Towards Understanding the Role of Representation Dimension in Meta-Learning

Anonymous Author
Anonymous Institution

Abstract

Meta-learning aims to uncover the principle features of the tasks, which are often low-dimensional, from limited data available for related tasks. In this paper, we consider a setup where task features are approximately in a low dimensional subspace, and we do not know the subspace and its dimension beforehand.

A low rank approximation step is commonly used to retrieve the low dimensional space. When the tasks are only approximately low dimensional, the dimension reduction step is not necessarily optimal for generalizing to the new meta-test task.

This paper provides a theoretical comparison of more traditional low-dimensional representations, obtained by low-rank truncation, and high-dimensional representations which might actually lead to an ill-posed overparameterized problem when learning a new task from limited data. We show that learning large representations where directions are weighted by their relative importance can in fact be the right choice over small representations. Furthermore, our findings also reveal a double descent phenomena when the representation dimension coincides with the sample size. Extensive experiments corroborate the benefit of large and weighted representations over utilizing simple low-dimensional subspaces.

Preliminary work. Under review by AISTATS 2021. Do not distribute.

1 Introduction

When available training data is limited, meta-learning has proved to be a powerful technique. It can exploit the information provided by other tasks with more data to help learn the features of a new task. This idea was introduced decades ago [Caruana, 1997, Baxter, 2000] and has shown promise in modern machine learning tasks, e.g., in image classification [Deng et al., 2009], and in machine translation between thousands of English and French sentences [Bojar et al., 2014], which involve numerous tasks to be learned with limited data per task.

A well-studied classical model for multi-task problems is mixed linear regression, for which efficient algorithms and sample complexity bounds are discussed in [Zhong et al., 2016, Li and Liang, 2018, Chen et al., 2020]. Mixed linear regression can also used in representation learning: the low dimensional subspace of the features, learned by mixed linear regression, is utilized as the search space for training on new tasks. When the feature vector of a new task comes from the same low dimensional subspace, [Maurer et al., 2016] relates generalization error with the Rademacher complexity of the features, showing less data is needed for the new task. The recent papers [Kong et al., 2020b, Kong et al., 2020a] clarify the procedures of representation learning that involve dimension reduction, clustering and training. When the low-dimensional feature space is used as the search space for training, the regression step is overdetermined with restricted data, which guarantees a low generalization error.

Another approach to representation learning [Du et al., 2020, Tripuraneni et al., 2020] is to set up a nonconvex optimization problem with matrix factors of appropriate sizes, which captures the low dimensional structure. One can apply gradient descent to this nonconvex problem, and studying its behavior requires a nontrivial landscape analysis of

the matrix factorization problem.

However, the current setups for representation learning have some restrictions. First, the features need to lie exactly in a low dimensional space. If the covariance of the features is small but non-zero in some directions, the low rank approximation error can lead to generalization error. Second, it is assumed that the approximate dimension (or effective dimension, see Def 2.5) of the feature space is known apriori, which may not be true. With limited data, estimating this dimension may also be challenging.

At the high level, we ask whether we need to perform dimension reduction to retrieve the feature subspace before meta-testing, to make the regression step in the meta-testing phase overdetermined. Are there alternatives for handling approximately low rank features?

When the amount of data is between the effective dimension and the full parameter dimension, if one sets up a linear regression problem, it is overdetermined in the low-dimensional space but underdetermined in the full-sized problem. This is a motivation for using dimension reduction. However, a phenomenon termed "double-descent" has been recently observed in related settings: when the model becomes more complex, the generalization error first increases, and then drops together with the training loss. Recent literature study the role of overparameterization in deep neural networks where an overparametrized model provably generalizes well; see, for example, papers on the neural tangent kernel [Jacot et al., 2018, Arora et al., 2019, Allen-Zhu et al., 2019], and the neural ODE [Lu et al., 2020]. With the model of linear regression, the double-descent curve is studied in [Hastie et al., 2019, Montanari et al., 2019, Wu and Xu, 2020]. [Hastie et al., 2019] introduces the ridgeless linear regression problem of different dimensions, which is the most similar to ours. It quantifies the generalization error asymptotically when the amount of training data tends to infinity. Their result aligns with the double-descent theory, that the error is small either (i) when the representation dimension (where we perform dimension reduction, see Def. 2.9) is small, or (ii) when the regression problem is solved in the whole parameter space without dimension reduction.

Contributions:

In this paper, we suggest a weighting approach that allows low loss in the overparametrized regime, and point out the connection between generalization error and the approximate low-dimensionality of the features. We argue that the dimension reduction step, used in [Kong et al., 2020b], is not necessary for getting a small generalization error in the meta-testing stage. We study two cases: a model where we use a low-dimensional subspace to represent the features, and a model where the representation is full-dimensional. In the full-dimensional (overparametrized) case, when we apply a weighting scheme on the representation of data and set up a least squares problem, we obtain generalization error guarantee in Theorem 3.3 and we show the error is no worse than the $R = r$ case. We also describe a double-descent phenomenon in this setting, i.e., linear regression suffers from exploding generalization error when the representation dimension gets close to the size of meta-test data (Theorem 3.4), and the error decreases again when the representation dimension gets even larger.

Corresponding to the contributions above, Section 3 contains the main theorems. Subsection 3.1 upper bounds the expected generalization error in the overparametrized regime; Subsection 3.2 lower bounds this error in marginally overdetemined regime and Subsection 3.3 gives exact (up to constants) expected generalization error in overdetemined regime. In Section 4, we run numerical experiments on both synthetic data and MNIST data to demonstrate our training and testing algorithms, and compare the empirical generalization error with different representation dimensions. The experiments verify the theoretical behavior proposed in Section 3.

2 Problem formulation

The meta-learning algorithm consists of two phases, meta-training and meta-testing. In the meta-training phase, we adopt a mixed linear regression setup to learn the low dimensional feature space. In the meta-testing phase, we use the feature space to learn the feature of a new task.

In the first phase, there are a few tasks, each associated with its own parameters, called features. One accesses batches of data, each of whom is collected from a task, but we do not know which task it comes from. We make this setup more precise using the following definitions.

Definition 2.1 ***Task.*** *A task is associated with parameter $\beta \in \mathbb{R}^d$. For any parameter β, one can generate data $(\boldsymbol{x}, y)$ from it. Here $\boldsymbol{x} \in \mathbb{R}^d$, $y \in \mathbb{R}$. $\boldsymbol{x}$, y satisfy $y = \boldsymbol{x}^\top \beta$. We assume there are in total K tasks in the training dataset.*

Definition 2.2 ***Task distribution.*** *All the training tasks β_k for $k = 1, ..., K$ are generated i.i.d. from a Gaussian distribution $\mathcal{N}(0, \mathbf{\Sigma})$. For simplicity of notation, we assume all β_k are distinct*[1]*. We denote the set of tasks as $\mathcal{S}^{\text{task}} = \{\beta_k \mid k = 1, ..., K\}$.*

Definition 2.3 ***Meta-training data and batches.*** *There are in total n batches of data, and $n \geq K$. The j-th batch is denoted as $\mathcal{S}_j$ and let $t_j := |\mathcal{S}_j|$. Denote $\mathcal{S}_j = \{(\boldsymbol{x}_{i,j}, y_{i,j}) \mid i = 1, ..., t_j\}$ and $\mathcal{S} = \{(\boldsymbol{x}_{i,j}, y_{i,j}) \mid j = 1, ..., n,\ i = 1, ..., t_j\}$. The data from the j-th batch is generated from a task $\theta_j \in \mathbb{R}^d$, which means $y_{i,j} = \boldsymbol{x}_{i,j}^\top \theta_j$.*

We have defined the set of tasks $\mathcal{S}^{\text{task}}$ in Def. 2.2. We assume that $\theta_j \in \mathcal{S}^{\text{task}}$, i.e., $\theta_j = \beta_k$ for some number $k \in [1, K]$. However, we do not know the correspondence between j and k.

For a fixed k, β_k can correspond to multiple batches. Let b_k denote the number of batches corresponding to β_k, defined as

$$b_k = |\{j \mid \theta_j = \beta_k\}|, \ k = 1, ..., K.$$

Note that $\sum_{k=1}^K b_k = n$ since there are n batches in total.

The number of tasks is K, and the number of batches is n. We use β_k to denote the feature of the k-th task and θ_j to denote the feature corresponding to the j-th batch. To explain the relation of β_k and θ_j, we list two examples.

1. The feature vector of each batch of data is independently generated from $\mathcal{N}(0, \mathbf{\Sigma})$. In this case $k = n$, and we can define $\beta_j = \theta_j$, $j = 1, ..., n$. $b_k = 1$ for all k from 1 to K.
2. The set of task features $\mathcal{S}^{\text{task}}$ is generated independently from $\mathbf{\Sigma}$. The task corresponding to the data of the j-th batch, is randomly chosen from $\mathcal{S}^{\text{task}}$, with probability $p_1, ..., p_K$. In expectation of $p_1, ..., p_K$, we count how many times the k-th task appears among all batches of data, and it follows

$$\boldsymbol{E}_{\boldsymbol{P}(\theta_j = \beta_k) = p_k} b_k = np_k.$$

Definition 2.4 ***Data distribution.*** *All $\boldsymbol{x}_{i,j} \in \mathbb{R}^d$ are generated i.i.d. from distribution $\mathcal{N}(0, \mathbf{\Sigma}_x)$ and $y_{i,j} = \boldsymbol{x}_{i,j}^\top \theta_j$. For simplicity, we discuss $\mathbf{\Sigma}_x = I$.*

Remark 2.1 *If $\mathbf{\Sigma}_x \neq I$ and we know $\mathbf{\Sigma}_x$ when running the algorithm, we can consider*

$$\begin{aligned} y_{i,j} &= \boldsymbol{x}_{i,j}^\top \theta_j \\ &= (\mathbf{\Sigma}_x^{-1/2} \boldsymbol{x}_{i,j})^\top \mathbf{\Sigma}_x^{1/2} \theta_j. \end{aligned}$$

Then the distribution of $\mathbf{\Sigma}_x^{-1/2} \boldsymbol{x}_{i,j}$ is standard normal. We can use the same method to estimate $\mathbf{\Sigma}^{1/2} \theta_j$.

Let the eigendecomposition of $\mathbf{\Sigma}$ be $\mathbf{\Sigma} = \boldsymbol{U} \mathbf{\Lambda} \boldsymbol{U}^\top$. We study the case that $\mathbf{\Lambda}$ is spiked with size r, say $\lambda_1, ..., \lambda_r \gg \lambda_{r+1}, ..., \lambda_d$.

Definition 2.5 ***Effective dimension.*** *(Informal) When the eigenvalue of Σ satisfies $\lambda_r \gg \lambda_{r+1}$, we can use r as the effective dimension.*

The effective dimension is informally defined, as we do not quantify $\lambda_r \gg \lambda_{r+1}$ exactly *MF: we can easily do this*. We define effective dimension because when we fix any r, we will bound the expected generalization error (Def. 2.11) by the magnitude of $\lambda_{r+1}, ..., \lambda_d$. When λ_{r+1} is small, the expected generalization error is guaranteed to be small.

When the dimension of the span of features is low, [Kong et al., 2020b] performs a dimension reduction algorithm to find the low dimensional subspace that the features span. This is done by selecting the top eigenvectors of the feature covariance estimator.

Definition 2.6 ***Moment estimator of feature covariance.*** *The covariance (of all β_j) estimator is*

$$\hat{\boldsymbol{M}} = \sum_{j=1}^n \frac{2}{t_j^2} \Bigg[\Big(\sum_{i=1}^{t_j/2} y_{i,j} \boldsymbol{x}_{i,j} \Big) \Big(\sum_{i=t_j/2+1}^{t_j} y_{i,j} \boldsymbol{x}_{i,j} \Big)^\top \tag{2.1a}$$

$$+ \Big(\sum_{i=t_j/2+1}^{t_j} y_{i,j} \boldsymbol{x}_{i,j} \Big) \Big(\sum_{i=1}^{t_j/2} y_{i,j} \boldsymbol{x}_{i,j} \Big)^\top \Bigg] \tag{2.1b}$$

When $\mathbf{\Sigma}_x = I$, the mean of the subGaussian matrix random variable $\hat{\boldsymbol{M}}$ is

$$\boldsymbol{M} = \boldsymbol{E}_{\boldsymbol{x} \sim \mathcal{N}(0, \mathbf{\Sigma}_x)} (\hat{\boldsymbol{M}}) = \frac{1}{n} \sum_{k=1}^K b_k \beta_k \beta_k^\top$$

If we consider the randomness of β_k, then $\boldsymbol{E}_{\beta_1, ..., \beta_K \sim \mathcal{N}(0, \mathbf{\Sigma})} (\boldsymbol{M}) = \mathbf{\Sigma}$.

We mention two lemmas that are straightforward corollaries of [Kong et al., 2020b].

Lemma 2.7 *([Kong et al., 2020b, Lemma 5.1, A.9]) Let $\epsilon > 0$ be the estimation error, $t_j \geq \underline{t} \geq 2$ for all $j = 1, ..., n$, let $\delta \in (0, 1)$ be failure probability, $n\underline{t} \gtrsim d \max\{\epsilon^{-2}, \epsilon^{-1} \log(nd/\delta)\} \log^2(nd/\delta)$.*[2] *With probability $1 - \delta$, we have $\|\hat{\boldsymbol{M}} - \boldsymbol{M}\| \leq \epsilon d$.*

[1] That happens almost surely.

[2] The notations $\gtrsim$, $\lesssim$, $\approx$ mean comparison regardless of constant factors.

Lemma 2.8 *([Kong et al., 2020b, Lemma A.10]) Let $\epsilon > 0$, $\delta \in (0,1)$ be failure probability, $n \gtrsim \log^3(K/\delta)/\epsilon^2$. With probability $1-\delta$, we have $\|\boldsymbol{\Sigma} - \hat{\boldsymbol{M}}\| \le \epsilon d$.*

Throughout the paper, we start from the analysis for the case $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$, and generalize to the case where $\|\hat{\boldsymbol{M}} - \boldsymbol{\Sigma}\|$ is bounded by a positive number, which is justified using the lemmas above.

Eigenvalue truncation. We do the R-eigenvalue truncation to $\hat{\boldsymbol{M}}$ to get a low dimensional truncation. Let $\hat{\boldsymbol{U}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{U}}^\top$ be the eigen-decomposition of $\hat{\boldsymbol{M}}$. Denote $\hat{\lambda}_j$ as the jth eigenvalue of $\hat{\boldsymbol{\Lambda}}$. Let $\hat{\boldsymbol{U}}_R$ be the first R columns of $\hat{\boldsymbol{U}}$, and the R-eigenvalue truncation is $\hat{\boldsymbol{M}}_R = \hat{\boldsymbol{U}}_R \mathrm{diag}(\hat{\lambda}_1, ..., \hat{\lambda}_R)\hat{\boldsymbol{U}}_R^\top$. Note that eigenvalues can be negative. When $R < r$, the recovery is incorrect since a big eigenvalue is truncated, which means an important feature is missing. So we only consider $r \le R \le d$.

Definition 2.9 ***Representation dimension.*** *We call R, the rank at which we do eigenvalue truncation, as representation dimension.*

The moment estimator is utilized for learning the feature of new task, which is generated independently from a common distribution with the same approximately low rank covariance matrix.

Definition 2.10 ***Meta-test task.*** *The meta-test task corresponds to a vector β, which is generated from $\mathcal{N}(0, \boldsymbol{\Sigma})$ independently from the training tasks. There are t samples from the dataset. With $i = 1, ..., t$, $\tilde{\boldsymbol{x}}_i \in \mathbb{R}^d$ are i.i.d. standard normal and we have $\tilde{y}_i = \tilde{\boldsymbol{x}}_i^\top \beta$. We use $\mathcal{S}_{\mathrm{MT}} = \{(\tilde{\boldsymbol{x}}_i, \tilde{y}_i) \mid i = 1, ..., t\}$ to denote the mata-training dataset.*

The amount of data is not sufficient, $t \in [r, d]$. We first project $\boldsymbol{x}$ onto a R dimensional subspace. Let $\boldsymbol{A} \in \mathbb{R}^{R \times R}$, we regress the map $\tilde{y}_i = (\boldsymbol{A}\hat{\boldsymbol{U}}_R^\top \tilde{\boldsymbol{x}}_i)^\top \alpha$ to find $\alpha \in \mathbb{R}^R$. Let $\boldsymbol{X} \in \mathbb{R}^{t \times d}$ and the i-th row is $\tilde{\boldsymbol{x}}_i^\top$, $\boldsymbol{y} = [\tilde{y}_1, ..., \tilde{y}_t]^\top$, then we estimate

$$\hat{\alpha} = (\boldsymbol{X}\boldsymbol{B})^\dagger \boldsymbol{y}.$$

We study the generalization guarantee.

Definition 2.11 ***Generalization error and expected generalization error.*** *For any vector $\alpha \in \mathbb{R}^R$, we define the generalization error as*

$$\begin{aligned}\mathcal{L}(\alpha) &= \boldsymbol{E}_{\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_x), y = \boldsymbol{x}^\top \beta} (\alpha^\top \boldsymbol{A}\hat{\boldsymbol{U}}_R^\top \boldsymbol{x} - y)^2 \\ &= \boldsymbol{E}_{\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_x)} ((\hat{\boldsymbol{U}}_R \boldsymbol{A}^\top \alpha - \beta)^\top \boldsymbol{x})^2\end{aligned}$$

The expected generalization error is

$$\boldsymbol{E}_{\beta \sim \mathcal{N}(0, \boldsymbol{\Sigma})} \mathcal{L}(\alpha)$$

Note that, if α is the solution of a linear regression problem in meta-test phase, with the meta-test data dependent on β, e.g., the least squares solution $\hat{\alpha}$, then $\mathcal{L}(\hat{\alpha})$ implicitly depends on β.

Algorithm 1 Representation learning

Input: Dataset $\mathcal{S}$ with n batches, j-th batch has size t_j. Dimension d, number of tasks K. The meta-test task dataset $\mathcal{S}_{\mathrm{MT}}$ with batch size t. Representation dimension R.
Compute $\hat{\boldsymbol{M}}$ by (2.1).
R-eigenvalue truncation:
$\hat{\boldsymbol{M}}_R \leftarrow \hat{\boldsymbol{U}}_R \mathrm{diag}(\hat{\boldsymbol{\Lambda}}_{1,1}, ..., \hat{\boldsymbol{\Lambda}}_{R,R})\hat{U}b_R^\top$.
$\boldsymbol{A} \leftarrow \sqrt{\max\{0, \hat{\boldsymbol{\Lambda}}_R\}}$
$\boldsymbol{B} \leftarrow \hat{\boldsymbol{U}}_R \boldsymbol{A}^\top$, $\boldsymbol{y} \leftarrow [y_1, ..., y_t]^\top$.
$\hat{\alpha} \leftarrow (\boldsymbol{X}\boldsymbol{B})^\dagger \boldsymbol{y}$.
return $\hat{\alpha}$

The representation learning algorithm is detailed in Algorithm 1. As a summary, it involves the following steps:

1. Construct the moment estimator $\hat{\boldsymbol{M}}$ from the training data. $\hat{\boldsymbol{M}}$ estimates the covariance matrix $\boldsymbol{\Sigma}$ of the tasks β.
2. Execute dimension reduction by applying truncated eigen-decompostion to moment estimator $\hat{\boldsymbol{M}}$, with truncation level R. We will study different choices of R. Specifically, when $R = d$, we do not reduce the dimension of $\hat{\boldsymbol{M}}$.
3. Construct a weighting matrix $\boldsymbol{A}$, the shaping matrix $\boldsymbol{B}$ and the shaped data matrix $\boldsymbol{X}\boldsymbol{B}$, used in the next linear regression step.
4. Solve linear regression with $\boldsymbol{X}\boldsymbol{B}$ and $\boldsymbol{y}$.

We study the following choices of parameters and the corresponding generalization error.

1. Choice of R: $R = r$, $R = t$, $R = d$.
2. Choice of $\boldsymbol{A}$ when $R = d$: $\boldsymbol{A} = I_R$, $\boldsymbol{A} = \sqrt{\hat{\boldsymbol{\Lambda}}_R}$.

When $R \le t$, different choice of $\boldsymbol{A}$ leads to the same least squares solution (whenever $\boldsymbol{A}$ is invertible). In the overparametrized case when $R > t$, we adopt the smallest ℓ_2 norm solution, which depends on the choice of $\boldsymbol{A}$. We briefly summarize the contributions in the following section.

1. (Theorem 3.3) When $R = d$ and $\boldsymbol{A} = \sqrt{\max\{0, \hat{\boldsymbol{\Lambda}}\}}$, the generalization error can be **upper** bounded by $O(d\lambda_{r+1})$.
2. (Theorem 3.4) When $R = t$, the generalization error can be **lower** bounded by $O(t^2 d\lambda_{r+1})$.

3. (Theorem 3.7) When $R = r$, the generalization error is $O(d\lambda_{r+1})$.

Based on those theorems, we are able to argue the double descent phenomenon: When $R = d$ with weight matrix $\boldsymbol{A} = \sqrt{\hat{\boldsymbol{\Lambda}}}$, the expected generalization error for meta-testing via least squares is as small as the case $R = r$; When $R = t$ with least squares set up, the expected generalization error for meta test is big.

3 Main theorems

3.1 Overparametrized case ($R = d$).

We start from a simple case, when $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$. Denote $\alpha^* = \boldsymbol{A}^{-1}\boldsymbol{U}^\top\beta$. We have that

$$\alpha^* \sim \mathcal{N}(0, I)$$
$$(\boldsymbol{X}\boldsymbol{B})_{j,:} \sim \mathcal{N}(0, \hat{\boldsymbol{M}}), \quad \forall j = 1, ..., t$$

And $\mathcal{L}(\alpha^*) = 0$.

We warm up with the simple case $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$, which means we have an accurate information of the covariance of the features. When we say this, we mean we have large training dataset so that the moment estimator $\hat{\boldsymbol{M}}$ is very accurate. Another case is that, we know $\boldsymbol{\Sigma}$ beforehand, so that we just need to set $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$ rather than estimating the empirical covariance from the training set. Beyond that, we will extend to the case when $\hat{\boldsymbol{M}} - \boldsymbol{\Sigma}$ is bounded, corresponding to Lemma 2.7 and 2.8.

• $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$. In this and the following subsections, we will propose theorems that bounds the expected generalization error caused by the linear regression algorithm.

Theorem 3.1 *If $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$, the generalization error of Algorithm 1 satisfies with probability $1 - \exp(-\Omega(r))$,*

$$\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha}) \lesssim \frac{4\lambda_{r+1}\lambda_1 d^2}{\lambda_r t}.$$

With overparametrization, if λ_{r+1} is small, the generalization error is upper bounded by $O(\lambda_{r+1})$. As the least squares is underdetermined when $R = d > t$, we have to set up the weight importance $\boldsymbol{A} = \sqrt{\boldsymbol{\Lambda}}$ correctly, and get the least squares solution with the smallest ℓ_2 norm. If we set $\boldsymbol{A} = I$, the solution is different and causes big error. As mentioned before, in the cases $R = t$, $R = r$, different choices of $\boldsymbol{A}$ result in the same generalization error.

We sketch the proof in the main paper, and put the details in the supplement. We first quote a lemma about the random matrix concentration.

Lemma 3.2 *([Vershynin, 2010, Cor 5.35] Let $g > 0$, $Q \in \mathbb{R}^{q_1\times q_2}$, $q_1 > q_2$, the entries of Q are i.i.d. standard Gaussian. Then let σ be singular values of Q, with probability $1 - 2\exp(-g^2/2)$,*

$$\sqrt{q_1} + \sqrt{q_2} + g \geq \sigma_{\max}(Q) > \sigma_{\min}(Q) \geq \sqrt{q_1} - \sqrt{q_2} - g.$$

For our proof, note that $\hat{\alpha} = (\boldsymbol{X}\boldsymbol{B})^\dagger\boldsymbol{y}$ and it's not necessary that $\hat{\alpha} = \alpha^*$. $\hat{\alpha} = (\boldsymbol{X}\boldsymbol{B})^\dagger\boldsymbol{y}$ gives the smallest ℓ_2 norm solution to the underdetermined linear regression problem, it is guaranteed that $\|\hat{\alpha}\| \leq \|\alpha^*\|$. Let $\hat{\alpha}_r = \hat{\boldsymbol{U}}_r^\top\hat{\alpha}$, $\alpha_r^* = \hat{\boldsymbol{U}}_r^\top\alpha^*$.

We study $\Delta_r := \hat{\alpha}_r - \alpha_r^*$ and $\Delta_{r^\perp} := (\hat{\alpha} - \alpha^*) - (\hat{\alpha}_r - \alpha_r^*)$. Then we have that $\boldsymbol{X}\boldsymbol{B}\Delta_r = -\boldsymbol{X}\boldsymbol{B}\Delta_{r^\perp}$.

With the help of random matrix theory, we argue that

$$\|\boldsymbol{X}\boldsymbol{B}\Delta_r\| \geq \sqrt{\lambda_r}\|\Delta_r\|,$$
$$\|\boldsymbol{X}\boldsymbol{B}\Delta_{r^\perp}\| \leq \sqrt{\lambda_{r+1}}\|\Delta_{r^\perp}\|.$$

This means

$$\|\Delta_r\|/\|\Delta_{r^\perp}\| \ll 1.$$

And we use the fact that the expected generalization error is only sensitive to $\|\Delta_r\|$ to complete the proof.

• $\hat{\boldsymbol{M}} \neq \boldsymbol{\Sigma}$. We use $\hat{\lambda}$ to denote the diagonal terms of $\hat{\boldsymbol{\Lambda}}$. Let $\boldsymbol{\Delta} = \hat{\boldsymbol{U}}\boldsymbol{A} - \boldsymbol{U}\sqrt{\boldsymbol{\Lambda}}$, $\|\boldsymbol{\Delta}\| \leq \Delta_0$.

Theorem 3.3 *Denote $\boldsymbol{\Delta} = \hat{\boldsymbol{U}}\boldsymbol{A} - \boldsymbol{U}\sqrt{\boldsymbol{\Lambda}}$, $\|\boldsymbol{\Delta}\| \leq \Delta_0$ (whici is justified by Lemma 2.7, 2.8), the generalization error of Algorithm 1 satisfies with probability more than $1 - \exp(-\Omega(r))$*

$$\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha}) \lesssim (\Delta_0^2 d + (\frac{\hat{\lambda}_1 d}{\hat{\lambda}_r t} + 1)\hat{\lambda}_{r+1})d.$$

The probability comes from that the condition in Lemma 3.2 holds. The challenge of bounding the error caused by Δ is that, the least squares solution involves the inverse of the weighted data matrix $\boldsymbol{X}\boldsymbol{B}$. Although we know from Lemma 2.7 and 2.8 that $\hat{\boldsymbol{M}} - \boldsymbol{\Sigma}$ is bounded, the difference of their pseudo-inverse, $\hat{\boldsymbol{M}}^\dagger$ and $\boldsymbol{\Sigma}^\dagger$, can be arbitrarily big, especially when they are both approximately low rank. An instance is that, if $\lambda_1 = 100\lambda_d$ (λ is the eigenvalue of $\boldsymbol{\Sigma}$), even if $\|\hat{\boldsymbol{M}} - \boldsymbol{\Sigma}\| = 0.01\|\boldsymbol{\Sigma}\|$, $\hat{\boldsymbol{M}}$ can be singular and $\hat{\boldsymbol{M}}^\dagger$ and $\boldsymbol{\Sigma}^\dagger$ are completely different.

The proof is in the supplementary material, and we give a brief sketch of the proof. Let $s > 0$ and study the solution to ridge regression problem with ridge weight s, which is

$$\check{\alpha} = ((\boldsymbol{X}\boldsymbol{B})^\top\boldsymbol{X}\boldsymbol{B} + sI)^{-1}(\boldsymbol{X}\boldsymbol{B})^\top\boldsymbol{y}. \tag{3.1}$$

And we separate the error into two parts, one caused by Δ and the other caused by the underdetermined nature and the randomness of data. The proof of the second part is similar to that of Theorem 3.1. Finally we take the limit

$$\lim_{s\to 0} \boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\check{\alpha}). \tag{3.2}$$

3.2 Marginally overdetermined case $R = t$

The generalization error blows up when the effective dimension is same as the size of the meta-test set.

Theorem 3.4 *In Lemma 3.5, choose C_3 such that the event in $\sigma_{\min}(\mathbf{Q}) \geq C_3/\sqrt{q}$ happens with probability less than δ when $q = t$. We have that with probability $1-\delta$ (conditioned on this case being true), we have that*

$$\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha}) \gtrsim t^2 \sum_{j=t}^{d} \lambda_j \geq t^2(d-t)\lambda_d$$

Remark 3.3 *We compare Theorem 3.3 and 3.4. When $R = d$, suppose Δ_0 is small, the error is linear in d. When $R = t$ and we assume $\lambda_{r+1} = ... = \lambda_d$, and $d = 2t$, the error is cubic in d.*

Before the proof sketch of Theorem 3.4, we quote a lemma on random matrix which we used in Theorem 3.4.

Lemma 3.5 *([Rudelson and Vershynin, 2008, Thm 1.1, 1.2]) There exists constants C_1, C_2, C_3, such that for a matrix $\mathbf{Q} \in \mathbb{R}^{q\times q}$ whose entries are i.i.d. standard normal, we have*

$$\mathbf{P}(\sigma_{\min}(\mathbf{Q}) \geq C_3/\sqrt{q}) \leq (C_1/C_3)\log(C_3) + C_2^q. \tag{3.3}$$

Lemma 3.5 suggests that the square random matrix is approximately low rank, which amplifies error caused by λ_{t+1} when we solve the linear regression. With Lemma 3.5, we will prove Theorem 3.4.

The proof is placed in supplement, and we give a brief sketch here.

Let $\boldsymbol{X} = [\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_t]^\top$, Then we can write $\boldsymbol{y}$ as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{B}\alpha^* + \boldsymbol{X}(I - \hat{\boldsymbol{U}}_t\hat{\boldsymbol{U}}_t^\top)\beta$$

and $\hat{\alpha} = (\boldsymbol{X}\boldsymbol{B})^{-1}\boldsymbol{y} = \alpha^* + (\boldsymbol{X}\hat{\boldsymbol{U}}_t\boldsymbol{A}^\top)^{-1}\boldsymbol{X}(I - \hat{\boldsymbol{U}}_t\hat{\boldsymbol{U}}_t^\top)\beta$. The second term in the summation is the error term we consider. We can use the fact that $\boldsymbol{X}\hat{\boldsymbol{U}}_t$ and $\boldsymbol{X}(I - \hat{\boldsymbol{U}}_t\hat{\boldsymbol{U}}_t^\top)$ are independent, and $\boldsymbol{X}\hat{\boldsymbol{U}}_t$ is approximately singular to show that

$$\|(\boldsymbol{X}\hat{\boldsymbol{U}}_t)^{-1}\boldsymbol{X}(I - \hat{\boldsymbol{U}}_t\hat{\boldsymbol{U}}_t^\top)\beta\| \gg \|\boldsymbol{X}(I - \hat{\boldsymbol{U}}_t\hat{\boldsymbol{U}}_t^\top)\beta\|$$

which causes explosion of expected generalization error.

Remark 3.4 *In this part, we use the spectral norm to bound the the random matrix itself, which is not tight. For rigorous analysis, we refer readers to [Li et al., 2020, Def. 3.2].*

3.3 Overdetermined case ($R = r$)

In this regime, we will get an exact expectation of the generalization error, both ready for comparing with the upper bound in the overparametrized case (Theorem 3.3) and the lower bound in the marginally overdetermined case (Theorem 3.4).

Same as before, we warm up with the case that $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$.

Theorem 3.6 *If $\hat{\boldsymbol{M}} = \boldsymbol{\Sigma}$, the generalization error satisfies with probability more than $1 - \exp(-\Omega(r))$, with a constant c,*

$$\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha}) \in (1 \pm c\sqrt{r/t})^{-1} \sum_{j=r+1}^{d} \lambda_j \tag{3.4}$$

$R = r$ is the standard choice of dimension reduction for representation learning [Kong et al., 2020b], and we will use its generalization error as baseline to compare with other schemes. Theorem 3.6 proposes an exact value for the expected generalization error, $\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha}) \approx \sum_{j=r+1}^{d}\lambda_j$. This enables us to compare with both the upper bound in Theorem 3.1 and the lower bound in Theorem 3.4.

Theorem 3.1 suggests that

$$\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha}) \lesssim \frac{4\lambda_{r+1}\lambda_1 d^2}{\lambda_r t}.$$

Compare with (3.4), if $\lambda_1 = c_1\lambda_d$, $d = c_2 t$ and $\lambda_{r+1} = c_3\lambda_d$ for some constants c_1, c_2, c_3, then the expected generalization error of overparametrization case is upper bounded by the error of overdetermined case, up to constant factors.

Theorem 3.4 proposes that

$$\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha}) \gtrsim t^2 \sum_{j=t}^{d} \lambda_j$$

In the instance we discussed above for comparing overparametrization and overdetermined cases, that $\lambda_1 = c_1\lambda_d$, and $d = c_2 t$ and $\lambda_{r+1} = c_3\lambda_d$ for some constants c_1, c_2, c_3, the expected generalization error of marginally overdetermined case is at least d^2 times

of the error of the baseline overdetermined case. We will see in Section 4 that marginally overdetermined setup causes explosion of error in our numerical experiments.

The proof of Theorem 3.6 is similar to Theorem 3.4. When the linear regression is sufficiently overdetermined, the smallest singular value of the random matrix is bounded, which is different from the approximate singularity in Theorem 3.4. Therefore the expected generalization error does not explode. We put its proof in the supplement.

Next we study the case when the moment estimator $\hat{\boldsymbol{M}}$ is inexact.

Theorem 3.7 *The expected generalization error satisfies with probability more than* $1-\exp(-\Omega(r))$*, with a constant* c*,*

$$\begin{aligned}&\boldsymbol{E}_{\beta\sim\mathcal{N}(0,\boldsymbol{\Sigma})}\mathcal{L}(\hat{\alpha})\\&\in(1\pm c\sqrt{r/t})^{-1}\cdot\mathbf{tr}(I-\hat{\boldsymbol{U}}_r\hat{\boldsymbol{U}}_r^\top)\boldsymbol{\Sigma}(I-\hat{\boldsymbol{U}}_r\hat{\boldsymbol{U}}_r^\top).\end{aligned}$$

Remark 3.5 *We can see that, when the moment estimator is accurate, i.e.,* $\hat{\boldsymbol{U}}_r=\boldsymbol{U}_r$*, the trace term reaches its minimum. The error of estimating the subspace spanned by the top* r *eigenvector results in bigger generalization error.*

The proof of Theorem 3.7 is an easy extension of Theorem 3.6. We place the proof in the supplement.

4 Experiments

In this section, we will illustrate Theorems 3.3, 3.4 and 3.7 by simulation and an experiment on the MNIST dataset. When we choose the representation dimension $R=d,r$ to make overparametrized or overdetermined, we should see small generalization error, compared to $R=t$ when we should see big error. We will run the algorithm with the representation dimension ranging, and plot their empirical generalization error, which verifies Theorems 3.3, 3.4 and 3.7.

The dimension of the problem is d, the effective dimension of the features is r, which means the feature vector $\beta\in\mathbb{R}^d$ is generated from $\mathcal{N}(0,\boldsymbol{\Sigma})$, and λ_{r+1}, the $r+1$-th eigenvalue of $\boldsymbol{\Sigma}$, is much smaller than the r-th eigenvalue λ_r. The number of meta-test data is t, and $r<t<d$. The representation dimension, at which we perform dimension reduction, is R.

We considered three cases. The baseline we compare with is $R=r$, where we take the top r eigenvectors of covariance estimator for meta-test, and we give an exact (up to constant constant) theoretical error bound. When $R=d$, we solve an overparametrized linear regression. The error is as small as when $R=r$. When solving the linear regression at $R=t$, the error is much bigger than the other two cases.

We will plot the empirical generalization error versus different choice of R. Set $r=10$, and the 11th eigenvalue of $\boldsymbol{\Sigma}$ is either 0 or very small. In Fig. 1, we first choose $\boldsymbol{\Sigma}=[I_{10},0;0,0]$, an exact low dimensional feature, which means the covariance is exactly rank-r. We first generate 50 tasks for estimating the covariance of β, and get the matrix $\hat{\boldsymbol{M}}$. Note that $\hat{\boldsymbol{M}}$ is not necessarily equal to $\boldsymbol{\Sigma}$. Then we range R from 2 to 100, and train it on meta-test task, which is also generated from $\boldsymbol{\Sigma}$. After training we test it with the new data generated i.i.d. from the same task, and record the generalization error. We repeat meta-test for 50 times and report the average generalization error. We plot two lines with different $\boldsymbol{A}$. The "weighted" case corresponds to $\boldsymbol{A}=\sqrt{\max\{\hat{\boldsymbol{\Lambda}}_R,0\}}$, and the "plain" case corresponds to $\boldsymbol{A}=I$. The solutions are same in overdetermined case, but with overparametrization the solutions are different. In the second figure we plot the case when $\boldsymbol{\Sigma}=[I_{10},0;0,0.01\cdot I_{90}]$, the other settings are the same.

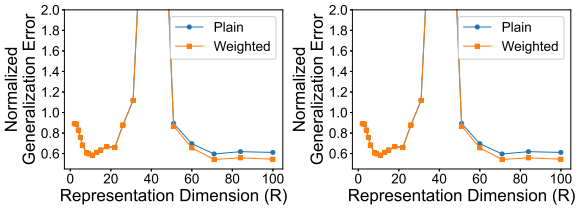

Figure 1: Synthetic data, error of meta-test with different representation dimension. Left, $\boldsymbol{\Sigma}=[I_{10},0;0,0]$; Right, $\boldsymbol{\Sigma}=[I_{10},0;0,0.01\cdot I_{90}]$. "Plain" and "Weighted" corresponds to $\boldsymbol{A}=I$ or $\boldsymbol{A}=\sqrt{\max\{\hat{\boldsymbol{\Lambda}}_R,0\}}$. Small error when $R=r$ or $R=d$, big error when $R=t$.

In Figure 1, the generalization error when $R=r$ and $R=d$ are both small, because the covariance is approximately low rank. In Figure 2, we choose the covariance of task as $\boldsymbol{\Sigma}=\mathrm{diag}(\mathbf{1}_{10},0.2\cdot\mathbf{1}_{10},0.1\cdot\mathbf{1}_{80})$, or $\boldsymbol{\Sigma}$ such that $\boldsymbol{\Sigma}_{j,j}=(1+9j/100)^{-3}$, whose eigenvalues decrease fast but has no clear gaps in between. We can still see in the figure that the generalization error is small on two sides $R=r$ or $R=d$ and explodes when $R=t$. Also the generalization error of $R=d$ case is slightly smaller than $R=r$. This is due to that the tail of the eigenvalues is heavier than the settings in Figure 1, thus the eigenvalue

truncation causes more error.

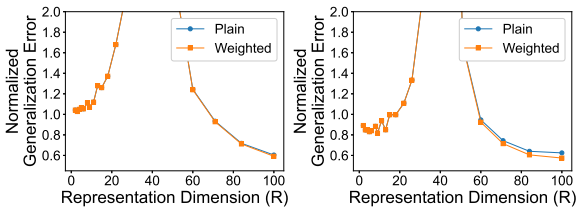

Figure 2: Synthetic data, error of meta-test with different representation dimension. Left, $\mathbf{\Sigma} = \mathrm{diag}(\mathbf{1}_{10}, 0.2 \cdot \mathbf{1}_{10}, 0.1 \cdot \mathbf{1}_{80})$; Right, $\mathbf{\Sigma} = \mathrm{diag}((1 + 9j/100)^{-3})$. Small error when $R = r$ or $R = d$, big error when $R = t$.

We also run linear regression on MNIST dataset to classify the images of numbers 0 to 9. Each image is of size 28×28 and we flatten it to $\mathbb{R}^{784}$. Let $\boldsymbol{x}_{i,j}$ be the i-th image in the j-th class, where $j = 0, ..., 9$. We take the images of number 3 to 9, each 3000 samples for estimating the feature covariance. Different from mixed linear regression, we split the images of each class into two disjoint batches, each consisting of 1500 samples. Let $t_j = 3000$ for all j, The covariance estimator is

$$\hat{\boldsymbol{M}} = \sum_{j=3}^{9} \frac{2}{t_j^2} \Bigg[\Big(\sum_{i=1}^{t_j/2} \boldsymbol{x}_{i,j} \Big) \Big(\sum_{i=t_j/2+1}^{t_j} \boldsymbol{x}_{i,j} \Big)^\top + \Big(\sum_{i=t_j/2+1}^{t_j} \boldsymbol{x}_{i,j} \Big) \Big(\sum_{i=1}^{t_j/2} \boldsymbol{x}_{i,j} \Big)^\top \Bigg]$$

Then we record the eigenvalues of $\hat{\boldsymbol{M}}$ and truncate $\hat{\boldsymbol{M}}$ to different ranks for further use. The meta-test data contains 125 images of number 0 to 2, and we will learn a classifier for them. We apply one-hot encoding to the labels. If the image is 0, the label is transformed to $[1, 0, 0]$, similarly for $1, 2$. Note that the label is 3 dimensional, but we can still run linear regression. Let $\boldsymbol{y} \in \mathbb{R}^{125 \times 3}$ where each row is a one-hot encoded label. The solution is

$$\hat{\alpha} = \mathrm{argmin}_{\alpha \in \mathbb{R}^{R \times 3}} \| \boldsymbol{X}\boldsymbol{B}\alpha - \boldsymbol{y} \|^2$$

The empirical generalization error is plotted in Figure 3. Surprisingly, the generalization error without weighting is smaller than that with weighting. We believe the reason is that, the MNIST dataset do not satisfy the assumptions in our theorem. The images of each class is clustered about a point rather than zero centered Gaussian random vectors, and the feature space of numbers 3 to 9 might not cover the feature space of numbers 0 to 2. Although the real data does not have the desired statistical propertied above, it still suffices to validate the double descent theory. The generalization error is low when the rank is small but above 10 (the representation dimension is above the effective dimension). With R becoming bigger, the generalization error increases but drops again when $R > t$ and gets closer to 784. We also see that the error is lower when $R = 784$ compared to any low-dimensional representation.

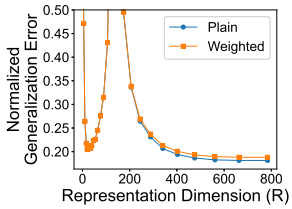

Figure 3: MNIST data, error of meta-test with different representation dimension. Training data: images of 3-9; Test data: images of 0-2.

5 Conclusion and discussion

This paper sheds light on the role of representation dimension setup, with respect to the estimation of approximate low-dimensional task features. With restricted data, although dimension reduction is often believed as necessary, our theory proposes that, with the benefit of the weighting scheme, the solution of overparametrized least square without dimension reduction generalizes well. We further compare the overparametrization, marginally overdetermined and overdetermined cases, whose generalization errors match the numerical experiments that show the double descent behavior.

There are several directions to study in the future. First, although overparameterization benefits for parameter estimation, the performance can be worse with respect to the perturbation by noise. A trade-off scheme that balances the effects is of interest. Secondly, the moment estimator requires that the the covariance of $\boldsymbol{x}_{i,j}$ is known. The investigation of a more flexible estimator, which allows $\boldsymbol{x}_{i,j}$ following an unknown subGaussian distribution, is more practical and enables us to see how the distribution of data affects the estimation guarantee. Finally, going beyond linear regression, we are interested in how to formulate the representation learning procedure for non-linear tasks, such as classification, and what is the role of representation dimension.

References

[Allen-Zhu et al., 2019] Allen-Zhu, Z., Li, Y., and Liang, Y. (2019). Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6158–6169.

[Arora et al., 2019] Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. (2019). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332.

[Baxter, 2000] Baxter, J. (2000). A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198.

[Bojar et al., 2014] Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[Chen et al., 2020] Chen, S., Li, J., and Song, Z. (2020). Learning mixtures of linear regressions in subexponential time via fourier moments. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 587–600.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Du et al., 2020] Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. (2020). Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*.

[Hastie et al., 2019] Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. (2019). Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*.

[Jacot et al., 2018] Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580.

[Kong et al., 2020a] Kong, W., Somani, R., Kakade, S., and Oh, S. (2020a). Robust meta-learning for mixed linear regression with small batches. *arXiv preprint arXiv:2006.09702*.

[Kong et al., 2020b] Kong, W., Somani, R., Song, Z., Kakade, S., and Oh, S. (2020b). Meta-learning for mixed linear regression. *arXiv preprint arXiv:2002.08936*.

[Li et al., 2020] Li, M., Sattar, Y., Thrampoulidis, C., and Oymak, S. (2020). Exploring weight importance and hessian bias in model pruning. *arXiv preprint arXiv:2006.10903*.

[Li and Liang, 2018] Li, Y. and Liang, Y. (2018). Learning mixtures of linear regressions with nearly optimal complexity. In *Conference On Learning Theory*, pages 1125–1144.

[Lu et al., 2020] Lu, Y., Ma, C., Lu, Y., Lu, J., and Ying, L. (2020). A mean-field analysis of deep resnet and beyond: Towards provable optimization via overparameterization from depth. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*.

[Maurer et al., 2016] Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(1):2853–2884.

[Montanari et al., 2019] Montanari, A., Ruan, F., Sohn, Y., and Yan, J. (2019). The generalization error of max-margin linear classifiers: High-dimensional asymptotics in the overparametrized regime. *arXiv preprint arXiv:1911.01544*.

[Rudelson and Vershynin, 2008] Rudelson, M. and Vershynin, R. (2008). The least singular value of a random square matrix is o (n- 1/2). *Comptes Rendus Mathematique*, 346(15-16):893–896.

[Tripuraneni et al., 2020] Tripuraneni, N., Jin, C., and Jordan, M. I. (2020). Provable meta-learning of linear representations. *arXiv preprint arXiv:2002.11684*.

[Vershynin, 2010] Vershynin, R. (2010). Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*.

[Wu and Xu, 2020] Wu, D. and Xu, J. (2020). On the optimal weighted ℓ_2 regularization in overparameterized linear regression.

[Zhong et al., 2016] Zhong, K., Jain, P., and Dhillon, I. S. (2016). Mixed linear regression with multiple

components. In *Advances in neural information processing systems*, pages 2190–2198.