta}'$ be the trajectory of NGF on the transformed data:

$$\boldsymbol{\beta}'_t = \boldsymbol{\beta}_t(XA^\top, \mathbf{y}). \tag{54}$$

To run NGF on $\boldsymbol{\beta}'$ we need its Fisher information matrix. Note that the Fisher information matrix of linear models with logistic-loss is

$$F(\boldsymbol{\beta}) = X^\top \text{diag}[\phi(X\boldsymbol{\beta}) \odot \phi(-X\boldsymbol{\beta})]X. \tag{55}$$

by Appendix B.1. Note that in this case the rank of the Fisher information matrix is D, so it is invertible. Same is true for $\boldsymbol{\beta}'$. Let's compute the Fisher information matrix of $\boldsymbol{\beta}'$.

$$F(\boldsymbol{\beta}') = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{y_n|A\mathbf{x}_n}[\nabla_{\boldsymbol{\beta}'}\ell(y_n\boldsymbol{\beta}'^\top A\mathbf{x}_n)\nabla^\top_{\boldsymbol{\beta}'}\ell(y_n\boldsymbol{\beta}'^\top A\mathbf{x}_n)] \tag{56}$$

First, specify $\nabla_{\boldsymbol{\beta}'}\ell(y_n\boldsymbol{\beta}'^\top A\mathbf{x}_n)$ and use the notation $\boldsymbol{v}^\top = \boldsymbol{\beta}'^\top A$.

$$\nabla_{\boldsymbol{\beta}'}\ell(y_n\boldsymbol{\beta}'^\top A\mathbf{x}_n) = J^\top\nabla_{\boldsymbol{v}^\top}\ell(y_n\boldsymbol{v}^\top\mathbf{x}_n) \tag{57}$$

where $J = \frac{\partial v^\top}{\partial \boldsymbol{\beta}'}$.

$$J_{i,j} = \frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{\beta}'_j} = \frac{\partial \sum_{k=1}^{d}\boldsymbol{\beta}'_k A_{k,i}}{\partial \boldsymbol{\beta}'_j} = A_{j,i} \tag{58}$$

Therefore $J = A^\top \Leftrightarrow J^\top = A$ and

$$\nabla_{\boldsymbol{\beta}'}\ell(y_n\boldsymbol{\beta}'^\top A\mathbf{x}_n) = A\nabla_{\boldsymbol{v}}\ell(y_n\boldsymbol{v}^\top\mathbf{x}_n). \tag{59}$$

We now can continue the computation of the Fisher:

$$\begin{aligned} F(\boldsymbol{\beta}') &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{y_n|A\mathbf{x}_n}[A\nabla_{\boldsymbol{v}^\top}\ell(y_n\boldsymbol{v}^\top\mathbf{x}_n)\nabla^\top_{\boldsymbol{v}^\top}\ell(y_n\boldsymbol{v}^\top\mathbf{x}_n)A^\top] = \\ &= A(\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{y_n|\mathbf{x}_n}[\nabla_{\boldsymbol{v}^\top}\ell(y_n\boldsymbol{v}^\top\mathbf{x}_n)\nabla^\top_{\boldsymbol{v}^\top}\ell(y_n\boldsymbol{v}^\top\mathbf{x}_n)])A^\top = AF(\boldsymbol{v})A^\top. \end{aligned} \tag{60}$$

Note, that the Fisher of $\boldsymbol{v}$ must be invertible as well from the previous Equation. Let's see the NGF on $\boldsymbol{\beta}'$:

$$\dot{\boldsymbol{\beta}}' = -F(\boldsymbol{\beta}')^{-1}\nabla_{\boldsymbol{\beta}'}\mathcal{L}(\boldsymbol{\beta}'^\top XA^T, y) = -(AF(\boldsymbol{v})A^\top)^{-1}\sum_{n=1}^{N}\nabla_{\boldsymbol{\beta}'}\ell(y_n\boldsymbol{\beta}'^\top Ax_n) =$$
$$= -(A^\top)^{-1}F(\boldsymbol{v})^{-1}A^{-1}A\sum_{n=1}^{N}\nabla_{\boldsymbol{v}}\ell(y_n\boldsymbol{v}^\top x_n) = -(A^\top)^{-1}F(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}) \tag{61}$$

We also have the following (by the Chain Rule):

$$\dot{\boldsymbol{v}} = J\dot{\boldsymbol{\beta}}' = A^\top\dot{\boldsymbol{\beta}}' \tag{62}$$

Now from Equation (61) and (62) we get:

$$\dot{\boldsymbol{v}} = F(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}) \tag{63}$$

This is the same differential equation as the one $\boldsymbol{\beta}$ is a solution of. So if they are initialized the same way $\boldsymbol{v}_t = \boldsymbol{\beta}_t(X, \mathbf{y})$, so $\boldsymbol{\beta}_t = \boldsymbol{v}_t = A^\top\boldsymbol{\beta}'$. □

C.3 Proof of Theorem 4

Statement. *If $N < D$ and $\boldsymbol{\beta}$ is the separator. Let $\mathbf{s}$ be the output. Let the Jacobi matrix from $\boldsymbol{\beta}$ to $\mathbf{s}$ be $J = \frac{\partial \mathbf{s}}{\partial \boldsymbol{\beta}}$. If J is full rank $\mathbf{s}$ is asymptotically linear with direction vector $\mathbf{y}$.*

Proof. First let's note that by the invariance property of NGF the trajectory of $\mathbf{s}$ is defined by the trajectory of $\boldsymbol{\beta}$.

$$\dot{\mathbf{s}} = -F^{-1}(\mathbf{s})\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s}) \tag{64}$$

Let's assume $\mathbf{s}$ is 1-dimensional. In this case $\mathbf{s} = s$, $\mathbf{x}_1 = \mathbf{x}$ and $\mathbf{y} = y$ can be used since we have only one data point. To solve equation (64) we need the gradient and the Fisher information matrix which are the following (the calculation can be found in the Appendix B.2)

$$\nabla_s\mathcal{L}(s) = -y(1 - \phi(ys)) \tag{65}$$

The Fisher information matrix:

$$F(s) = \phi(s)(1 - \phi(s)) \tag{66}$$

Then Equation (64) can be written as:

$$\dot{s} = \frac{y(1 - \phi(ys))}{\phi(s)(1 - \phi(s))} \tag{67}$$

Now we rescale our data points s.t. $\tilde{x} = -x$, so $\tilde{s} = -s$ and $\tilde{y} = -y = 1$. Hence we get:

$$\frac{\partial \tilde{s}}{\partial t} = \frac{1}{\phi(\tilde{s})} \tag{68}$$

Which can be solved and the solution is

$$\log(1 + e^{\tilde{s}}) = t + c \quad \Longleftrightarrow \quad \tilde{s} = \log(e^{t+c} - 1) \tag{69}$$

By equation (69) we get the asymptotic behaviour

$$\lim_{t\to\infty}\frac{\tilde{s}}{t + c} = \lim_{t\to\infty}\frac{\log(e^{t+c} - 1)}{t + c} = \tag{70}$$
$$= \lim_{t\to\infty}\frac{e^{t+c}}{e^{t+c} - 1} = \lim_{t\to\infty}\frac{1}{1 - e^{-(t+c)}} = 1 \tag{71}$$

(From (70) to (71) we use L'Hopital Rule).
Hence we proved Theorem 4. for $N = 1$. Now let's assume, that $N > 1$. Now we write down the gradient again:

$$[\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})]_i = -y_i(1 - \phi(y_i s_i)) \tag{72}$$

And the Fisher information matrix:

$$[F(\mathbf{s})]_{i,j} = \delta_{i,j}\phi(s_i)(1-\phi(s_i)) \tag{73}$$

So now if we substitute in Equation (72) and Equation (73) to Equation (64). We can rescale, so $\tilde{y}_i = 1 \quad \forall i$ as we did in the previous case. Hence we get the following:

$$\frac{\partial \tilde{\mathbf{s}}}{\partial t} = -\begin{pmatrix} \frac{1}{\phi(\tilde{s}_1)(1-\phi(\tilde{s}_1))} & 0 & \cdots & 0 \\ 0 & \frac{1}{\phi(\tilde{s}_2)(1-\phi(\tilde{s}_2))} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\phi(\tilde{s}_N)(1-\phi(\tilde{s}_N))} \end{pmatrix} \begin{pmatrix} -(1-\phi(\tilde{s}_1)) \\ -(1-\phi(\tilde{s}_2)) \\ \vdots \\ -(1-\phi(\tilde{s}_N)) \end{pmatrix} = \begin{pmatrix} \frac{1}{\phi(\tilde{s}_1)} \\ \frac{1}{\phi(\tilde{s}_2)} \\ \vdots \\ \frac{1}{\phi(\tilde{s}_N)} \end{pmatrix} \tag{74}$$

Hence we got N independent differential equations which are exactly the same as in the $N = 1$ case. So in each dimension $\tilde{\mathbf{s}}$ is asymptotically $t + c$ for some c. Hence $\tilde{\mathbf{s}} \approx t\mathbb{1} + \boldsymbol{c}$, where $\boldsymbol{c} \in \mathbb{R}^D$ is a constant. So $\mathbf{s} \approx t\mathbf{y} + \boldsymbol{c}_s$, where $\boldsymbol{c}_s \in \mathbb{R}^D$ is a constant. □

D COUNTEREXAMPLE FOR THE INVARIANCE OF ℓ_2 LARGE MARGIN SOLUTION

The counterexample is the following:

$$A = \begin{pmatrix} 1 & 2 \\ -1 & 0 \end{pmatrix}, y = 1 \text{ and } X = (2 \quad -3)$$

Then $\boldsymbol{\beta}^*(X, y) = argmin\|\boldsymbol{\beta}\|_2$ subject to $2\boldsymbol{\beta}_1 - 3\boldsymbol{\beta}_2 \geq 1$, therefore $\boldsymbol{\beta}^*(X, y) = \begin{pmatrix} 0 \\ -\frac{1}{3} \end{pmatrix}$. Furthermore $\boldsymbol{\beta}^*(XA^\top, y) = argmin\|\boldsymbol{\beta}\|_2$ subject to $-4\boldsymbol{\beta}_1 - 2\boldsymbol{\beta}_2 \geq 1$, therefore $\boldsymbol{\beta}^*(XA^\top, y) = \begin{pmatrix} -\frac{1}{4} \\ 0 \end{pmatrix}$, but $A^\top\boldsymbol{\beta}^*(XA^\top, y) = \begin{pmatrix} -\frac{1}{4} \\ -\frac{1}{2} \end{pmatrix} \neq \begin{pmatrix} 0 \\ -\frac{1}{3} \end{pmatrix} = \boldsymbol{\beta}^*(X, y)$.

E INVARIANCE PROPERTY OF OLS

We show the same transformation invariance property for OLS that we showed in Theorem 1,2 for NGF. Again, we split the problem into two cases: $N < D$ and $N \geq D$. Note that for the problem $X\boldsymbol{\beta} = y$ the Ordinary least squares solution is $\boldsymbol{\beta} = (X^\top X)^{-1}X^\top y$ if the columns of X is linearly independent.

Statement. *Let $N < D$, A is an invertible $D \times D$ matrix. If $\boldsymbol{\beta}$ is the solution of the Ordenary least squares problem for the matrix X and $\boldsymbol{\beta}'$ for $XA^\top$, then $X\boldsymbol{\beta} = XA^\top\boldsymbol{\beta}'$.*

Proof. Immediately follows from the definition of the problems: $X\boldsymbol{\beta} = y$ and $XA^\top\boldsymbol{\beta}' = y$. □

Statement. *Let $N \geq D$, X has full rank and A is an invertible $D \times D$ matrix. If $\boldsymbol{\beta}$ is the solution of the OLS problem for the matrix X and $\boldsymbol{\beta}'$ for $XA^\top$, then $A^\top\boldsymbol{\beta}' = \boldsymbol{\beta}$.*

Proof.

$$A^\top\boldsymbol{\beta}' = A^\top((XA^\top)^\top XA^\top)^{-1}(XA^\top)^\top y = A^\top A^{\top -1}(X^\top X)^{-1}A^{-1}AX^\top y = \boldsymbol{\beta}$$

□