Improving Learning Efficiency for Wireless Resource Allocation with Symmetric Prior

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

***Abstract*—Improving learning efficiency is paramount for learning resource allocation with deep neural networks (DNNs) in wireless communications over highly dynamic environments. Incorporating domain knowledge into learning is a promising way of dealing with this issue, which is an emerging topic in wireless community. In this article, we first briefly summarize two classes of approaches of using domain knowledge: introducing mathematical model or prior knowledge to deep learning. Then, we consider a kind of symmetric prior, permutation equivariance, which widely exists in wireless tasks. To explain how such a generic prior is harnessed to improve learning efficiency, we resort to ranking, which jointly sorts the input and output of a DNN. We use power allocation among subcarriers, probabilistic content caching, and interference coordination to illustrate the improvement of learning efficiency by exploiting the property. From the case study, we find that the required training samples to achieve given system performance decreases with the number of subcarriers or contents, owing to an interesting phenomenon: "sample hardening". Simulation results show that the training samples, the free parameters in DNNs and the training time can be reduced dramatically by harnessing the prior knowledge. The samples required to train a DNN after ranking can be reduced by $15 \sim 2,400$ folds to achieve the same system performance as the counterpart without using prior.**

***Index Terms*—Wireless communications, resource allocation, deep learning, training complexity, prior knowledge, permutation equivariance**

I. Introduction

Machine learning has been envisioned as one of the most important features of 6G [1,2], owing to its successful applications in a variety of complex tasks. In the past few years, deep learning has been widely applied in wireless communications [2–6]. The motivation is various, say improving spectral efficiency or user experience with low cost by providing real-time solutions for NP-hard optimization problems, handling the problems with inaccurate models or even without models, with well-trained deep neural networks (DNNs) [1,3,7].

DNN is a powerful tool in expressing complex functions due to its multilayer structure and non-linear neurons, which has achieved great success in many fields such as computer vision and natural language processing [8].

Existing research results in wireless communications have demonstrated that learning-based solution is promising in improving the usage efficiency of radio resources in spatial, temporal, frequency, and power domain, as well as cache and computing resources. Yet training a DNN also consumes network resources, which is not negligible for learning at wireless edge. To enable intelligent wireless networks with affordable overall expense, learning efficiency is becoming another key performance indicator.

Learning efficiency can be reflected by generalization ability and training complexity. Generalization ability concerns with the system performance over a test set that differs from the training set for a given amount of training resources, i.e., data, computing and storage resources. Training complexity concerns with the amount of training resources, i.e., the numbers of training samples and trainable parameters as well as the training time, required to achieve a desired system performance over the test set. These two metrics characterize the trade-off between the system performance in the inference phase and the resource consumption in the training phase.

Generalization ability has been extensively discussed in the literatures of machine learning and wireless communications. When used in static scenarios, DNNs can be trained offline, where training resources may be sufficient and hence are not a concern. However, as the development of wireless AI, machine learning has been introduced to mobile edges and expected to be applied in highly dynamic environment. For instance, radio resource allocation is often operated in time-varying fading channels, where training resources become the bottle-neck of the system performance. Whenever channels change, the DNN used for radio resource allocation has to be re-trained, and the training samples have to be re-gathered in a timely manner, otherwise the samples may be outdated [9]. Hence, training has to complete in a short time with a small number of samples.

To reduce the training complexity required to achieve a desired performance, a rational choice is trading off flexibility with complexity by resorting to domain knowledge. Domain knowledge is quite general, including mathematical models for relations (say Shannon formula), structures of iterative algorithms, assumptions, and properties of task functions to be learned. To exploit the knowledge, two distinctive while compatible classes of approaches have been proposed in wireless communications. One class is to incorporate principled models, the other is to integrate prior knowledge, into the learning process.

Model-based deep learning takes the advantages of both data-driven and model-driven methods. Data-driven methods are flexible and hence are applicable to broad range of tasks,

The source code for reproducing the results of this article is available at https://github.com/wjj19950101/Magrank-v1.

This work is supported by National Natural Science Foundation of China (NSFC) under Grant 61731002.

which however are with high training complexity and are non-interpretable. Model-driven methods strive to derive analytical expressions of resource allocation, by considering unique characteristics of a scenario without using any data. The resulting solutions are usually interpretable but only applicable to specific tasks. By tailoring the structure of a DNN to a scenario of interest, model-based deep learning simplifies the task function to be learned. This can be achieved by only learning a part of a task that is hard to model, or only learning an operation or a block in a task with high complexity [1]. By customizing DNNs for specific tasks, model-based deep learning may need very few samples [10] or with low computational complexity for training [11].

Prior knowledge is the information of a task function available before training a DNN for the task. The knowledge for the input-output relation of a task can be leveraged for guiding the learning procedure, say by adding a regularization term in the loss function for training a DNN or by designing the DNN structure. By constructing DNNs to satisfy a desired property, the functions without the property are excluded from the function family, which reduces training complexity. Two classical examples are convolutional neural networks and recurrent neural networks, which respectively exploit the knowledge of spatial and temporal translational invariance of a task [8]. Another notable example is a class of symmetric DNNs [12], which exploit a kind of prior knowledge broadly existed in wireless tasks [3–7,13,14]: permutation equivariance (PE).

Permutation equivariance is a kind of symmetric property of multivariate functions. It has been harnessed by constructing various DNNs with special structures, say permutation equivariant neural network (PENN) [12] and graph neural networks [14], for learning the policies with the property. These structures have been demonstrated capable of reducing the the numbers of training samples and trainable parameters of DNNs [13] and generalizing well to the problem scales (say the number of users) [14]. As a kind of relational prior, this property can also be used for data representation and data argumentation.

In this article, we attempt to interpret how the learning efficiency of DNNs for permutation equivariant policies can be improved from the perspective of training complexity. In order to explain why many wireless tasks exhibit such a property, we first introduce the notions of object and the state of each object of a task, and figure out several kinds of objects commonly existed in resource allocation. Given that interpreting DNNs is very hard, we explain the mechanism of reducing training complexity for tasks with symmetric property from the angle of data representation. In particular, we jointly sort the input and output variables of a policy, called ranking. By taking ranking as a tool, we reveal that the reduction of training complexity comes from a fact that a task function with PE property can be learned in a small feature space. In the cases where the states are scalars, we further find that the training samples required to achieve a given performance decrease dramatically with the number of objects, which comes from an interesting phenomenon: "sample hardening", i.e., the distribution of training samples is narrowed down after ranking.

The rest of this article is organized as follows. We first identify the PE property in wireless tasks. Then, we show how the property can be used to reduce sample complexity by ranking. Next, we provide case study to illustrate the potential of exploiting the property in reducing training complexity by comparing two approaches for exploiting the property, PENN and ranking, in the tasks of power allocation, caching and interference coordination, to the fully-connected DNN (FC-DNN) without using prior. Finally, we conclude the article and discuss the future works.

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 first explain why many tasks have the PE property by figuring out their latent inputs, objects. Then, we 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 policy to achieve a desired performance, which yields a solution for every impacting parameter. The policy is usually a set of multivariate functions, where the input variables are the impacting parameters and the output variables are the solution. Representative tasks include (but not limited to) resource allocation in spatial-temporal-frequency-power domain [3, 5, 13, 14], transceiver design [4], uplink/downlink channel calibration [7], 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. The state or the action of each object itself can be multi-valued and 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), 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 [12]. 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 states. That is to say, a wireless policy for a set of objects will be permutation equivariant to the states, if the states are appropriately selected.

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

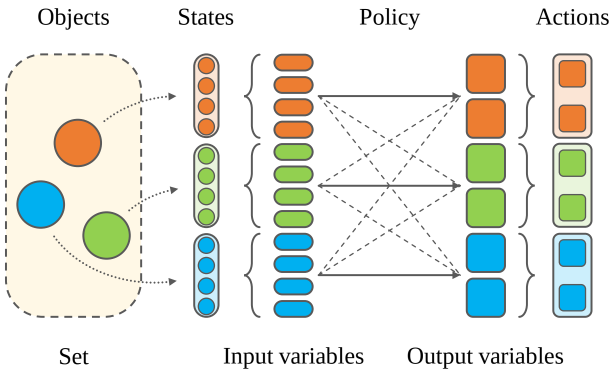

Fig. 1. Relation between key components of a policy.

B. Several Examples of Resource Allocation

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

1) Power Allocation among Subcarriers: We first consider a classical water-filling power allocation policy in a single user multi-subcarrier system. The policy is a set of multivariate functions, where the input and output variables are respectively the channel gains and transmit powers at the subcarriers. The policy is permutation equivariant since its response is indifferent to the ordering of the subcarriers. A subcarrier is an object, the channel gain is the state of the object, and the transmit power allocated to the subcarrier is 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 change in the same way whereas the policy remains unchanged.

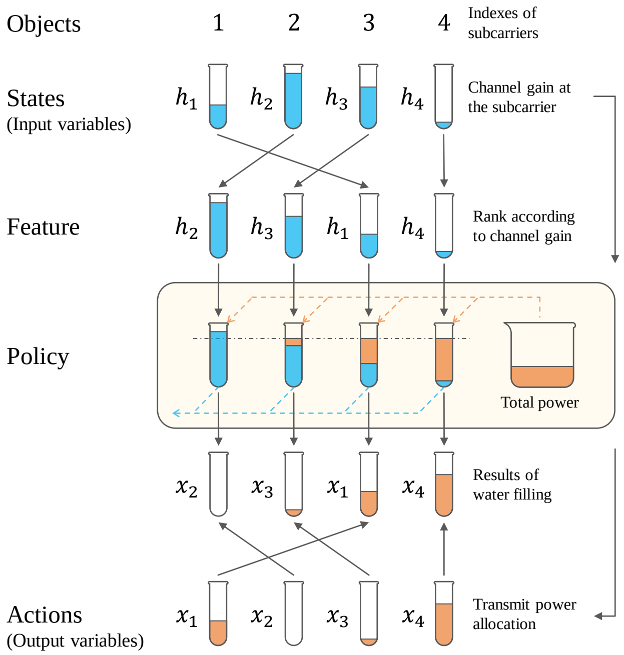

Fig. 2. Permutation equivariance of a power allocation policy.

2) Interference Coordination: Another typical example is interference coordination. For easy exposition, consider a scenario in [3], where each single-antenna base station (BS) serves a single-antenna user, and the interference is mitigated by controlling the transmit power of each BS to maximize a weighted sum rate. The policy is a set of multivariate functions, where the input variables are the channels from all BSs to all users, and the output variables are the transmit powers at the BSs to their associated users. A BS and its associated user is an object, and the transmit power at the BS is the action of the object. The action relies on the channels from the BS to all users, the channels from all BSs to the user, and the weight on the data rate, as shown in Fig. 3. Hence, these channel gains and the weight are 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 of an object does not contain the weight on the data rate 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 systems, the performance such as successful offloading probability (SOP) can be improved [6] by optimizing probabilistic caching policy based on future file popularity. The policy is a set of functions, where the input variables are the popularity of all files, and the output variables are the caching probabilities for files. A file is an object, the future popularity of the file is the state, and the future caching probability for the file is the action.

III. How Training Complexity is Reduced?

In this section, we first introduce a toy example to show the intuition of reducing training complexity for learning symmetric function. Then, we resort to ranking to explain how the training complexity is reduced for the PE policies.

To find a policy for a task with supervised learning, a DNN is trained with the samples each consisting 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., fully-connected DNN (FC-DNN), from original data. When prior knowledge is available, fewer training samples are required either by designing the DNN structure [8, 12], or by mapping the observation space where the data is gathered to the feature space where the samples are used.

A. Reducing Training Complexity with Symmetric Prior

A common practice in deep learning is to directly take the observed data as the feature for training. To understand why the training complexity for learning a task with symmetric property can be decreased by transforming observations to features, we provide a toy example.

Consider a task of learning an axial symmetric function, say quadratic function, with labeled training samples, where

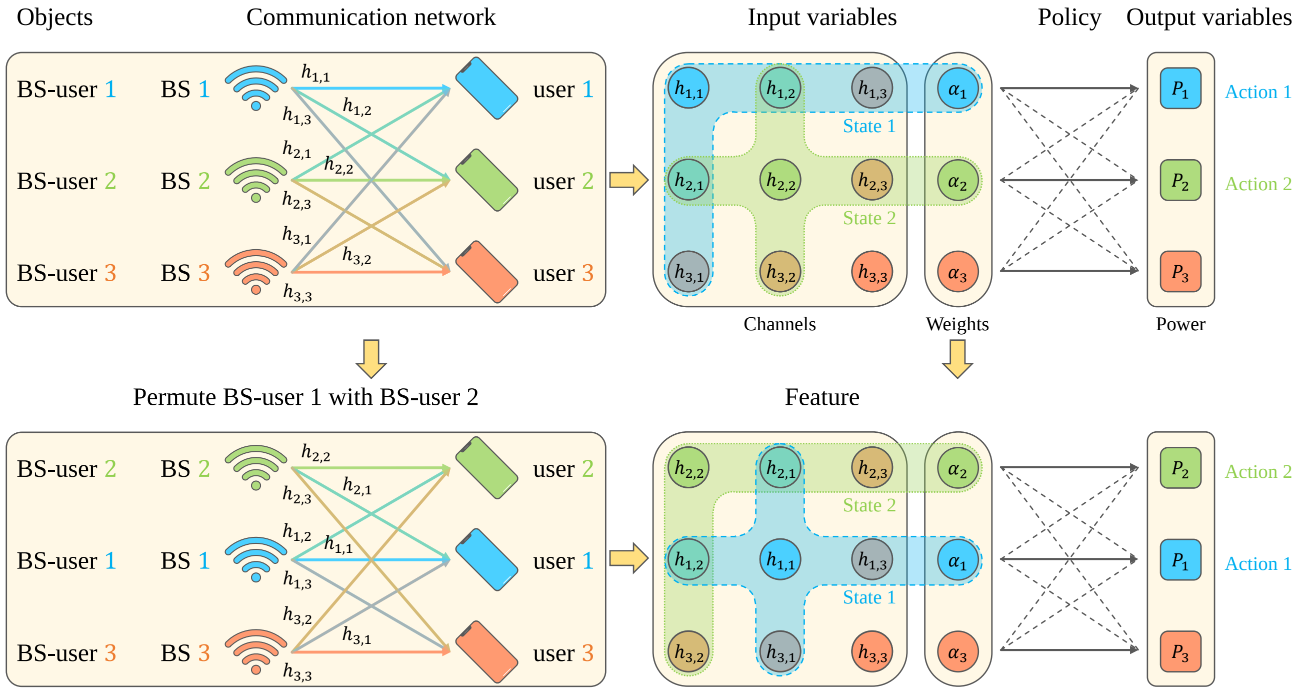

Fig. 3. Permutation equivariance of an interference coordination policy.

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. After transforming each real-valued observation into a positive-valued feature, the training set is halved. Using the training samples each only containing positive input variable and the corresponding response, the function can be learnt with low complexity without sacrificing the learning performance.

B. Ranking and Sample Hardening

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

For a PE policy, the order information of the objects implicitly embedded in the samples is useless for seeking the multivariate functions. To remove the order information, we can jointly sort the states and the labels in each sample according to a ranking rule that depends on the states. For instance, the states can be sorted in a descending order, and the actions are sorted accordingly. In this way, the state vectors with a same set of states arranged in different orders are mapped into a single feature vector, and so are the corresponding label vectors.

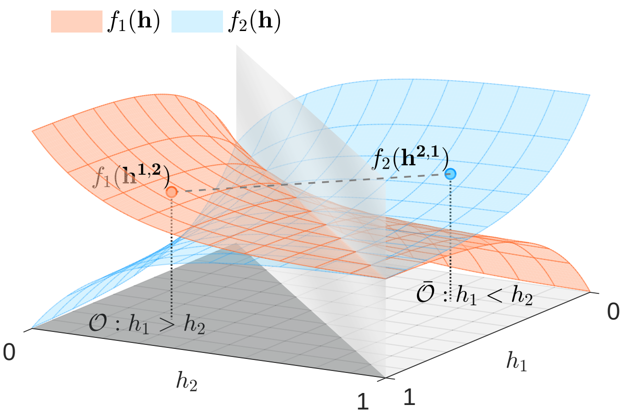

Fig. 4. Illustration of mirror-symmetric functions. h_1 and h_2 are states of two objects, $\mathbf{h}^{1,2} = [h_1, h_2]$ and $\mathbf{h}^{2,1} = [h_2, h_1]$ are state vectors.

In Fig. 4, we show two permutation equivariant multivariate functions for two objects, where each object has a single-valued state. The samples for the states satisfying $h_1 > h_2$, represented by a 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 swapping 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.

This suggests that only a half of each symmetric function needs to be found in the halved observation space. Since the task function to be learned is simplified, the training complexity of the DNN can be reduced.

Moreover, the distribution of training samples is changed by ranking, since the state vectors only take values in the shrunken region. In particular, the distribution of the first element (and also the second element) of the two state vectors is narrower than the probability distribution of the non-ranked states. According to the theory of order statistic [15], the variance of an element in any given position of a random vector with large number of elements approaches zero if the elements are ranked in ascending or descending order. This implies that ranking leads to "sample hardening" for a PE policy with scalar state for every object. As a consequence, the required training samples for learning a PE policy can be decreased dramatically when the number of objects is large.

When the state is multi-valued (say for the interference coordination policy), ranking is still able to reduce the sample complexity. However, finding a good metric that maps the multiple values in the state to a real number for ranking is challenging, which requires domain knowledge.

IV. Case Study

In this section, we illustrate the gain in reducing training complexity from harnessing the PE property, by learning to optimize the three policies in section II with DNNs trained by supervision.

The training complexity includes the number of free parameters in each DNN, as well as the minimal number of samples and the computing time required for training each DNN to achieve (almost) the same system performance. All results are obtained with TensorFlow 1.14.0 on a computer with Intel®-Core™-i7-8700K CPU and a single NVIDIA®-GeForce-GTX™-1080-Ti GPU.

A. System Setups and Data Generation

The power allocation policy is optimized to maximize the sum rate of all subcarriers under the total transmit power constraint. The separation of subcarriers is 1 MHz, and the signal-to-noise ratio is 10 dB. The input of the DNNs consists of the channel gains of all subcarriers, whose samples are generated from Rayleigh distribution. The output of the DNNs consists of the powers allocated to all subcarriers, whose labels are obtained from the classical water-filling algorithm.

The interference coordination policy is optimized to maximize the sum rate of all users under the maximal transmit power constraint. The system setup is the same as the Gaussian interference channel case with equal weights in [3]. The input of the DNNs consists of the channel gains among all BS-user pairs, whose samples are generated from Rayleigh distribution. The output of the DNNs consists of the transmit power of all BSs, whose labels are obtained from the weighted minimum mean square error (WMMSE) algorithm as in [3].

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 DNNs consists of the future popularities of all files, whose samples are generated from Zipf distribution with the skewness parameter as 0.6. The output of the DNNs consists of the caching probabilities of all files, whose labels are obtained from the water-filling algorithm in [6].

B. Performance Comparison

We compare the performance of each policy learned by the following three DNNs.

- "W/o-Prior": 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.

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

With ranking, each sample used for training or testing the DNNs to learn the power allocation and the caching policies is obtained by sorting the input and output vectors of 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). Given that each state is composed of multiple channel gains, 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, for an illustration.

The generated data set is divided into training set, validation set and test set. The size of the training set is not fixed in order to evaluate the sample complexity. The size of the validation set is the minimal integer no smaller than 10% of the size of the training set, and the test set contains $1,000$ samples. The DNNs are trained by minimizing the empirical mean square error between the labels and the outputs on the training set. The training samples are divided into batches. Each batch is used to compute the gradient for updating the trainable parameters in DNNs in each iteration. To exploit the limited samples, the training set is repeatedly used for multiple times, counted in epochs. The hyper-parameters are fine-tuned on the validation set and are shown in Table I. Here, $[*_1, *_2, \cdots]$ denotes there are $*_1$ neurons in the 1st hidden layer, $*_2$ in the 2nd and so on, and the activation functions are the non-linear functions used in the neurons.

In what follows, we evaluate the system performance achieved by each resource allocation policy learned by each well-trained DNN on the test set and the corresponding training complexity.

TABLE I
FINE-TUNED HYPER-PARAMETERS

Case		Power allocation			Caching			Interference coordination		
Number of objects		10	20	30	10	20	30	10	20	30
Number of neurons in hidden layers	W/o-Prior	[100]	[100]	[100]	[50]	[90]	[120]	[200, 80, 80] [3]		
	Rank	[10]	[5]	[5]	[20]	[5]	[4]	[150, 50, 50]		
	PENN	[100]	[200]	[300]	[100]	[400]	[300]	[100, 100]	[200, 200]	[300, 300]
Activation function of the hidden layers	W/o-Prior	ReLU			ReLU			ReLU [3]		
	Rank									
	PENN									
Activation function of the output layer	W/o-Prior	Softplus			Sigmoid			ReLU6/6 [3]		
	Rank									
	PENN									
Learning algorithm	W/o-Prior	Adam (Initial: 0.1)			Adam (Initial: 0.01)			RMSProp (Initial: 0.001) [3]		
	Rank									
	PENN	Adam (Initial: 0.001)						Adam (Initial: 0.01)		
Batch size	W/o-Prior	32	32	32	128	128	128	1,000 [3]		
	Rank	20	6	3	15	8	5			
	PENN	20	50	150	15	50	100	120	800	1,000
Epochs	W/o-Prior	3,000	3,000	3,000	3,000	3,000	3,000	300	300	300
	Rank	3,000	3,000	3,000	10,000	10,000	10,000	500	500	500
	PENN	10,000	10,000	15,000	10,000	10,000	15,000	500	3,000	8,000

For power allocation or interference coordination, the system 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 system 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 show the impact of the random training set. For each time of training, the training and validation samples are randomly generated, while the test set is fixed. The sum rate of the power allocation policy and the SOP of the caching policy are obtained by selecting the second worst testing results from 10 well-trained DNNs, hence the results are with the confidence level of 90%. Because in the case of 30 BS-user pairs the computing time for training a PENN for interference coordination is too long, and the sum rate achieved by a PENN may be low (say 0.6 or 0.7), the performance of the interference coordination policy is obtained only 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 "W/o-Prior", which cannot be improved by further increasing training samples and free parameters in the DNN according to our results.

The performance is provided in Table II. We can observe that "Rank" and "PENN" require much less training samples and free parameters to achieve similar performance to "W/o-Prior". Moreover, the training samples required by "Rank" decreases with the number of objects. In particular, only three or five training samples are required for the power allocation policy or caching policy with 30 objects, with a compression rate of $1,350/3 = 450$ or $12,000/5 = 2,400$ with respect to "W/o-Prior". In fact, our result shows that only one training sample is required for the power allocation policy to achieve the system performance of 0.99 when there are 35 subcarriers! Such a surprising result comes from the "sample hardening" phenomenon mentioned in section III-B. "PENN" is with much fewer free parameters, but needs more training samples and training time than "Rank" and performs worse than the other two methods in most cases (see the boldfaced values).

For interference coordination, the system performance in the table is lower, because we intend to fairly compare the training complexity of the DNNs. Our results show that the sum rate achieved by "Rank" can be improved by using more samples for training, but "W/o-Prior" and "PENN" cannot. For example, the system performance of 0.99, 0.97 and 0.96 can be achieved by "Rank" for the cases of 10, 20 and 30 BS-user pairs with 200,000, 300,000 and 500,000 training samples, respectively, which is superior to "W/o-Prior" trained with doubled or even much more samples.

In Table I, the number of neurons in hidden layers differs for different DNNs. Our further results show that the sample complexity can still be reduced significantly by exploiting the PE property when they are identical (say 300 neurons are used in the hidden layers for three DNNs in the case of 30 subcarriers or files).

V. CONCLUDING REMARKS

In this article, we discussed how to leverage PE property inherent in many resource allocation policies for improving the learning efficiency of DNNs. We identified several representative wireless policies that are permutation equivariant and explained why they exhibit such a property. We interpreted

TABLE II
COMPARISON OF SYSTEM PERFORMANCE AND TRAINING COMPLEXITY.

Case		Power allocation			Caching			Interference coordination		
Number of objects		10	20	30	10	20	30	10	20	30
System performance	W/o-Prior	0.9948	0.9992	0.9999	0.9904	0.9909	0.9912	0.9795	0.9063	0.8562
	Rank	0.9943	0.9951	0.9999	0.9900	0.9903	0.9915	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**
Training samples	W/o-Prior	300	900	1,350	5,000	9,000	12,000	500,000	1,000,000	1,000,000
	Rank	20	6	3	15	8	5	10,000	3,000	2,000
	PENN	20	50	150	15	50	100	120	800	4,000
Free parameters	W/o-Prior	2,110	4,120	6,130	1,060	3,710	7,350	25,570	28,380	31,190
	Rank	220	225	335	430	225	274	12,260	14,070	16,080
	PENN	51	51	51	51	101	51	480	480	480
Training time in seconds	W/o-Prior	21.64	61.2	89.01	90.96	165.61	225.78	574.10	4,198.18	9,844.96
	Rank	5.45	5.51	5.52	18.14	18.19	18.48	16.88	16.52	24.02
	PENN	31.74	32.05	50.32	31.70	32.02	50.29	6.28	219.16	6,737.43

why PE policies can be learnt with low training complexity by using ranking, which not only simplifies the task function for a DNN to learn but also changes the input distribution. The results in the case study showed that ranking can achieve the same system performance as the counterpart without using prior with much lower training complexity, which is more pronounced in reducing sample complexity for systems with large number of objects due to the "sample hardening".

Though this article considered the training of DNNs with supervision, both ranking and PENN are also applicable for the DNNs trained without labels and for other machine learning techniques. To achieve the promising gain, many issues remain open, say how to identify the objects and states of a PE policy, how to sort the samples when each object has more than one states, and how to improve the system performance of PENN and reduce the computing time for training PENN.

REFERENCES

[1] L. Liang, H. Ye, G. Yu, and G. Y. Li, "Deep-learning-based wireless resource allocation with application to vehicular networks," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 341–356, Feb. 2020.

[2] F. Restuccia and T. Melodia, "Deep learning at the physical layer: System challenges and applications to 5G and beyond," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 58–63, Oct. 2020.

[3] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: training deep neural networks for interference management," *IEEE Trans. on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.

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

[5] 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.

[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] 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.

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

[9] J. Zhang, C. Sun, and C. Yang, "Resource allocation in URLLC with online learning for mobile users," *IEEE VTC Spring*, 2021.

[10] 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.

[11] Q. Hu, Y. Cai, Q. Shi, K. Xu, G. Yu, and Z. Ding, "Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser MIMO systems," *IEEE Trans. on Wireless Commun.*, p. early access, 2020.

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

[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.

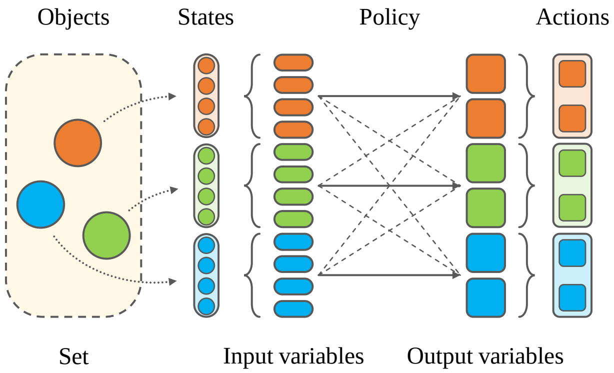

Fig. 5. Relation between key components of a policy.

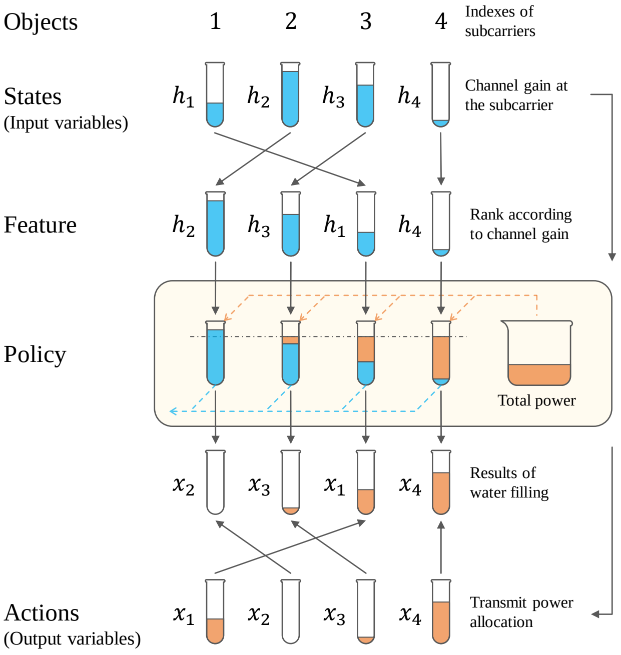

Fig. 6. Permutation equivariance of a power allocation policy.

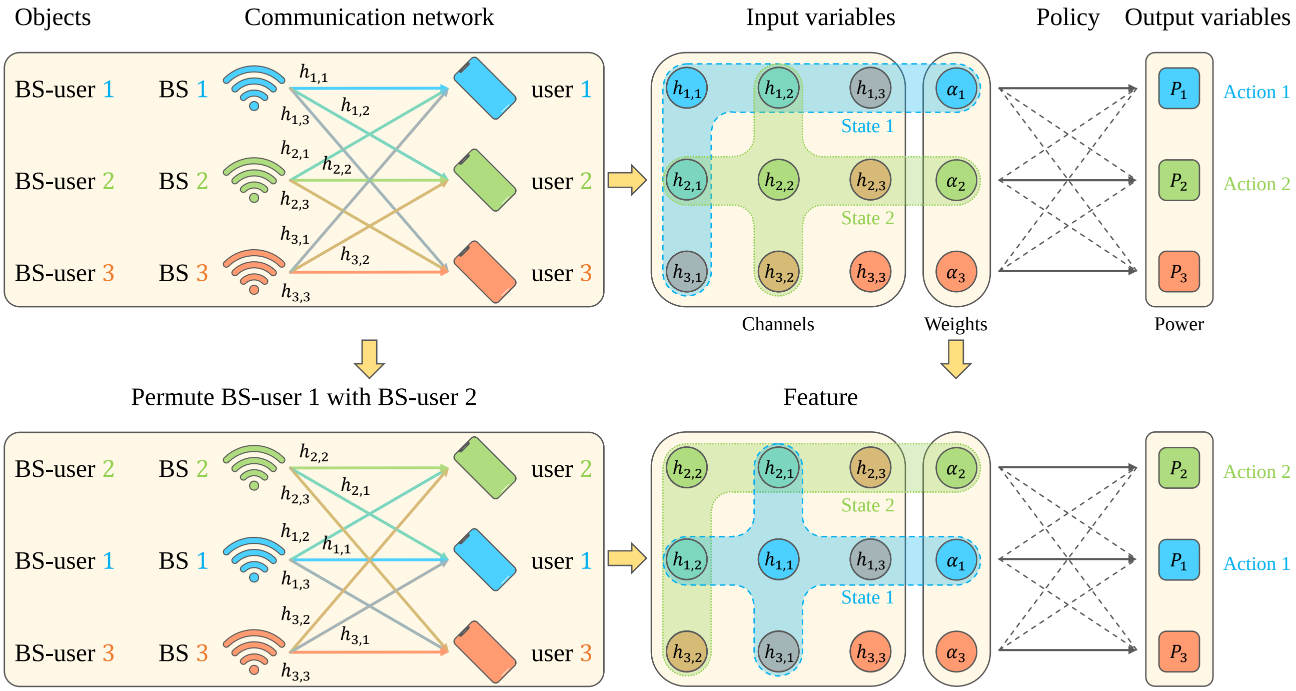

Fig. 7. Permutation equivariance of an interference coordination policy.

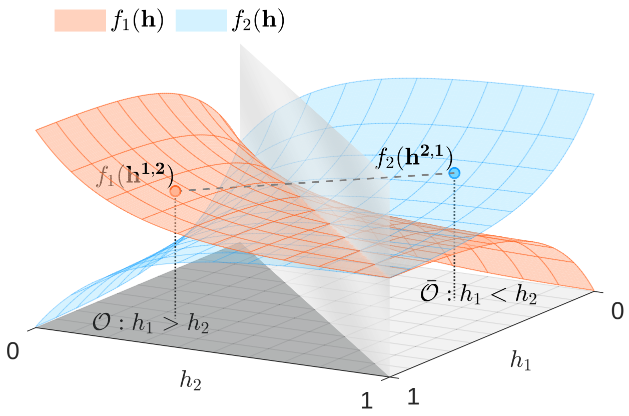

Fig. 8. Illustration of mirror-symmetric functions. h_1 and h_2 are states of two objects, $\mathbf{h}^{1,2} = [h_1, h_2]$ and $\mathbf{h}^{2,1} = [h_2, h_1]$ are state vectors.