Reducing Sample Complexity of Deep Learning with Symmetric Prior of Wireless Tasks

Chengjian Sun, Jiajun Wu and Chenyang Yang
Beihang University, Beijing, China
Email: {sunchengjian, jiajunwu, cyyang}@buaa.edu.cn

***Abstract*—Deep neural networks (DNNs) have been applied to deal with various wireless problems, which usually need a large number of samples for training. Considering that wireless environments are highly dynamic and gathering data is expensive, reducing sample complexity is paramount for learning wireless tasks. Incorporating domain knowledge into learning is a promising way of improving sample efficiency, which is an emerging topic in the wireless community. In this article, we first briefly summarize several approaches to address this core issue. Then, we show that a kind of symmetric property, permutation equivariance, widely exists in wireless tasks. To harness such a generic prior, we introduce a simple data representation method to compress the training set, which is to jointly sort the input and output of the DNNs. We use power allocation among subcarriers, probabilistic content caching, and interference coordination to illustrate how to apply this method, i.e., ranking, and how much the sample complexity can be reduced. From the case study, we find an interesting phenomenon: "sample hardening", where the required number of training samples decreases with the number of subchannels or contents. We compare this method of data representation with the DNNs embedded with the same prior or without any prior. Simulation results show that the samples required to train a DNN after ranking can be reduced by $15 \sim 2400$ folds to achieve the same learning performance as the non-prior counterpart.**

***Index Terms*—Prior knowledge, permutation equivariance, sample complexity, deep learning**

I. Introduction

Deep learning has been applied for a variety of wireless tasks [1–6]. Early research efforts adopt pure data-driven methodology, which entails high sample complexity, defined as the minimal number of training samples to achieve a given performance of learning. For example, to learn an interference coordination policy, about one million samples are used for training a deep neural network (DNN) [1]. For highly dynamic wireless systems, such a large number of samples need to be obtained frequently for re-training DNNs, which incurs high overhead for computing and signalling.

While the availability of vast amount of data is one of the main driver for the success of deep learning, gathering data in wireless networks, say the labels for supervised learning, is expensive. To facilitate training with much fewer samples, an effective approach is to integrate prior knowledge and mathematical models, into the learning process.

Different techniques have been developed along this line, say structure design for DNNs [7–9], transfer learning [10, 11], optimization with unsupervised learning [12], and data representation.

DNN structure design leverages the prior knowledge of the input-output relation of a task for reducing the hypothesis space of the functions. By constructing the DNNs to satisfy a desired property, the family of functions without the property is excluded, which yields low sample complexity. Convolutional neural networks and recurrent neural networks are two notable examples, which respectively exploit the knowledge of spatial and temporal translational invariance of a task [7]. Recently, a class of symmetric DNNs were proposed [8], which are constructed to learn permutation equivariant functions. How to incorporate the well-established models in wireless communications into the DNN structure has also drawn significant attention [9].

Transfer learning strives to transfer the knowledge for a task in a scenario to learn a different but related task or the same task in a different scenario. This technique has been used to address the issue of designing systems with inaccurate models, where an assumed model is used to train a DNN that will be refined by real data [10]. Another example is to learn the complex mixed integer problem, where a pruning policy learned with a pre-trained DNN is used to accelerate the searching process of finding the solution with a few samples gathered in a new environment [11].

Optimization with unsupervised learning leverages the available expressions of the objective and constraint functions of optimization problems. When learning to optimize wireless problems, labels are usually obtained at high cost [1, 2] or even unable to obtain. One approach to circumvent this difficulty is resorting to reinforcement learning, which employs some undirected searching algorithms, say ε-greedy. In the unsupervised learning framework [12], the Lagrangian is taken as the loss function. With the derived gradient of the loss function with respect to (w.r.t.) the output of a DNN, the searching process of gradient based algorithms converge faster with less samples.

Data representation mines the latent relationship between the data and the target variables of a task or among the observed data, which is called feature engineering. Features are the input of DNN, which can be set as or transformed from raw data. Data representation is to find a mapping from the observation space to a feature space for easy learning, which has been widely used for traditional machine learning. For instance, to learn a policy whose output variable is sensitive to the small value of input variable but insensitive to the large input variable, the feature can be set as the logarithm

of the input data. A well-known example of harnessing the data correlation is to transform the observed data into a low dimensional feature space with principle components analysis (PCA) or autoencoder (AE).

Among the four techniques, data representation has rarely been investigated in the literature of wireless communications, except PCA and AE. This may be owing to the fact that deep learning can jointly learn a policy and the informative features to the policy. Yet this does not mean that data pre-processing is not beneficial for deep learning. Data representation can be cast into a function searching problem, which is sample-hungry because the mapping from observation space to feature space has vast possibilities. Task-oriented data representation is challenging. This is because it is difficult to determine which features are relevant to a task, and the connection between the desired features and the task strongly relies on the domain knowledge.

In this article, we discuss how to reduce the sample complexity of deep learning by leveraging a kind of symmetric prior: permutation equivariance (PE) [8]. This kind of prior knowledge has been discovered in many engineering fields and has been harnessed by constructing various DNNs with special structures, say permutation equivariant neural network (PENN) [8, 13] and graph neural networks [14]. We notice that the PE property is inherent in many wireless tasks [1–6, 10–14], thanks to an obvious but overlooked fact: a variety of wireless polices are acted on sets. Moreover, this property has never been leveraged for data representation to assist in learning wireless tasks.

The rest of this article is organized as follows. We first identify the PE property in wireless tasks. Then, we introduce a method for training set compression, ranking, which jointly sorts the observed data and target variables of a task. Next, we provide case study to illustrate the performance of ranking by comparing with fully-connected DNN (FC-DNN) and PENN, and finally we conclude the article.

II. A Generic prior knowledge in Wireless Tasks: Permutation Equivariance

In this section, we show that permutation equivariant wireless tasks are widespread, ranging from physical layer to application layer. We proceed to provide the key components of a policy for such tasks with several concrete examples.

A. What Wireless Tasks are Permutation Equivariant?

Many problems in wireless communications aim to find a set of multivariate functions, which is referred to as a policy in this article. The learning task for these problems is to find the policy that yields a solution for every impacting parameter to achieve a desired metric, which amounts to find the mapping from multiple input variables to multiple output variables. Representative tasks include (but not limited to) interference coordination among transmitter-receiver pairs with power allocation [1] or beam coordination [4], transceiver design [3], resource allocation [2, 12–14], uplink/downlink channel calibration [10], pilot assignment [5], and proactive caching [6].

These wireless policies are executed on a set of objects, e.g., users or contents, as illustrated in Fig. 1 with three objects. The input variables of a policy reflect the states of the objects relevant to the policy, say the channel gains of the users and the popularity of the contents (say files). The output variables of the policy reflect the actions taken on these objects, say the transmit powers allocated to the users and the caching probabilities for the files, which are the target variables of a task. The state or the action of each object can be expressed as a vector, noting that a scalar is a special case of a vector and a matrix can be expressed as a vector. These objects compose a set (e.g., a user set or a file set), where each object is an element of the set.

A group of multivariate functions on a set must be permutation equivariant to the elements in the set [8]. In other words, the response of such a function is indifferent to the ordering of the elements. If the state of every object can fully represent the useful information for the policy, then the policy must be permutation equivariant to the state vectors. That is to say, a wireless policy for a set of objects will be permutation equivariant to the state vectors, if they are appropriately selected.

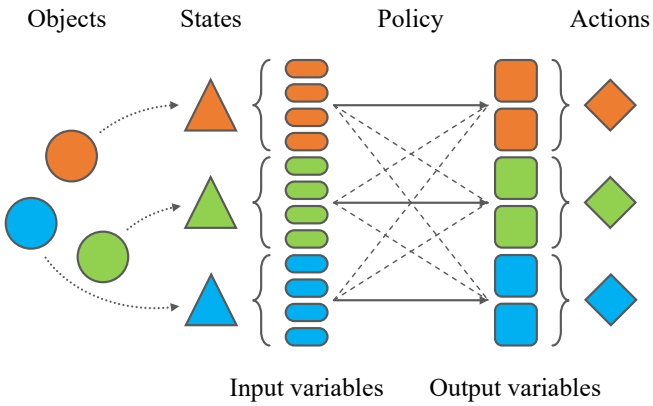

Fig. 1. Relation between states of objects and input variables of a policy and relation between actions of objects and output variables of the policy.

Despite that the PE property widely exists in wireless tasks, it has been realized in wireless community only very recently [13, 14], possibly because of the overlooking of the "latent variables": objects, over which the actions are taken.

B. Several Example Tasks

Now we provide several types of wireless tasks whose policies satisfy the PE property (called PE policies for short in the following), and identify the objects and their states in each policy.

1) Power Allocation among Subcarriers: To ease of understanding, we first consider a classical water-filling power allocation policy in a single user multi-subcarrier system, where the transmit power is allocated among subcarriers. Since the response of the policy is indifferent to the ordering

of the subcarriers, the set of multivariate functions is permutation equivariant. In this example, a subcarrier is an object, the effective noise power at the receiver that is proportional to the channel inverse is the state of the object, and the transmit power allocated to a subcarrier is an action. As illustrated in Fig. 2, when the orders of the first, second and third objects change into the second, third and first, the orders of the input and output variables of the policy change in the same way whereas the policy remains unchanged.

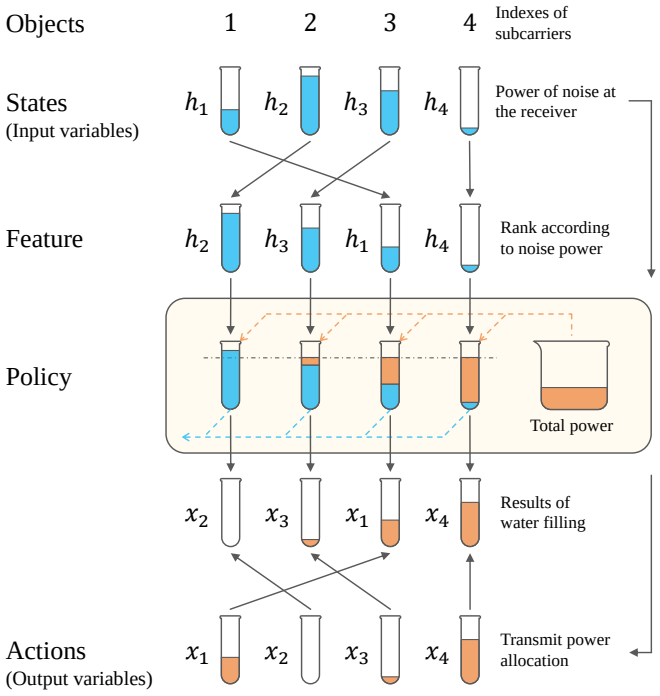

Fig. 2. Permutation equivariance of a power allocation policy to subcarriers.

2) Interference Coordination: Another typical example is interference management, where all base stations (BSs) communicate with the users associated to them over the same frequency band. For easy exposition, consider a scenario in [1], where each single-antenna BS serves one single-antenna user, and the inter-cell interference is mitigated by controlling the transmit power of each BS to maximize a weighted sum rate according to the instantaneous channel gains. In this example, a BS and its associated user is an object, and the transmit power at the BS to the user is the action of the object. The action relies on the channels between the BS to all users, the channels between all BSs to the user, and the weight on the rate, as shown in Fig. 3. Hence, these channel gains and the weight can be set as the state of the object. By defining the object and state in this way, the interference coordination policy is permutation equivariant to the BS-user pairs. It is noteworthy that the policy to be learned for this task will not exhibit PE property if the state does not contain the weight when the weights are unequal. This illustrates the importance of appropriately defining the object and state in order to leverage the PE property inherent in a task.

3) Caching at Wireless Edge: In cache-enabled cellular networks, files are cached at BSs or users according to the states of the files such as their probabilities being requested (i.e., file popularity). By optimizing proactive caching policy (say probabilistic caching policy) based on future file popularity, the network performance such as successful offloading probability (SOP) can be improved [6]. If the order of files changes, the order of the caching probabilities changes in the same way whereas the caching policy remains the same. Therefore, the proactive caching policy is permutation equivariant to the files. In this example, a file is an object, the popularity of the file is the state of the object, and the caching probability for a file is the action.

III. An Approach to Exploit PE Prior: Ranking

In this section, we first introduce the intuition of reducing training samples with data representation by exploiting symmetric prior. Then, we present a simple method to compress the training set for learning the PE policies, ranking.

A. Reducing Sample Complexity with Symmetric Prior

To find a policy for a task with supervised learning, one can train a DNN with the training samples, each consists of a realization of the random states and the corresponding actions of the policy (i.e., the label). All possible training samples span an observation space. Without prior knowledge for the task, the policy can only be learned accurately by a non-structural DNN, i.e., FC-DNN, from training samples. When prior knowledge is available for a task, fewer samples are required either by designing the DNN structure [7, 8], or by mapping the observation space where the data is gathered to the feature space where the samples are used.

A common practice in deep learning is to directly take the observed data as the feature for training. To show why data representation for a task with symmetric prior can decrease training samples, we provide a toy example. Consider a task of learning an axial symmetric function, say quadratic function, with labeled training samples, where each sample contains a positive or negative input variable and the corresponding response of the function. The symmetric property of the function is the prior knowledge, with which the function can be determined by only given the observations of the function on the positive or negative real axis. Therefore, the sign of the input variable is uninformative for learning with the prior knowledge, and the absolute value of the input variable can be taken as the feature. Then, by using the halved training set where each sample only contains positive input variable and the corresponding response, the function can be learnt without sacrificing the learning performance.

B. Ranking and Sample Hardening

PE policies exhibit a kind of symmetric property.

For easy visualization, let us consider a PE policy where the state of every object is a scalar. The policy for the objects is composed of multiple multivariate functions of a state vector and yields an action vector. The state vector of the

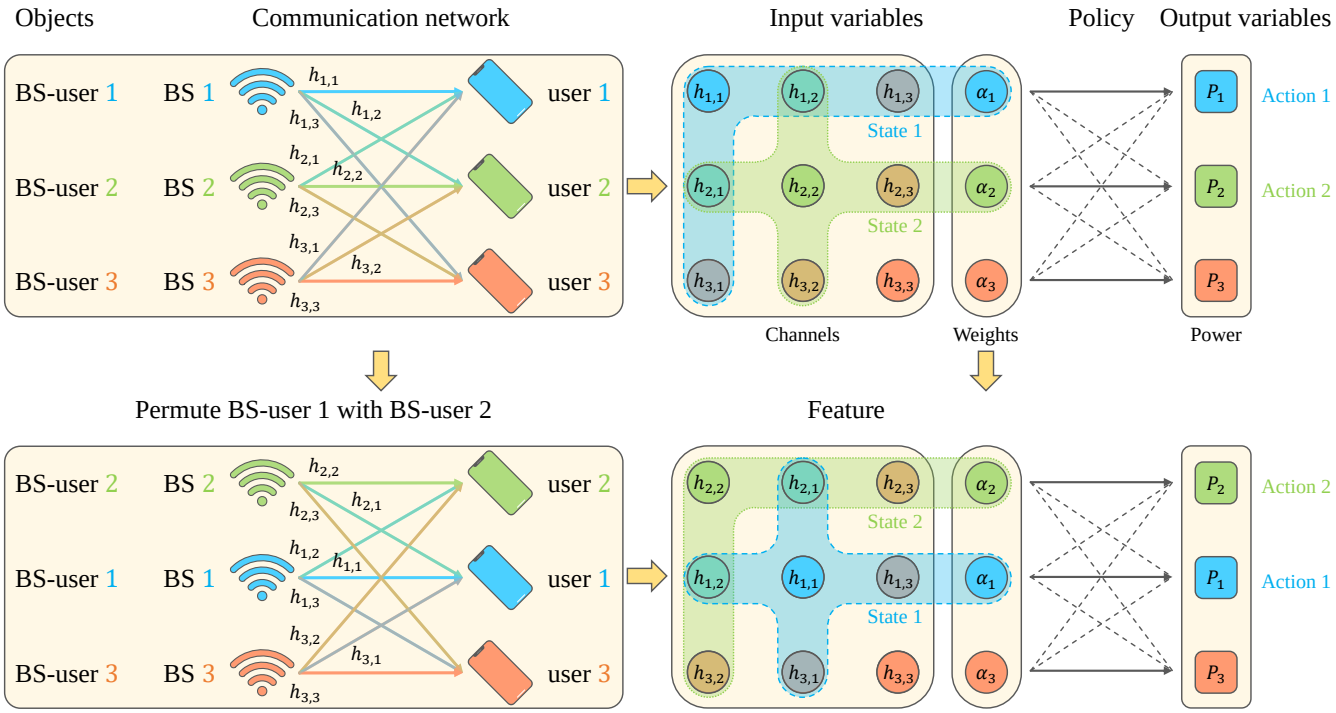

Fig. 3. Permutation equivariance of an interference coordination policy to BS-user pairs.

policy consists of the states of all objects and spans the state space, which is the same as the observation space.

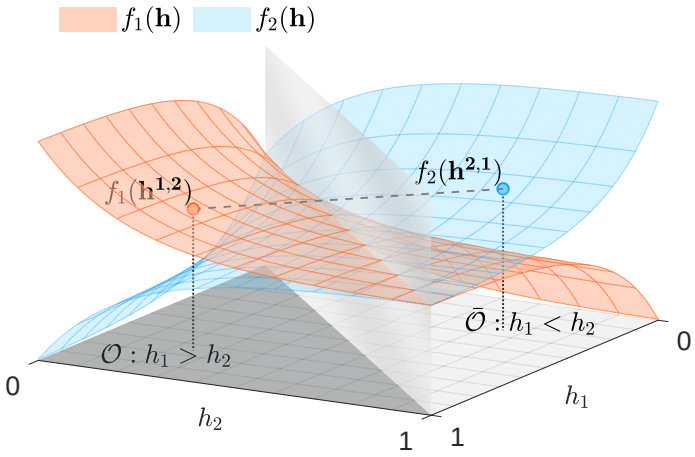

Fig. 4. Illustration of the training set compression. h_1 and h_2 are states of two objects, $\mathbf{h}^{1,2} = [h_1, h_2]$ and $\mathbf{h}^{2,1} = [h_2, h_1]$ are state vectors.

For a PE policy, the order information of the objects implicitly embedded in the training samples of the observation space is useless for seeking the multivariate functions. To remove the order information, we can jointly sort the states and labels in each training sample according to a specific order of the objects. For instance, the states can be sorted in a descending order, and the actions are sorted accordingly. In this way, the state vectors with same set of states arranged in different orders are mapped into a single feature vector, i.e., the useless information of the order is discarded.

In Fig. 4, we show two mirror symmetric (i.e., permutation equivariant) multivariate functions for two objects. The samples for the states satisfying $h_1 > h_2$, represented by the state vector $\mathbf{h}^{1,2}$, lie in the shadowed region $\mathcal{O}$. Other samples, represented by state vector $\mathbf{h}^{2,1}$, fall in the non-shadowed region $\bar{\mathcal{O}}$, each can find its symmetric point $\mathbf{h}^{1,2}$ in $\mathcal{O}$ by permutating the two states. Owing to the symmetric property, the responses of the functions in the non-shadowed region can be determined from the responses of the functions in the shadowed region, i.e., $f_2(\mathbf{h}^{2,1}) = f_1(\mathbf{h}^{1,2})$ and $f_1(\mathbf{h}^{2,1}) = f_2(\mathbf{h}^{1,2})$.

This suggests that only a half of each symmetric function in the halved observation space needs to be found. Since the state vectors only take values in the shrunken region, the distribution of the state vector changes after ranking. In particular, the distribution of the first element (and also the second element) of $\mathbf{h}^{1,2}$ and $\mathbf{h}^{2,1}$ is narrower than the probability distribution of the states h_1 and h_2.

When the number of objects approaches infinity, we can resort to the theory of order statistic [15] to explain why the training set can be compressed remarkably. For a random vector with large number of elements, if the elements are ranked in arbitrary (say descending) order, then the variance of an element in any given position of the random vector approaches zero [15].

Since after ranking, the random samples of each state are distributed in small region with high probability, the required training samples for learning a PE policy can be reduced. This implies an interesting phenomenon for a PE policy with scalar state for every object: "sample hardening", as to be

more clear from the case study.

IV. Case Study

In this section, we illustrate the gain in reducing sample complexity from ranking, when learning to optimize the three policies in section II with the DNNs trained by supervision.

A. Training and Testing the DNNs

The power allocation policy among subcarries is optimized to maximize the sum rate under the maximal transmit power constraint. The separation of subcarriers is 1 MHz, and the signal-to-noise ratio is 10 dB. The input of the DNN for learning the optimal policy is the effective noise power (i.e., the ratio between noise variance and channel gain) of the subcarriers. The labels used for training are obtained from solving the optimization problem by water-filling algorithm.

The interference coordination policy is optimized to maximize the sum rate under the maximal transmit power constraint. The system setup is the same as the Gaussian interference channel case with equal weights in [1]. The input of the DNN is the channel gains among all the BS-user pairs. The labels are the transmit powers of multiple BSs obtained from the weighted minimum mean square error (WMMSE) algorithm as in [1].

For both policies, the inputs of the DNNs are channel gains generated from Rayleigh distribution.

The caching policy is optimized to maximize the SOP, i.e., the probability that the data rate exceeds a threshold for a requested file cached at BSs. The system setup is the same as in [6] where 10% files are cached at each BS, except that here we consider homogeneous network. The input of the DNN is the popularity generated from Zipf distribution with the skewness parameter as 0.6. The labels are the caching probabilities optimized with the water-filling algorithm in [6].

Without ranking, the original sample used for training or testing each DNN consists of the input of the DNN and the corresponding label. Each sample for the DNN to learn the interference coordination policy can be expressed as a matrix, consisting of a channel matrix and a column vector of the actions.

With ranking, each sample used for training or testing the DNNs to learn the power allocation policy and the caching policy is obtained by sorting the original sample in a descending order according to the magnitudes of the input of the DNN. Each sample used for the DNN to learn the interference coordination policy is obtained by permuting both column and row of the matrix in the same manner (e.g., permute the first and second columns and permute the first and second rows at the same time, as shown in Fig. 3). We sort each sample according to the channel gains between BSs and their associated users (i.e., the diagonal values of the channel matrix) in descending order.

The DNNs are trained by minimizing the empirical mean square error between the label and the output. The hyper-parameters are tuned via a validation set, which is randomly generated as for the training set, and the number of samples is the minimal integer no smaller than 10% of the training set. The fine-tuned hyper-parameters are shown in Table I, where $[*]$ denotes that the number of neurons in one hidden layer is $*$, and $[*_1, *_2, *_3]$ denotes that the numbers of neurons in the first, second and third hidden layers are respectively $*_1$, $*_2$ and $*_3$. 1000 samples are used for testing each DNN.

B. Performance Comparison

We compare the system performance, sample complexity and the size of DNN of the following three learning methods.

- "No-rank": FC-DNN trained by the samples without ranking, which does not exploit any prior knowledge.
- "Rank": FC-DNN trained by the samples with ranking, which exploits the PE prior by data representation.
- "PENN": PENN trained by the samples without ranking, which exploits the PE prior by DNN structure.

For power allocation or interference coordination, the performance is the ratio of the sum rate achieved by the learning methods to the optimal solution or the solution obtained by the WMMSE algorithm. For probabilistic caching, the performance is the ratio of the SOP achieved by the learning methods to the solution obtained from the water-filling algorithm in [6].

Since only a few training samples are required for the power allocation or caching policy, we train the DNNs more than once to reduce the uncertainty caused by the random training set. For each time of training, the training and validation samples are randomly generated, while the test set is fixed. In particular, the sum rate of power allocation and the SOP of caching are obtained by selecting the second worst testing results from 10 well-trained DNNs, i.e., the results are with the confidence level of 90%.[1] Considering that in the case of 30 BS-user pairs the computing time consumed by training a PENN for interference coordination is too long, and the sum rate obtained from the well-trained PENNs may be low (say 0.6 or 0.7), the performance of interference coordination is only obtained from three well-trained DNNs by selecting the best testing results. Even though, the sum rate achieved by "PENN" is still lower than "Rank" and "No-rank", which cannot be improved by further increasing the numbers of samples and weights significantly according to our results (not shown for conciseness).

In Table II, we provide the system performance as well as the corresponding numbers of samples and free parameters for training each DNN to achieve (almost) the same performance. We can observe that "Rank" and "PENN" require much less samples and weights for training to achieve similar performance to "No-rank". Moreover, the number of training samples for "Rank" decreases with the number of objects. In particular, only three or five training samples are required for power allocation or caching with 30 objects, with the compression rate of $1350/3 = 450$ or $12000/5 = 2400$. In fact, our result shows that only one training sample is

[1]If we select the best one from the 10 inference results, then the PENN can also achieve the performance of 0.99.

TABLE I
FINE-TUNED HYPER-PARAMETERS FOR THE DNNS

Case		Power allocation			Caching			Interference coordination		
Number of objects		10	20	30	10	20	30	10	20	30
Number of neurons in Hidden layers	No-rank	[100]	[100]	[100]	[40]	[60]	[100]	[200, 80, 80] [1]		
	Rank	[10]	[5]	[5]	[10]	[5]	[4]	[150, 50, 50]		
	PENN	[100]	[200]	[300]	[100]	[400]	[300]	[100, 100]	[200, 200]	[300, 300]
Activation function of the hidden layer	No-rank	`ReLU`			`ReLU`			`ReLU` [1]		
	Rank									
	PENN									
Activation function of the output layer	No-rank	`Softplus`			`Sigmoid`			`ReLU6/6` [1]		
	Rank									
	PENN									
Learning algorithm	No-rank	$0.1/(1+0.001\times$iteration$)$			$1/(1+0.001\times$iteration$)$			RMSProp (Initial: 0.001) [1]		
	Rank									
	PENN	Adam (Initial: 0.001)			Adam (Initial: 0.01)			Adam (Initial: 0.01)		
Batch size	No-rank	32	32	32	32	32	32	1000 [1]		
	Rank	20	6	3	15	8	5			
	PENN	20	50	150	15	50	100	120	800	1000

TABLE II
COMPARISON FOR RANKING AND PENN.

Case		Power allocation			Caching			Interference coordination		
Number of objects		10	20	30	10	20	30	10	20	30
System performance	No-rank	0.9951	0.9995	0.9995	0.9907	0.9909	0.9905	0.9795	0.9063	0.8562
	Rank	0.9927	0.9975	0.9992	0.9900	0.9911	0.9926	0.9784	0.9042	0.8560
	PENN	0.9926	**0.9860**	**0.9770**	0.9925	0.9903	**0.9739**	**0.9003**	**0.8571**	**0.8418**
Number of training samples	No-rank	300	900	1350	5000	9000	12,000	500,000	1,000,000	1,000,000
	Rank	20	6	3	15	8	5	10,000	3000	2000
	PENN	20	50	150	15	50	100	120	800	4000
Number of weights for training	No-rank	2110	4120	6130	850	2480	6130	25,570	28,380	31,190
	Rank	220	225	335	220	225	274	12260	14070	16,080
	PENN	51	51	51	51	101	51	480	480	480

required for power allocation with 35 subcarriers! Such a surprising result comes from "sample harding" explained in section III-B. "PENN" is with very small mode size, but needs more training samples than "Rank" and achieves worse performance than the other two methods in most cases (see the boldfaced values).

The system performance of interference coordination in the table is relatively low, because we intend to fairly compare the sample complexity and DNN size of these methods. Our results show that the sum rate of "Rank" can be improved by using more samples for training, but "No-rank" and "PENN" cannot. For example, the performance of 0.99, 0.97 and 0.96 for the cases of 10, 20 and 30 BS-users pairs can be achieved by "Rank" with 200,000, 300,000 and 500,000 training samples, respectively, superior to the performance of "No-rank" trained with doubled or even much more samples.

V. CONCLUDING REMARKS

In this article, we discussed how to leverage a sort of prior knowledge inherent in many wireless tasks, permutation equivariance property, for reducing the sample complexity of DNNs. We showed that several representative wireless problems are permutation equivariant. We introduced a data representation method, ranking, to compress the training set for permutation equivariant policies, explained why training samples can be reduced, and found an interesting phenomenon of "sample hardening". The results in the case study showed that this simple method of ranking can achieve the same learning performance as FC-DNNs with much fewer training sample and free parameters. Moreover, the compression ratio increases with the number of objects, which suggests its potential for large scale wireless problems. Though this article considered the training of DNNs with supervision, the method of ranking is also applicable for the DNNs trained without labels and for other machine learning techniques. Despite of the promising gain, many problems remain open, say how to identify the permutation equivariance property as well as define objects and states in a wireless task, how to sort the samples when the state of each object is vector or matrix, and how to improve the learning

performance of PENN.

REFERENCES

[1] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: training deep neural networks for wireless resource management," *IEEE SPAWC*, 2017.

[2] X. Cao, R. Ma, L. Liu, H. Shi, Y. Cheng, and C. Sun, "A machine learning-based algorithm for joint scheduling and power control in wireless networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4308–4318, Dec. 2018.

[3] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, May 2019.

[4] P. Zhou, X. Fang, X. Wang, Y. Long, R. He, and X. Han, "Deep learning-based beam management and interference coordination in dense mmwave networks," *IEEE Trans. on Vehicular Technology*, vol. 68, no. 1, pp. 592–603, Jan. 2019.

[5] J. Xu, P. Zhu, J. Li, and X. You, "Deep learning-based pilot design for multi-user distributed massive MIMO systems," *IEEE Wireless Commun. Letters*, vol. 8, no. 4, pp. 1016–1019, Aug. 2019.

[6] J. Wu, C. Yang, and B. Chen, "Proactive caching and bandwidth allocation in heterogenous networks by learning from historical numbers of requests," *IEEE Trans. on Commun.*, vol. 68, no. 7, pp. 4394–4410, July 2020.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature Cell Biology*, vol. 521, no. 7553, pp. 436–444, May 2015.

[8] Z. Manzil, K. Satwik, R. Siamak, P. Barnabas, S. Ruslan, and S. Alexander, "Deep sets," *Advances in Neural Information Processing Systems*, 2017.

[9] H. He, S. Jin, C. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 77–83, Oct. 2019.

[10] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. on Commun.*, vol. 67, no. 10, pp. 7331–7376, Oct. 2019.

[11] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "LORM: Learning to optimize for resource management in wireless networks with few training samples," *IEEE Trans. on Wireless Commun.*, vol. 19, no. 1, pp. 665–679, Jan. 2020.

[12] D. Liu, C. Sun, C. Yang, and L. Hanzo, "Optimizing wireless systems using unsupervised and reinforced-unsupervised deep learning," *IEEE Network*, vol. 34, no. 4, pp. 270–277, July-Aug. 2020.

[13] J. Guo and C. Yang, "Structure of deep neural networks with a priori information in wireless tasks," *IEEE ICC*, 2020.

[14] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Trans. on Signal Processing*, vol. 68, no. 10, pp. 2977–2991, 2020.

[15] H. A. David and H. N. Nagaraja, *Order Statistics*. John Wiley and Sons, 2003.