SAMPLE EFFICIENT SUBSPACE-BASED REPRESENTATIONS FOR NONLINEAR META-LEARNING

Author(s) Name(s)

Author Affiliation(s)

ABSTRACT

Constructing good representations is critical for learning complex tasks in a sample efficient manner. In the context of meta-learning, representations can be constructed from common patterns of previously seen tasks so that a future task can be learned quickly. While recent works show the benefit of subspace-based representations, such results are limited to linear-regression tasks. This work explores a more general class of nonlinear tasks with applications ranging from binary classification, generalized linear models and neural nets. We prove that subspace-based representations can be learned in a sample-efficient manner and provably benefit future tasks in terms of sample complexity. Numerical results verify the theoretical predictions in classification and neural-network regression tasks.

***Index Terms*—** representation learning, binary classification, generalized linear models, nonlinear problems

1 Introduction

Meta-learning (and multi-task learning) has proved to be a powerful technique when available training data is limited. The central idea is exploiting the information (e.g. training data) provided by earlier related tasks to quickly adapt a new task using few samples. This idea has a rich history [1, 2] and has shown promise in modern machine learning tasks, e.g., in image classification [3], machine translation [4] and reinforcement learning [5], all of which may involve numerous tasks to be learned with limited data per task.

Modern deep learning algorithms typically exploit the shared information between tasks by learning useful representations [6, 7]. The multi-task system was studied by [1], and the idea of meta-learning or transfer learning is investigated empirically in modern machine learning framework, showing that the shared representation benefits for training on the new tasks [8, 9, 10, 11]. An instructive and well-studied problem for meta-learning is mixed linear regression, for which efficient algorithms and sample complexity bounds are discussed in [12, 13, 14]. If the tasks lie on a shared low-dimensional subspace, learning this subspace would serve as an efficient representation which helps reduce the search space for future tasks. Once the search space is low dimensional, in order to get the same accuracy, the amount of data required for training is reduced compared to training over the full parameter space. [15, 16, 17, 18] propose sample complexity bounds for representation learning for linear multi-task systems. There are study of mixed linear tasks combined with other structures, such as boolean combination of features [19], half-spaces [20] and sparse representations [21].

The recent papers [22, 23] propose meta-learning procedures that involve dimension reduction, clustering and few-shot learning. Here a low-dimensional task subspace is used as the search space for few-shot learning for the new task. Another related approach [24, 25] sets 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, existing provable algorithms for representation learning are restricted to linear-regression tasks, whereas typical machine learning tasks involve nonlinearity. This can arise from the use of nonlinear models as well as nonlinear label link function (e.g. generalized linear models). A good example is classification problems which represent much of the machine learning applications including computer vision and natural language processing []. In classification tasks, the model is a map from images to labels, and the labels are discrete and not linear with respect to the input images (i.e. logistic link function). Another example is the use of nonlinear models such as deep networks, which contain many nonlinear activation functions within their layers. The existing results for representation learning for the linear-regression setting cannot be easily extended to the nonlinear case.

Can we learn efficient subspace representations for nonlinear tasks such as generalized linear models and neural nets?

We consider a realizable setup where the input data is high-dimensional, the *relevant features* lie in a low dimensional subspace and the labels depend only on the *relevant features*. These assumptions are the same as in the existing literature, however we additionally allow for the scenario where labels are possibly an arbitrary nonlinear function of the relevant features. We make the following contributions.

Thanks to XYZ agency for funding.

mf: main poin
low-dimensiona
approximation
feature space
learning easie
new task, with
samples. Thus
many ML applic
one seeks such
representation
can omit 'lear
by mixed linea
regression'.
parag.

mf: swap the
with next para

SO: What is tr
You mean few-s
adaptation?

mf: what does
'objective' mea
do you mean th
are different
to be performe
different goal
the same data?

SO: Redundant
sentence: In
datasets, ther
images of diff
types of objec
and the task i
to classify wh
categories eac
comes from.

ere? give
n
ould we
term
arning or
sk learning?
tter to pick
consistency

• **Efficient representations for nonlinear tasks:** We show that subspace found via method-of-moments (MOM) leads to a consistent estimate of the ground-truth subspace despite arbitrary task nonlinearities, when the data is normally distributed. We combine this with non-asymptotic learning results to establish sample complexity bounds for representation learning.

• **Few-shot learning and Applications:** We specialize our results to practical settings with tasks involving binary classification and neural nets. We theoretically and empirically show that subspace-based representation can greatly improve sample efficiency of future tasks.

2 Problem Formulation

The meta-learning setup that will be considered in this work consists of two phases: (i) meta-training: prior tasks are used to learn a good representation and (ii) few-shot learning: the new task is learned with few samples. In the meta-training phase, we learn the low dimensional space spanned by parameters. In the few-shot learning phase, we use the subspace to learn the model of a new task ideally with few samples.

In the first phase, there are multiple tasks to infer from, each with its own distribution. We consider a realizable model where the input and label is associated via a labeling function. One accesses batches of data, each of whom is collected from a task, however we may not know which task it comes from. We make this setup more precise using the following definitions. Below, the ground-truth representation will be denoted by a matrix $\boldsymbol{W} \in \mathbb{R}^{r\times d}$ row space of which corresponds to the subspace of interest.

Definition 2.1. ***Meta-training data.*** *Fix a matrix $\boldsymbol{W} \in \mathbb{R}^{r\times d}$ satisfying $\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I}$. The j-th task is associated with function $f^j : \mathbb{R}^r \to \mathbb{R}$. Given input $\mathbf{x}$, the label y is distributed as $p_j(y|\mathbf{x}) = p_j(y|\boldsymbol{W}\mathbf{x})$[1] and the expectation satisfies $\boldsymbol{E}(y) = f^j(\boldsymbol{W}\mathbf{x})$. Suppose there are n_j samples from the j-th task sampled i.i.d. from this distribution and we denote the dataset $\mathcal{S}^j = (\mathbf{x}_{i,j}, y_{i,j})_{i=1}^{n_j}$. Define the full meta-training dataset to be $\mathcal{S} = \bigcup_{j=1}^{k} \mathcal{S}^j$.*

Definition 2.2. ***Binary classification.*** *The labels $y_{i,j}$ are generated as*

$$y_{i,j} = \begin{cases} 1, & \text{with probability } f^j(\boldsymbol{W}\mathbf{x}_{i,j}), \\ 0, & \text{with probability } 1 - f^j(\boldsymbol{W}\mathbf{x}_{i,j}). \end{cases}$$

Here $f(a) \in [0,1]$ for all $a \in [-1,1]$, and the event that $f(a) = 0$ is not true with probability 1 with respect to Lebesgue measure on $[-1,1]$[2]. Denote the distribution of $\mathbf{x}, y$ as $\mathscr{P}_{x,y}$.

An example of the class of f^j functions is a parametrized family

$$f^j(\boldsymbol{W}\mathbf{x}_{i,j}) := \phi(\theta_j^T \boldsymbol{W}\mathbf{x}_{i,j}) \tag{2.1}$$

for $\theta_j \in \mathbb{R}^r$, ϕ is a function mapping from $\mathbb{R}$ to $(0,1)$.

Definition 2.3. ***Non-linear regression.*** *More generally, for $\Theta^j \in \mathbb{R}^{r\times d}$, the labels $y_{i,j}$ are generated as*

$$y_{i,j} = f^j(\boldsymbol{W}\mathbf{x}_{i,j}) + \epsilon,$$

ϵ is a zero mean bounded random variable. f can be any Lipschitz non-linear function, i.e., a neural network[3].

When the dimension of the span of parameters is low, [22] performs a dimension reduction algorithm to find the low dimensional subspace that the parameters span. This is done by selecting the top eigenvectors of the covariance estimator.

Definition 2.4. ***Moment estimator of covariance.*** *We define the covariance estimator as*

$$\hat{\boldsymbol{M}} = \sum_{j=1}^{k} \frac{2}{n_j^2} \Bigg[\Big(\sum_{i=1}^{n_j/2} y_{i,j}\mathbf{x}_{i,j} \Big) \Big(\sum_{i=n_j/2+1}^{n_j} y_{i,j}\mathbf{x}_{i,j} \Big)^\top \tag{2.2a}$$

$$+ \Big(\sum_{i=n_j/2+1}^{n_j} y_{i,j}\mathbf{x}_{i,j} \Big) \Big(\sum_{i=1}^{n_j/2} y_{i,j}\mathbf{x}_{i,j} \Big)^\top \Bigg] \tag{2.2b}$$

.

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 $\hat{\boldsymbol{M}}_r = \hat{\boldsymbol{U}}_r \mathrm{diag}(\hat{\lambda}_1, ..., \hat{\lambda}_r)\hat{\boldsymbol{U}}_r^\top$.

In the next section, we will prove that the column space of $\hat{\boldsymbol{U}}$ is close to the column space of $\boldsymbol{\Sigma}$.

In Algorithm 1, the output $\hat{\boldsymbol{U}}_r$ is the estimator of the subspace spanned by the parameters. $\hat{\boldsymbol{U}}_r$ is used for training step in few-shot learning phase. For the new task, we search for the function f^* that minimizes the population loss. We give a simple analysis for cross entropy loss, which is usually adopted for classification tasks.

Definition 2.5. ***Few-shot learning.*** *In the few-shot learning phase, suppose If using population loss, we shouldn't say "generate" $\mathbf{x} \in \mathbb{R}^d$ are generated independently, following standard normal distribution, $y \in \mathbb{R}$. $\mathbf{x}$, y satisfy $\boldsymbol{E}(y) = f(\boldsymbol{W}\mathbf{x})$. Lef $\mathcal{F}$ be a family of functions as the search space for few-shot activation function. We search for the optimal f^* by*

$$f^* = \operatorname*{argmin}_{f'\in\mathcal{F}} \mathcal{L}_{\mathrm{CE}}(f'; \boldsymbol{U}_r^\top) \tag{2.3}$$

where $\mathcal{L}_{\mathrm{CE}}$ is population cross entropy loss, defined as

$$\mathcal{L}_{\mathrm{CE}}(f; \boldsymbol{P}) : \mathcal{F} \times \mathbb{R}^{r\times d} \to \mathbb{R} =$$
$$- \boldsymbol{E}_{\mathbf{x},y\sim\mathscr{P}_{x,y}} (y \log f(\boldsymbol{P}\mathbf{x}) + (1-y)\log(1 - f(\boldsymbol{P}\mathbf{x}))) \tag{2.4}$$

If $\boldsymbol{P}$ is known, we use $\mathcal{L}_{\mathrm{CE}}(f(\boldsymbol{P}x))$ as an equivalent way of representing $\mathcal{L}_{\mathrm{CE}}(f; \boldsymbol{P})$.

[1] In words, the label only depends on the relevant features induced by $\boldsymbol{W}$.

[2] In other words, f is not a constant 0 function.

[3] In the theorem, we treat f as a general linear function, and in experiments we will simulate with a neural network with a specific structure.

Remark 2.1. *In binary classification example, if I want to say $\mathcal{F} = \phi \circ \theta \circ$ (variable)*

$$\mathcal{F}(u) = \{\phi(\theta^\top u) \mid \|\theta\| \le a\}, \tag{2.5}$$

Let the new data be generated with $f(\boldsymbol{W}\mathbf{x}) = \phi(\theta^\top \boldsymbol{W}\mathbf{x})$, $\mathbf{x}$ is standard normal random vector, then f^* can be defined as

$$f^* = \operatorname*{argmin}_{\tilde{\theta}} \mathcal{L}_{\mathrm{CE}}(\phi(\tilde{\theta}^\top \hat{\boldsymbol{U}}_r^\top \mathbf{x})), \text{ such that } \|\theta\| \le a. \tag{2.6}$$

Algorithm 1 Meta-training and Few-shot Learning

Require: Dataset $\mathcal{S}$, representation size r, function space $\mathcal{F}$

Compute $\hat{\boldsymbol{M}}$ via method-of-moments (2.2).

r-eigenvalue truncation:

$\hat{\boldsymbol{M}}_r \leftarrow \hat{\boldsymbol{U}}_r \mathrm{diag}(\hat{\boldsymbol{\Lambda}}_{1,1}, ..., \hat{\boldsymbol{\Lambda}}_{r,r}) \hat{\boldsymbol{U}}_r^\top$.

$f^* \leftarrow \operatorname{argmin}_{f \in \mathcal{F}} \mathcal{L}_{\mathrm{CE}}(f; \boldsymbol{U}_r^\top)$. $\mathcal{L}_{\mathrm{CE}}$ is defined as (2.4).

return $\hat{\boldsymbol{U}}_r$, f^*.

3 Main theorems

In this section, we shall establish error bounds for Algorithm 1. This involves two parts. Theorem A.1 upper bounds the spectral error $\|\hat{\boldsymbol{M}} - \boldsymbol{M}\|$ of method-of-moments estimator. The estimator is then used for few-shot learning, and Theorem 3.2 upper bounds the cross-entropy loss in the few-shot learning step.

Define

$$\boldsymbol{h}_j(\boldsymbol{W}) : \mathbb{R}^{r\times d} \to \mathbb{R}^d = \boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)} f_j(\boldsymbol{W}\mathbf{x})\mathbf{x}$$

In Algorithm 1, $\hat{\boldsymbol{M}}$ is the estimator of

$$\boldsymbol{M} := \boldsymbol{W}^\top \boldsymbol{W} \left(\frac{1}{k} \sum_{j=1}^{k} \boldsymbol{h}_j(\boldsymbol{W}) (\boldsymbol{h}_j(\boldsymbol{W}))^\top \right) \boldsymbol{W}^\top \boldsymbol{W}$$

whose rank is r and the column space is the same as the row space of $\boldsymbol{W}$. We first present the error of the estimator $\hat{\boldsymbol{M}}$.

Theorem 3.1. *Suppose the data are generated as Def. 2.1. We suppose $y\mathbf{x}$ is a subGaussian random vector with covariance being upper bounded by*

$$\|\mathbf{Cov}(y\mathbf{x})\| \le \sigma^2. \tag{3.1}$$

(e.g. if $f_j(x) < \sigma$ then $\|\mathbf{Cov}(y\mathbf{x})\| \le \sigma^2$.)

Let $\delta \in (0,1)$, $\epsilon \in (0,1)$, σ be defined in (3.1). Then there exists a constant c, with probability at least $1-\delta$, let

$$k \ge \frac{cd}{n} \log^2\left(\frac{kd}{\delta}\right) \max\left\{\frac{1}{\epsilon^2}, \frac{1}{\epsilon} \log\left(\frac{kd}{\delta}\right)\right\}$$

we have

$$\|\hat{\boldsymbol{M}} - \boldsymbol{M}\| \le \epsilon\sigma^2.$$

Theorem A.1 proposes an upper bound for the estimation error of $\boldsymbol{M}$. The subspace spanned by $\hat{\boldsymbol{M}}$ is used for few-shot learning. In the context of binary classification, we will show how this error affects the cross entropy loss for the few-shot learning stage.

Define the SVD of $\hat{\boldsymbol{M}}$ as $\hat{\boldsymbol{U}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{V}}^\top$. Should we use SVD or eigendecompostion? Let the first r columns of $\hat{\boldsymbol{U}}$ be $\hat{\boldsymbol{U}}_r$, and we denote $\hat{\boldsymbol{W}}$ as

$$\hat{\boldsymbol{W}} = \hat{\boldsymbol{U}}_r \hat{\boldsymbol{Q}}, \tag{3.2}$$

$$\hat{\boldsymbol{Q}} = \operatorname*{argmin}_{\boldsymbol{Q}\in\mathbb{R}^{r\times r}, \boldsymbol{Q}\boldsymbol{Q}^\top = I} \|\hat{\boldsymbol{U}}_r \boldsymbol{Q} - \boldsymbol{W}^\top\| \tag{3.3}$$

With the definition of $\hat{\boldsymbol{W}}$, $\|\hat{\boldsymbol{W}} - \boldsymbol{W}\|$ defines a distance between the row space of $\boldsymbol{W}$ and the column space of $\hat{\boldsymbol{U}}_r$. If the span of the two subspaces are the same, then there exists an orthonormal matrix $\boldsymbol{Q}$ such that $\hat{\boldsymbol{U}}_r \boldsymbol{Q} = \boldsymbol{W}^\top$.

Now we use $\hat{\boldsymbol{U}}$ for the downstream task, which means we get the r dimensional representation $\hat{\boldsymbol{U}}_r$, and use it for training on the new task, and we study the population loss of training caused by the error of the estimator $\hat{\boldsymbol{M}}$. We consider the binary classification case, where the data in the new task is generated as Def. 2.2, with the non-linear function being f and the true subspace being $\boldsymbol{W}$. The loss we consider, is cross-entropy loss, which is defined as (2.4).

To train on the data, we need to find the optimal function f that minimizes the population loss. Define the search space of f as $\mathcal{F}$, and we assume that,

1. For any function $f \in \mathcal{F}$, any orthonormal matrix $\boldsymbol{Q} \in \mathbb{R}^{r\times r}$ and any matrix $\boldsymbol{P} \in \mathbb{R}^{r\times d}$, there exists $g \in \mathcal{F}$ such that $f(\boldsymbol{P}\mathbf{x}) = g(\boldsymbol{Q}\boldsymbol{P}\mathbf{x})$.
2. $\mathcal{F} \subseteq \{f \mid \log f,\ \log(1-f) \text{ are } L \text{ Lipschitz}\} \bigcap \{f \mid 0 < f(x) < 1, \forall x \in \mathbb{R}^r\}$.

We can see that, if we adopt the binary classification example in (2.5), then both assumptions are true with $L = 2a$.

We learn the model from the optimization algorithm (2.3) (for binary classification it can be (2.6)). Now we will bound the cross-entropy loss of this model.

Theorem 3.2. *Let $\hat{\boldsymbol{W}}$ be defined as (3.2) and f^* be defined as (2.3). Then we have that*

$$\mathcal{L}_{\mathrm{CE}}(f^*; \hat{\boldsymbol{U}}_r^\top) - \mathcal{L}_{\mathrm{CE}}(f; \boldsymbol{U}_r^\top) \lesssim L\sqrt{r}\|\hat{\boldsymbol{W}} - \boldsymbol{W}\|.$$

Note that $\mathcal{L}_{\mathrm{CE}}(f; \boldsymbol{W})$ depends on the true model and data, so the error caused by inaccuracy of the moment estimator $\hat{M}$ is $O(L\sqrt{r}\|\hat{\boldsymbol{W}} - \boldsymbol{W}\|)$.

Sorry I haven't used Rademacher complexity to prove things before, do you know any references?

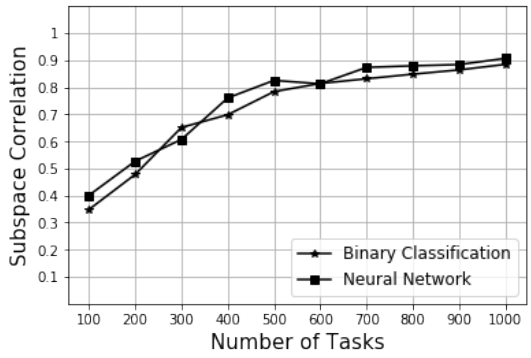
Specifically, the difficulty is that I have to get a **lower** bound for the high dimensional case. The questions are: 1. Rademacher complexity is defined at one point, but here I quantify the distance of the argmin, if the two functions are far away, can their minimizer still be close? 2. There are some

upper bounds for f, for example, f is L Lipschitz, but it is an upper bound of the slope, do I need assumptions about the lower bound, say $|f(x) - f(y)| > l\|x - y\|$?

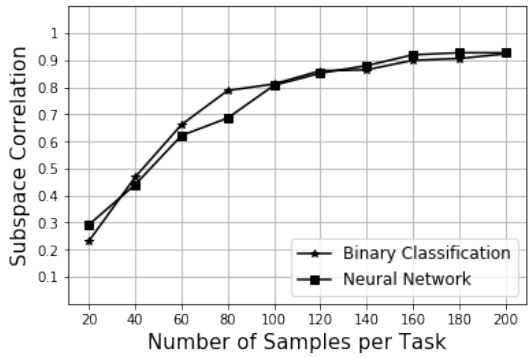
4 Experiment

We generate synthetic datasets with k different tasks, n_j samples of j-th task. As dimension of the data and dimension of the subspace we choose $d = 50$ and $r = 5$, respectively. In all experiments we choose n_j's are being equal to each other.

There are two different setups. In the first one data are generated according to definition 2.2 with f^j's being softmax function for all j's. But for the second setup, there is an underlying 3-layer neural network that fits the data perfectly as in definition 2.3. In both setups our aim is to try to retrieve subspace representations of the data using Algorithm 1.



(a)



(b)

Fig. 1: Subspace correlations, (a) fixed number of samples per task and different number of tasks (b) fixed number of tasks and different numbers of samples per task.

For neural network experiments, we assume that the data are generated from a ground truth neural network which has 3 layers, defined as

$$y_{i,j} = f^j(\mathbf{x}_{i,j}) + \epsilon_{i,j} = \boldsymbol{W}_{j3}(\boldsymbol{W}_{j2}(\boldsymbol{W}_{j1}(\boldsymbol{U}_r^\top \mathbf{x}_{i,j}))_+)_+ + \epsilon_{i,j}$$

where $\epsilon_{i,j} \sim \mathcal{N}(0,1)$ is gaussian noise, $(\cdot)_+$ is ReLU activation function, $\boldsymbol{U}_r \in \mathbb{R}^{50\times 5}$ is representation matrix which is same for all j's. The weight matrices $\boldsymbol{W}_{j1}, \boldsymbol{W}_{j2}$ and $\boldsymbol{W}_{j3}$ are different for each task and they are random gaussian matrices in $\mathbb{R}^{20\times 5}, \mathbb{R}^{20\times 20}, \mathbb{R}^{1\times 20}$ respectively.

In Fig.1, we apply Algorithm 1 to recover the r dimensional subspace and get $\hat{\boldsymbol{U}}_r$, and check its distance to the true feature space $\boldsymbol{U}_r$ under different numbers of samples per task and different number of tasks. We use the subspace correlation as the metric for evaluating the accuracy of subspace recovery, which is defined by $\frac{\|\hat{\boldsymbol{U}}_r{}^\top \boldsymbol{U}_r\|^2}{\|\boldsymbol{U}_r\|^2}$. In Fig.1a $n_j = 30$ for all tasks while k changes from 100 to 1000. In Fig.1b $k = 100$ is fixed but n_j's vary from 20 to 200.

In Fig. 2 new task accuracies are depicted. For new task, a new 1-layer neural network without any activation function is trained using binary cross-entropy loss. For this setup, $n_j = 50$ for all tasks, $k = 2000$ is fixed and the number of samples for new task varies from 20 to 400. Depending on the samples number, test data accuracy can be seen from the graph.

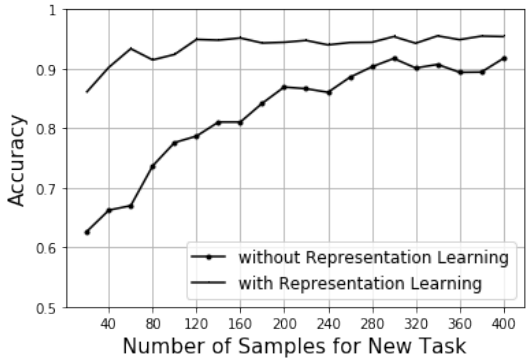


Fig. 2: Accuracy for downstream task with and without Representation Learning

5 References

[1] Rich Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[2] Jonathan Baxter, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[4] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling,

Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al., "Findings of the 2014 workshop on statistical machine translation," in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 12–58.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, 2017, pp. 1126–1135.

[6] Jürgen Schmidhuber, *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*, Ph.D. thesis, Technische Universität München, 1987.

[7] Sebastian Thrun and Lorien Pratt, *Learning to learn*, Springer Science & Business Media, 2012.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[9] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey, "Meta-learning in neural networks: A survey," *arXiv preprint arXiv:2004.05439*, 2020.

[10] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[11] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of maml," *arXiv preprint arXiv:1909.09157*, 2019.

[12] Kai Zhong, Prateek Jain, and Inderjit S Dhillon, "Mixed linear regression with multiple components," in *Advances in neural information processing systems*, 2016, pp. 2190–2198.

[13] Yuanzhi Li and Yingyu Liang, "Learning mixtures of linear regressions with nearly optimal complexity," in *Conference On Learning Theory*, 2018, pp. 1125–1144.

[14] Sitan Chen, Jerry Li, and Zhao Song, "Learning mixtures of linear regressions in subexponential time via fourier moments," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, pp. 587–600.

[15] Karim Lounici, Massimiliano Pontil, Sara Van De Geer, Alexandre B Tsybakov, et al., "Oracle inequalities and optimal inference under group sparsity," *The annals of statistics*, vol. 39, no. 4, pp. 2164–2204, 2011.

[16] Massimiliano Pontil and Andreas Maurer, "Excess risk bounds for multitask learning with trace norm regularization," in *Conference on Learning Theory*, 2013, pp. 55–76.

[17] Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile, "Linear algorithms for online multitask classification," *The Journal of Machine Learning Research*, vol. 11, pp. 2901–2934, 2010.

[18] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes, "The benefit of multitask representation learning," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2853–2884, 2016.

[19] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala, "Efficient representations for lifelong learning and autoencoding," in *Conference on Learning Theory*, 2015, pp. 191–210.

[20] Irina Rish, Genady Grabarnik, Guillermo Cecchi, Francisco Pereira, and Geoffrey J Gordon, "Closed-form supervised dimensionality reduction with generalized linear models," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 832–839.

[21] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil, "Convex multi-task feature learning," *Machine learning*, vol. 73, no. 3, pp. 243–272, 2008.

[22] Weihao Kong, Raghav Somani, Zhao Song, Sham Kakade, and Sewoong Oh, "Meta-learning for mixed linear regression," *arXiv preprint arXiv:2002.08936*, 2020.

[23] Weihao Kong, Raghav Somani, Sham Kakade, and Sewoong Oh, "Robust meta-learning for mixed linear regression with small batches," *arXiv preprint arXiv:2006.09702*, 2020.

[24] Simon S Du, Wei Hu, Sham M Kakade, Jason D Lee, and Qi Lei, "Few-shot learning via learning the representation, provably," *arXiv preprint arXiv:2002.09434*, 2020.

[25] Nilesh Tripuraneni, Chi Jin, and Michael I Jordan, "Provable meta-learning of linear representations," *arXiv preprint arXiv:2002.11684*, 2020.

[26] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan, "A short note on concentration inequalities for random vectors with subgaussian norm," *arXiv preprint arXiv:1902.03736*, 2019.

A Proof of main theorems

Theorem A.1. *Suppose the data are generated as Def. 2.1. Let $\delta \in (0,1)$, $\epsilon \in (0,1)$, σ be defined in (3.1). Define*

$$\boldsymbol{h}_j(\boldsymbol{W}) : \mathbb{R}^{r\times d} \to \mathbb{R}^d = \boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)} f_j(\boldsymbol{W}\mathbf{x})\mathbf{x}$$

And let

$$\boldsymbol{M} = \boldsymbol{W}^\top \boldsymbol{W}\left(\frac{1}{k}\sum_{j=1}^{k} \boldsymbol{h}_j(\boldsymbol{W})(\boldsymbol{h}_j(\boldsymbol{W}))^\top\right)\boldsymbol{W}^\top\boldsymbol{W}$$

Then there exists a constant c, with probability at least $1-\delta$, let

$$k = \frac{cd}{n}\log^2(\frac{kd}{\delta})\max\{\frac{1}{\epsilon^2}, \frac{1}{\epsilon}\log(\frac{kd}{\delta})\}$$

we have

$$\|\hat{\boldsymbol{M}} - \boldsymbol{M}\| \le \epsilon\sigma^2.$$

Proof. We first give the lemma for the mean of the random vector $y\mathbf{x}$.

Lemma A.2. *We assume that the data are generated as in Def. 2.1, and we study the j-th task whose activation function is f_j. Define*

$$\boldsymbol{h}_j(\boldsymbol{W}) : \mathbb{R}^{r\times d} \to \mathbb{R}^d = \boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)} f_j(\boldsymbol{W}\mathbf{x})\mathbf{x}$$

Denote the joint distribution of $(\mathbf{x}, y)$ as $\mathscr{P}_{x,y}$. Then $\boldsymbol{E}_{(\mathbf{x},y)\sim\mathscr{P}_{x,y}}(y\mathbf{x}) = \boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{h}_j(\boldsymbol{W})$.

Proof. Denote The expectation can be expanded as

$$\begin{aligned}
&\boldsymbol{E}_{(\mathbf{x},y)\sim\mathscr{P}_{x,y}}(y\mathbf{x})\\
&= \boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)} f_j(\boldsymbol{W}\mathbf{x})\mathbf{x}\\
&= \boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)}\left(f_j(\boldsymbol{W}\mathbf{x})\boldsymbol{W}^\top\boldsymbol{W}\mathbf{x} + f_j(\boldsymbol{W}\mathbf{x})(I-\boldsymbol{W}^\top\boldsymbol{W})\mathbf{x}\right).
\end{aligned}$$

Note that, because $\mathbf{x}\sim\mathcal{N}(0,I)$ so $\boldsymbol{W}\mathbf{x}$ and $(I-\boldsymbol{W}^\top\boldsymbol{W})\mathbf{x}$ are Gaussian. $\boldsymbol{W}\boldsymbol{W}^\top = I$ implies

$$\begin{aligned}
&\boldsymbol{E}(\boldsymbol{W}\mathbf{x})\mathbf{x}^\top((I-\boldsymbol{W}^\top\boldsymbol{W}))\\
&= \boldsymbol{W}(I-\boldsymbol{W}^\top\boldsymbol{W}) = 0.
\end{aligned}$$

So $\boldsymbol{W}\mathbf{x}$ and $(I-\boldsymbol{W}^\top\boldsymbol{W})\mathbf{x}$ are independent, so

$$\begin{aligned}
&\boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)} f_j(\boldsymbol{W}\mathbf{x})(I-\boldsymbol{W}^\top\boldsymbol{W})\mathbf{x}\\
&= (\boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)} f_j(\boldsymbol{W}\mathbf{x}))(\boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)}(I-\boldsymbol{W}^\top\boldsymbol{W})\mathbf{x}) = 0
\end{aligned}$$

Then

$$\begin{aligned}
\boldsymbol{E}_{(\mathbf{x},y)\sim\mathscr{P}_{x,y}}(y\mathbf{x}) &= \boldsymbol{W}^\top\boldsymbol{W}(\boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)} f_j(\boldsymbol{W}\mathbf{x})\mathbf{x})\\
&= \boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{h}_j(\boldsymbol{W}).
\end{aligned}$$

□

The following lemmas are similar to [22] Sec A.1. We extend the concentration inequalities from bounded random vectors to Gaussian random vectors. For completeness we place the lemmas here.

Lemma A.3. *([26] Cor. 7) Let $\delta \in (0,1)$, $t > 0$, with probability $1-\delta$, there exists a constant c such that*

$$\|\frac{1}{t}\sum_{i=1}^{t} y_{i,j}\mathbf{x}_{i,j} - \boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{h}_j(\boldsymbol{W})\| \le c\sigma\sqrt{\frac{d}{t}\log(\frac{kd}{\delta})} \quad \text{(A.1)}$$

Denote this event as $\mathcal{E}$.

With the covariance of $y\mathbf{x}$ being bounded by σ^2, the following inequalities are true.

$$\begin{aligned}
\boldsymbol{E}\left((\boldsymbol{v}^T\cdot(\frac{1}{t}\sum_{i=1}^{t} y_{i,j}\mathbf{x}_{i,j} - \theta^j))^2\right) &\le \sigma^2/t, \text{ for } \|\boldsymbol{v}\| = 1.\\
\boldsymbol{E}\left(\|\frac{1}{t}\sum_{i=1}^{t} y_{i,j}\mathbf{x}_{i,j} - \theta^j\|^2\right) &\le \sigma^2 d/t.
\end{aligned}$$

Lemma A.4. *Define*

$$\begin{aligned}
\boldsymbol{Z}_j = &(\frac{2}{n_j}\sum_{i=1}^{n_j/2} y_{i,j}\mathbf{x}_{i,j})(\frac{2}{n_j}\sum_{i=n_j/2+1}^{n_j} y_{i,j}\mathbf{x}_{i,j})^\top\\
&- \boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{h}_j(\boldsymbol{W})(\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{h}_j(\boldsymbol{W}))^\top,
\end{aligned}$$

Then there exists a constant c such that on the event $\mathcal{E}$ (thus with probability $1-\delta$), for all $j = 1, ..., k$,

$$\|\boldsymbol{Z}_j\| \le \frac{c\sigma^2 d}{n_j}\log(\frac{kd}{\delta}). \quad \text{(A.2)}$$

The proof is almost same as [22], the only difference is that we replace the bound in ([22] Prop A.1) by (A.1). With the similar replacement, we propose the following lemma.

Lemma A.5. *Let $\delta \in (0,1)$. There exists a constant c, such that for any $\epsilon \in (0,1)$, and*

$$k = \frac{cd}{n_j}\log^2(\frac{kd}{\delta})\max\{\frac{1}{\epsilon^2}, \frac{1}{\epsilon}\log(\frac{kd}{\delta})\}, \quad \text{(A.3)}$$

with probability $1-\delta$,

$$\|\frac{1}{k}\sum_{j=1}^{k}\boldsymbol{Z}_j\| \le \epsilon\sigma^2.$$

This means that when (A.3), with probability $1-\delta$,

$$\left\|\hat{\boldsymbol{M}} - \boldsymbol{W}^\top\boldsymbol{W}\left(\frac{1}{k}\sum_{j=1}^{k}\boldsymbol{h}_j(\boldsymbol{W})(\boldsymbol{h}_j(\boldsymbol{W}))^\top\right)\boldsymbol{W}^\top\boldsymbol{W}\right\| \le \epsilon\sigma^2.$$

□

Now we will prove Theorem 3.2. We first review the notations and assumptions mentioned in the theorem.

We define the SVD of $\hat{\boldsymbol{M}}$ as $\hat{\boldsymbol{U}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{V}}^\top$. Let the first r columns of $\hat{\boldsymbol{U}}$ be $\boldsymbol{U}_r$.

Let $\hat{\boldsymbol{W}}$ be defined as

$$\hat{\boldsymbol{W}} = \hat{\boldsymbol{U}}_r\hat{\boldsymbol{Q}}, \tag{A.4}$$

$$\hat{\boldsymbol{Q}} = \operatorname*{argmin}_{\boldsymbol{Q}\in\mathbb{R}^{r\times r},\boldsymbol{Q}\boldsymbol{Q}^\top=I} \|\hat{\boldsymbol{U}}_r\boldsymbol{Q} - \boldsymbol{W}\|. \tag{A.5}$$

The population cross entropy loss is defined as

$$\mathcal{L}_{\mathrm{CE}}(f;\boldsymbol{P}) : \mathcal{F}\times\mathbb{R}^{r\times d}\to\mathbb{R} =$$
$$-\boldsymbol{E}_{\mathbf{x},y\sim\mathscr{P}_{x,y}}(y\log f(\hat{\boldsymbol{P}}\mathbf{x}) + (1-y)\log(1-f(\hat{\boldsymbol{P}}\mathbf{x}))).$$

We will search for a function $f^*\in\mathcal{F}$ that minimizes the population cross entropy loss. $\mathcal{F}$ is a set of functions satisfying

1. For any function $f\in\mathcal{F}$, any orthonormal matrix $\boldsymbol{Q}\in\mathbb{R}^{r\times r}$ and any matrix $\boldsymbol{P}\in\mathbb{R}^{r\times d}$, there exists $g\in\mathcal{F}$ such that $f(\boldsymbol{P}\mathbf{x}) = g(\boldsymbol{Q}\boldsymbol{P}\mathbf{x})$.
2. $\mathcal{F}\subseteq\{f \mid \log f,\ \log(1-f) \text{ are } L \text{ Lipschitz}\}\bigcap\{f \mid 0 < f(x) < 1, \forall x\in\mathbb{R}^r\}$.

We solve for f^*, defined as

$$f^* = \operatorname*{argmin}_{f'\in\mathcal{F}} \mathcal{L}_{\mathrm{CE}}(f';\hat{\boldsymbol{U}}_r^\top). \tag{A.6}$$

Now we are ready to state the theorem.

Theorem A.6. *Based on the definition above, we have that*

$$\mathcal{L}_{\mathrm{CE}}(f^*;\hat{\boldsymbol{U}}_r^\top) - \mathcal{L}_{\mathrm{CE}}(f;\boldsymbol{U}_r^\top) \lesssim L\sqrt{r}\|\hat{\boldsymbol{W}} - \boldsymbol{W}\|.$$

Proof. We learn the model from the following optimization algorithm.

$$\begin{aligned}
f^* &= \operatorname*{argmin}_{f'\in\mathcal{F}} \mathcal{L}_{\mathrm{CE}}(f';\hat{\boldsymbol{U}}_r^\top)\\
&= \operatorname*{argmin}_{f'\in\mathcal{F}} -\boldsymbol{E}_{\mathbf{x},y\sim\mathscr{P}_{x,y}}(y\log f'(\hat{\boldsymbol{U}}_r^\top\mathbf{x})\\
&\quad + (1-y)\log(1-f'(\hat{\boldsymbol{U}}_r^\top\mathbf{x})))\\
&= \operatorname*{argmin}_{f\in\mathcal{F}} -\boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)}\left(f(\boldsymbol{W}\mathbf{x})\log f'(\hat{\boldsymbol{U}}_r^\top\mathbf{x})\right.\\
&\quad \left.+(1-f(\boldsymbol{W}\mathbf{x}))\log(1-f'(\hat{\boldsymbol{U}}_r^\top\mathbf{x}))\right).
\end{aligned}$$

Denote $\hat{f}\in\mathcal{F}$ as the function such that $\hat{f}(\hat{\boldsymbol{W}}\mathbf{x}) = f^*(\boldsymbol{U}_r^\top\mathbf{x})$. So that $\mathcal{L}_{\mathrm{CE}}(f^*;\hat{\boldsymbol{U}}_r^\top) = \mathcal{L}_{\mathrm{CE}}(\hat{f};\hat{\boldsymbol{W}})$. Since f^* minimizes the cross entropy loss, we have that $\mathcal{L}_{\mathrm{CE}}(\hat{f};\hat{\boldsymbol{W}}) \le \mathcal{L}_{\mathrm{CE}}(f;\hat{\boldsymbol{W}})$.

Now we have

$$\begin{aligned}
&\mathcal{L}_{\mathrm{CE}}(\hat{f};\hat{\boldsymbol{W}}) - \mathcal{L}_{\mathrm{CE}}(f;\boldsymbol{W})\\
\le\ &\mathcal{L}_{\mathrm{CE}}(f;\hat{\boldsymbol{W}}) - \mathcal{L}_{\mathrm{CE}}(f;\boldsymbol{W})\\
\le\ &-\boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)}\left(f(\boldsymbol{W}\mathbf{x})(\log f(\hat{\boldsymbol{W}}\mathbf{x}) - \log f(\boldsymbol{W}\mathbf{x}))\right.\\
&\left.+(1-f(\boldsymbol{W}\mathbf{x}))(\log(1-f(\hat{\boldsymbol{W}}\mathbf{x})) - \log(1-f(\boldsymbol{W}\mathbf{x})))\right)\\
\le\ &\boldsymbol{E}_{\mathbf{x}\sim\mathcal{N}(0,I)}L\|(\hat{\boldsymbol{W}} - \boldsymbol{W})x\|\\
\lesssim\ &L\sqrt{r}\|\hat{\boldsymbol{W}} - \boldsymbol{W}\|.
\end{aligned}$$

□