Master Thesis

# Deep Learning-aided Resource Allocation for Wireless Communication Systems

Taoyu Yang

**Institute for Digital Communications**
Prof. Dr.-Ing. Robert Schober
University of Erlangen-Nuremberg

Supervisors: M.Sc. Dongfang Xu, Dr. Ing. Yan Sun, and Dr. Ing. Vahid Jamali

December 20, 2019
revised on May 1, 2020


Institute for Digital Communications

Prof. Dr.-Ing. Robert Schober
University of Erlangen-Nürnberg

# Master Thesis

## Deep Learning-aided Resource Allocation for Wireless Communication Systems for Taoyu Yang

Recently, deep learning has received much attention as a promising approach to optimize resource allocation for wireless communication systems. Facilitated by recent technological advances and the capabilities of specialized hardware for data processing, deep learning can be employed to solve resource allocation optimization problems in a faster manner compared to conventional approaches (e.g., monotonic optimization, successive convex approximation). A powerful and universal approach to deep learning are artificial neural networks (ANNs).

Deep learning based on ANNs employs a data-driven approach in order to identify the best ANN architecture given a training set of input-output data pairs. Once the ANN is trained, it is capable of responding to never-observed inputs by providing the optimum output given the empirical data and prior knowledge. The objective of this thesis is to train an ANN model to solve resource allocation optimization problems in wireless communication systems. In particular, the considered problem is first transformed into an approximate mathematical model and solved by employing conventional optimization methods. The ANN model is trained based on the resulting solutions and then refined for the actual system including effects that were not accounted for in the approximate model such as amplifier nonlinearities, model approximation errors, etc. The trained ANN model is then used to solve new resource allocation optimization problems for wireless communication systems.

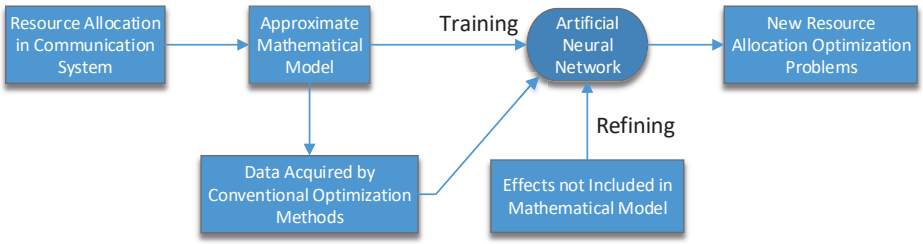


Figure 1: ANN model consisting of training phase and refining phase.

**Main guidelines for the work:**

- Acquisition of basic knowledge in communications and deep learning theory
- Optimization and refinement of an ANN model based on acquired data and prior knowledge
- Utilization of the trained ANN model to solve new resource allocation optimization problems and presentation of results via simulation

---

SUPERVISORS Dongfang Xu, Yan Sun, and Vahid Jamali
{dongfang.xu, yan.sun, vahid.jamali@fau.de}

Student: Taoyu Yang
Start date: Jun. 20, 2019
Duration: 6 months

Prof. Dr.-Ing. Robert Schober
Lehrstuhl für Digitale Übertragung

# Declaration

To the best of my knowledge and belief this work was prepared without aid from any other sources except where indicated. Any reference to material previously published by any other person has been duly acknowledged. This work contains no material which has been submitted or accepted for the award of any other degree in any institution.

Erlangen, December 20, 2019

Taoyu Yang
Drausnickstr, 6,
Erlangen, Germany

# Contents

# Abstract

Resource allocation in wireless communication networks has attracted much attention in both academia and industry. In this thesis, we study resource allocation algorithm design for a multiuser downlink communication system in the presence of eavesdroppers. We aim to minimize the total transmit power of the considered system by jointly optimizing the artificial noise and downlink beamforming taking into account the quality-of-service constraint of the users and the maximum information leakage tolerance of the eavesdropper. The resulting convex optimization problem can be solved by applying the semidefinite programming method. However, the bottleneck of this approach is the computational complexity. In fact, with the increase of users and antennas, we may not get the result in time. For solving the optimization problem in a dynamic environment, we must guarantee both time efficiency and accuracy. Based on these requirements, we propose a fully connected neural network model to facilitate machine learning-based resource allocation design. To obtain satisfying results via the proposed approach, we use the input and output of the semidefinite programming method as the feature values and labels of the training samples. Once the training is completed, the machine learning-based model can yield a decent result in a very short time. This model can fully meet the requirements of real-time communication. Furthermore, we take into account some limitations of machine learning and focus on how to effectively extract useful network features. We propose a convolutional neural network model based on deep learning and develop a specific training algorithm for the model. Then, through detailed mathematical demonstration and simulation experiments, we compare and analyze the predicted results of the two models. The experimental result shows that both schemes achieve a good system performance and the convolutional model can solve more general parametric problems in practice.

# Glossary

## Abbreviations

| | |
|---|---|
| **SDP** | Semidefinite Programming |
| **QoS** | Quality-of-Service |
| **LP** | Linear Programming |
| **QP** | Quadratic Programming |
| **FCN** | Fully Connected neural Network |
| **CNN** | Convolutional Neural Network |
| **1G** | First generation of communication systems |
| **5G** | Fifth generation of communication systems |
| **AN** | Artificial Noise |
| **PHY** | Physical Layer |
| **BS** | Base Station |
| **AWGN** | Additive White Gaussian Noise |
| **SINR** | Signal-to-Interference-plus-Noise Ratio |
| **MUI** | Multiuser Interference |
| **BF** | Beamforming |
| **DNN** | Deep Neural Network |

## Operators

| | |
|---|---|
| $\mathbb{C}$ | The set of all complex numbers |
| $\mathscr{E}\{x\}$ | Statistical expectation of random variable $x$ |
| $\forall x$ | Means that a statement holds for all $x$ |
| $\mathbb{H}^N$ | The set of all $N \times N$ Hermitian matrices |
| $\mathrm{Tr}(\mathbf{A})$ | The trace of matrix $\mathbf{A}$ |
| $\mathbf{A} \succeq 0$ | Means matrix $\mathbf{A}$ is positive semidefinite |
| $\lvert x \rvert$ | Absolute value of a complex scalar $x$ |
| $sign(x)$ | Calculate the sign of x |
| $\mathscr{CN}(\mu, \mathbf{R})$ | The circularly symmetric complex Gaussian distribution with mean $\mu$ and covariance matrix $\mathbf{R}$ |

## Symbols

| | |
|---|---|
| $d_k$ | Information bearing signal to active use $k$ |
| $\Gamma_k$ | Received SINR at active user $k$ |
| $\Gamma_{m,k}^E$ | Received SINR at idle user $m$ for eavesdropping active user $k$ |
| $\Gamma_{\mathrm{req}}$ | Minimum required SINR for active users |

| | |
|---|---|
| $\mathbf{h}_k$ | Channel vector between the base station and active user $k$ |
| $K$ | Number of active users |
| $\mathbf{l}_m$ | Channel vector between the base station and idle user $m$ |
| $M$ | Number of idle users |
| $N_T$ | Number of antennas at the base station |
| $P_{\max}$ | Maximum transmit power at the base station [dBm] |
| $\mathbf{v}$ | Artificial noise vector |
| $\mathbf{V}$ | Artificial noise covariance matrix |
| $\mathbf{w}_k$ | Beamforming vector for active user $k$ |
| $\mathbf{W}$ | Beamforming matrix |
| $w_{kn}$ | The $n$-th element of beamforming vector for active use $k$ |
| $\mathbf{x}$ | Combined transmit signal vector |
| $\mathbf{x}_k$ | Transmit signal to active user $k$ |
| $y_k$ | Received signal at active user $k$ |
| $y_m^E$ | Received signal at idle user $m$ |

# Chapter 1

# Introduction

Resource allocation optimization is a pivotal issue in radio frequency communication. Since the First generation of communication systems (1G) developed in the 1980s, the mobile communication industry has continued to develop, innovate and usher in technological upgrades. Even though Fifth generation of communication systems (5G) is currently not fully utilized in our daily [1], beyond 5G techniques has attracted much attention from both academia and industry [2], [3]. The era of big data in mobile communications has brought an unprecedented mobile communication experience, but at the same time, it has also caused the problem of processing large amounts of mobile data. Because all users receive dynamic information signals from the base station, we need to achieve dynamic and fast resource allocation under the requirements of ensuring the quality and security of communication signals.

Recently, resource allocation algorithm design has attracted much attention for wireless communication networks. In fact, if the considered optimization problem is convex, we can obtain the optimal solution by employing convex optimization techniques, i.e., [4]–[6]. In particular, convex programming contains many problem classes such as Linear Programming (LP) [7], Quadratic Programming (QP) [8], Semidefinite Programming (SDP) [9]. However, even though there exist many excellent convex optimization methods, there is still much space for improvement from a practical perspective. For example, when facing a large amount of data, even if it is a LP problem, the required amount of calculation is very large. As the amount of network data continues to increase at a high speed, this makes the network structure, especially the data center or base network structure, which face a large number of complex network business flows. Also, there are some problems that can be equivalently transformed into convex optimization. For example,the objective function of linear fractional programming [10], can be transformed into LP by a Charnes-Cooper transformation. On the other hand, there are also many non-convex problems in resource allocation algorithm design for wireless communications. In general, for these problems, the optimal solutions can only be ob-

tained with an exponentially computational complexity [11]–[14]. Moreover, to strike a balance between optimality and computational complexity, we can obtain a locally optimal solutions either by employing successive convex approximation [15], [16], or solving the considered problems in an alternating manner [17]–[21].

Based on the above background and problems, in this thesis, we choose the machine learning method [22] to solve resource allocation optimization problems and proposed an optimization model for network resource allocation based on Fully Connected neural Network (FCN) and Convolutional Neural Network (CNN), respectively. The advantages of these two models are compared with the SDP method and the applicable range of different methods is discussed. Since invented in the 1960s, machine learning [23] has undergone several technical upgrades. Thanks to the increase in computing power, its advantages in dealing with many practical problems have gradually emerged. Deep learning is an extended branch of machine learning. It is developed from the multi-layer perceptron [24]. Deep learning aims to build a neural network model with multiple hidden layers. In theory, a multi-layer perceptron model with a single hidden layer, as long as there are a sufficient number of hidden neurons, can approximate the value of any non-linear function. However, in terms of model complexity, adding hidden layers is more effective than increasing the number of neurons with a single hidden layer. Because adding the hidden layer increases not only the number of neurons but also the number of layers of activation function nesting in the neuron. If you use a neural network model with multiple hidden layers, you can learn higher-level abstract features of the data. Through multi-layer mapping from input to output, deep learning algorithms can learn different abstract features of the data layer by layer. Because the number of layers is deep, it can learn very complex non-linear functions. Through the neural network model with multiple hidden layers, the multi-level features of the input data are mined to obtain better prediction results. For traditional statistical methods, the characteristics of the data cannot be processed. Therefore, deep learning that has advantages in the face of large data and contains noise is selected.

The rest of the thesis is structured as follows. In chapter 2, we present the considered wireless communication model and discuss how to get optimization by SDP. In chapter 3, we briefly introduce some background of machine learning, give the training model and process of FCN. In chapter 4, based on the limitation of FCN, we present a deep neural network model and give the mathematical model of deep learning. In chapter 5, simulation results and corresponding analysis are presented. Chapter 6 includes the conclusion and future work.

# Chapter 2

# Multiuser System Model

In this chapter, we first introduce a multiuser wireless communication system model. In Section 2.1, the structure of the entire system and the function of each module are explained. Each parameter is specifically introduced, the process of signal transmission between the base station and each user is described in detail. In Section 2.2, we consider the performance matrix. In Section 2.3 we formulate resource allocation optimization problem and explain how to solve the optimization problem by the SDP method.

## 2.1 A Multiuser Wireless Communication System Model

We consider a multiuser wireless communication network practical applications, cf. Figure 2.1. The system includes a Base Station (BS), $K$ active users and $M$ idle users. The BS has more than one antenna, all active and idle users are single-antenna devices and have the ability to receive and decode signals. At the beginning of each time slot, active users can receive information including all user signals from the BS, while idle users who remain silent can also obtain transmission signals. This shows that idle silent users can also become eavesdroppers. Therefore, in order to ensure the security of signal transmission in the entire communication system, we assume that all idle users are eavesdroppers and analyze all resource allocation issues under this constraint.

At the beginning of each time slot, the BS transmits a signal stream including $K$ independent information signals to $K$ active users. Especially, the transmit signal to active user $k \in \{1,\ldots,K\}$ can be expressed as

$$\mathbf{x}_k = \mathbf{w}_k d_k, \tag{2.1}$$

where $d_k \in \mathbb{C}$ and $\mathbf{w}_k \in \mathbb{C}^{N_T \times 1}$ are the information bearing signal to active use $k$ and the corresponding Beamforming (BF) vector. We can assume $\mathscr{E}\left\{|d_k|^2\right\} = 1$, $\forall k \in \{1,\ldots,K\}$, without loss of generality. Moreover, we apply Artificial Noise (AN) technique to im-

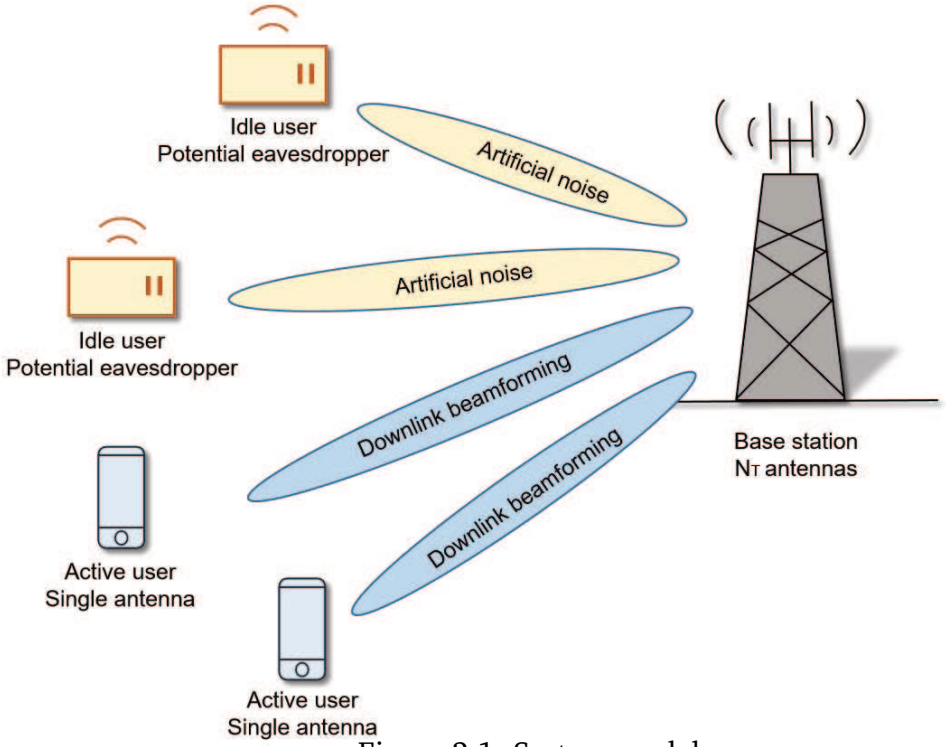


Figure 2.1: System model

prove the Physical Layer (PHY)-security of the considered system. Hence, the transmit signal vector, including $K$ information signals and AN, is given by

$$\mathbf{x} = \sum_{k=1}^{K} \mathbf{x}_k + \mathbf{v}, \tag{2.2}$$

where $\mathbf{v} \in \mathbb{C}^{N_T \times 1}$ are the AN vector generated by the BS to intentionally degrade the channels of the $M$ potential eavesdroppers. In this thesis, we model AN as a complex Gaussian distributed variable where $\mathbf{v} \sim \mathscr{CN}(\mathbf{0}, \mathbf{V})$ with covariance matrix $\mathbf{V} \in \mathbb{H}^{N_T}$, $\mathbf{V} \succeq \mathbf{0}$.

The signal received at active user $k \in \{1, \ldots, K\}$ and idle user $m \in \{1, \ldots, M\}$ are given by,

$$y_k = \mathbf{h}_k^H \mathbf{x}_k + \underbrace{\sum_{i \neq k}^{K} \mathbf{h}_k^H \mathbf{x}_i}_{\text{multiuser interference}} + \underbrace{\mathbf{h}_k^H \mathbf{v}}_{\text{artificial noise}} + n_k, \tag{2.3}$$

$$y_m^E = \sum_{k=1}^{K} \mathbf{l}_m^H \mathbf{x}_k + \underbrace{\mathbf{l}_m^H \mathbf{v}}_{\text{artificial noise}} + n_m, \tag{2.4}$$

respectively, where $\mathbf{h}_k \in \mathbb{C}^{N_T \times 1}$ and $\mathbf{l}_m \in \mathbb{C}^{N_T \times 1}$ are the channel vector between the BS and active user $k \in \{1, \ldots, K\}$ and the channel vector between the BS and the idle user $m \in \{1, \ldots, M\}$, respectively. We assume that all the channels are slowly time-varying

frequency flat fading channel. Variables $\mathbf{h}_k$ and $\mathbf{l}_m$ collect the effects of the path loss, shadowing and multipath fading of the corresponding frequency flat fading channels. Besides, $n_k$ and $n_m$ include background noises and thermal noises resulting from the receive antennas at active user $k$ and idle user $m$, respectively. Both of the two noises are modeled as Additive White Gaussian Noise (AWGN) which follow the same complex normal distribution with zero mean and variance $\sigma_n^2$.

## 2.2 Performance Metrics

The received Signal-to-Interference-plus-Noise Ratio (SINR) at active user $k$ is given by

$$\Gamma_k = \frac{\left|\mathbf{h}_k^H \mathbf{w}_k\right|^2}{\sum_{r\neq k}^{K} \left|\mathbf{h}_k^H \mathbf{w}_r\right|^2 + \left|\mathbf{h}_k^H \mathbf{v}\right|^2 + \sigma_n^2}. \tag{2.5}$$

In this thesis, Quality-of-Service (QoS) is taken into account for all active users which requires the SINR of all active users should satisfies a threshold $\Gamma_{\text{req}}$ to get a good link quality. At the same time, as mentioned earlier, all potential users are likely to become eavesdroppers and have the ability to decode signals, so in order to meet the security of the system, we need to assume a worst-case scenario and give a restriction. Suppose that the potential eavesdropper $m$ can cancel all Multiuser Interference (MUI) from all users except for active user $k$ and decode desired information to active user $k$. The received SINR at idle user $m$ for eavesdropping active user $k$ and is given by

$$\Gamma_{m,k}^E = \frac{\left|\mathbf{l}_m^H \mathbf{w}_k\right|^2}{\left|\mathbf{l}_m^H \mathbf{v}\right|^2 + \sigma_n^2}. \tag{2.6}$$

## 2.3 Resource Allocation Problem Formulation

In this section, we design an optimial resource allocation algorithm for minimizing the total transmit power of a multiuser system. Until now, we have both the SINR at active users $k \in \{1, \cdots, K\}$ and idle users $m \in \{1, \cdots, M\}$. To simplify the description, we define the following variable $\mathbf{W}_k = \mathbf{w}_k \mathbf{w}_k^H$, $\mathbf{H}_k = \mathbf{h}_k \mathbf{h}_k^H$ and $\mathbf{L}_k = \mathbf{l}_m \mathbf{l}_m^H$. Then the SINR at active user $k$ and SINR at idle user $m$ for eavesdropping user $k$ can be rewrite as follow, respectively

$$\Gamma_k = \frac{\text{Tr}(\mathbf{H}_k \mathbf{W}_k)}{\sum_{r\neq k}^{K} \text{Tr}(\mathbf{H}_k \mathbf{W}_r) + \text{Tr}(\mathbf{H}_k \mathbf{V}) + \sigma_n^2} \quad \text{and} \quad \Gamma_k^E = \frac{\text{Tr}(\mathbf{L}\mathbf{W}_k)}{\text{Tr}(\mathbf{L}\mathbf{V}) + \sigma_n^2}. \tag{2.7}$$

The optimal solution for minimization of the total transmit power of the considered system is obtained by solving the following optimization problem:

$$\underset{\mathbf{W}_k, \mathbf{V} \in \mathbb{H}^{N_T}}{\text{minimize}} \quad \sum_{k=1}^{K} \text{Tr}(\mathbf{W}_k) + \text{Tr}(\mathbf{V}) \tag{2.8}$$

$$\begin{aligned} \text{s.t.} \quad & \text{C1}: \sum_{k=1}^{K} \text{Tr}(\mathbf{W}_k) + \text{Tr}(\mathbf{V}) \leq P_{\max}, \quad \text{C2}: \frac{\text{Tr}(\mathbf{L}_m \mathbf{W}_k)}{\text{Tr}(\mathbf{L}_m \mathbf{V}) + \sigma^2_{\text{n}_m}} \leq \Gamma_{\text{tol}_m}, \forall m, \\ & \text{C3}: \frac{\text{Tr}(\mathbf{H}_k \mathbf{W}_k)}{\sum_{r \neq k}^{K} \text{Tr}(\mathbf{H}_k \mathbf{W}_r) + \text{Tr}(\mathbf{H}_k \mathbf{V}) + \sigma_n^2} \geq \Gamma_{\text{req}_k}, \ \forall k, \\ & \text{C4}: \mathbf{V} \succeq \mathbf{0}, \quad \text{C5}: \text{Rank}(\mathbf{W}_k) \leq 1, \ \forall k, \quad \text{C6}: \mathbf{W}_k \succeq \mathbf{0}, \ \forall k. \end{aligned}$$

In constraint C1, $P_{\max}$ represents the physical limitation of maximum transmit power at the BS. In constraint C2 and C3, we set the threshold $\Gamma_{\text{req}}$ and $\Gamma_{\text{tol}_\text{m}}$ which denote the minimum required SINR for active users and idle users to receive reliable information decoding, respectively. A high threshold $\Gamma_{\text{req}}$ leads to a better QoS, but also influences the feasibility of the considered optimization problem. With constraint C4 and $\mathbf{V} \in \mathbb{H}^{N_T}$, covariance matrix $\mathbf{V}$ should be a semi-definite Hermitian matrix. Constraint C5 and C6 are imposed to guarantee $\mathbf{W}_k = \mathbf{w}_k \mathbf{w}_k^H$ holds and covariance matrix $\mathbf{W}_k$ to be a positive semi-definite Hermitian matrix after optimization, respectively. The problem in 2.8 is a convex optimization problem and the optimal solution of 2.8 can be obtained via applying SDP.

Since the channel is varying over time, for a given $\Gamma_{\text{req}_k}$, we need to solve the optimization problem accordingly. With the increase of antennas and uses, we may be unable to return the optimal result via applying SDP within one time slot. How to quickly return the resulting resource allocation policy in a short time is a crucial issue. Because of this, many advanced algorithms cannot be implemented for practical application. In the next chapter, we will introduce a machine learning algorithm which is able to achieve the optimization results of the SDP algorithm while ensuring time efficiency.

# Chapter 3

# Resource Allocation Based on Machine Learning

In this chapter, we first introduce the theory and background in Section 3.1. In Section 3.2, we introduce the theoretical knowledge basis of machine learning and some applications on classification and regression issues. Under the condition of a large number of users, the SDP approach cannot yield the optimal solution in the effective time. To solve this problem, this chapter gives the design scheme of a dynamic network allocation architecture based on machine learning in Section 3.3. Finally, we consider some limitations of machine learning in Section 3.4..

## 3.1 Machine Learning Introduction

Since the earliest computers were invented, people have been working on ways to make computers smarter. Nowadays, artificial intelligence has become a hot topic with many practical applications such as autopilot [25], monitoring behavioral patterns of users [26] and natural language processing [27]. In the early days of artificial intelligence, the applications were often rule problems or complex mathematical formula problems, which are difficult for humans to solve but are very easy for computers. Computers can run complex mathematical calculations in a short time. However, the real challenges of artificial intelligence are those that are easy for humans to solve but difficult to describe, such as speech signal processing or image recognition. These problems can be easily solved by intuition or knowledge, but computers are not able to identify them in an efficiently manner. The main idea of machine learning is to provide a new solution to this type of problem. How to make a machine or computer program learn as what humans did is a challenge.

For decades, the process of machine learning can be summarized as, through the independent analysis of existing data sets, the internal laws are discovered and learned

and the learning rules are used to complete the accurate judgment and prediction of unknown new data. According to the characteristics of the model training samples, machine learning can be divided into supervised machine learning, unsupervised machine learning and semi-supervised machine learning.

1) Supervised machine learning

In the process of supervised machine learning, each data in the training set has a label or result. e.g. k-nearest neighbours classification [28], artificial neural networks and Random forest [29]. During the learning process, the model continuously adjusts the prediction results to the correct results and updates the model parameters until the required accuracy is achieved. When the model converges, we can classify the unlabeled data. The process shows in Figure 3.1.

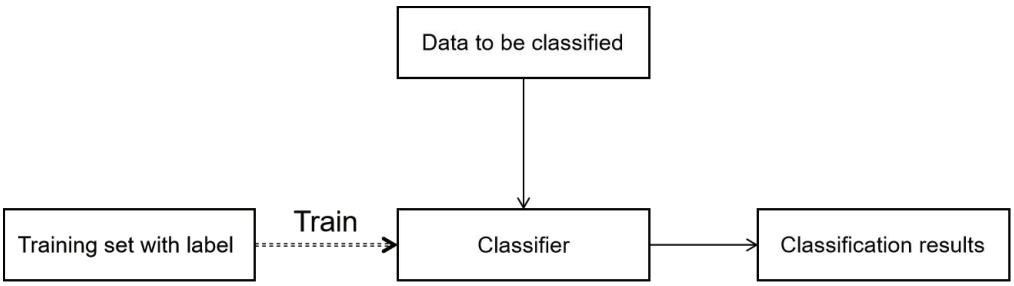


Figure 3.1: Supervised machine learning

2) Unsupervised machine learning

In the process of unsupervised machine learning, the data is not specifically identified, that is, there is no correct label or result. The learning model needs to discover some internal structure and classification rules of the data through its own mechanism and construct the corresponding classifier. e.g. clustering, AlexNet [30]. Figure 3.2 shows the learning process.


Figure 3.2: Unsupervised machine learning

3) Semi-supervised machine learning

The semi-supervised learning model is a combination of the above two models, that is, during the training process, only part of training samples contain labels.

The purpose of this practice is mainly to reduce the cost of marking samples. e.g. transductive support vector machine, co-training and label propagation [31]. The specific training process is shown in Figure 3.3 below,

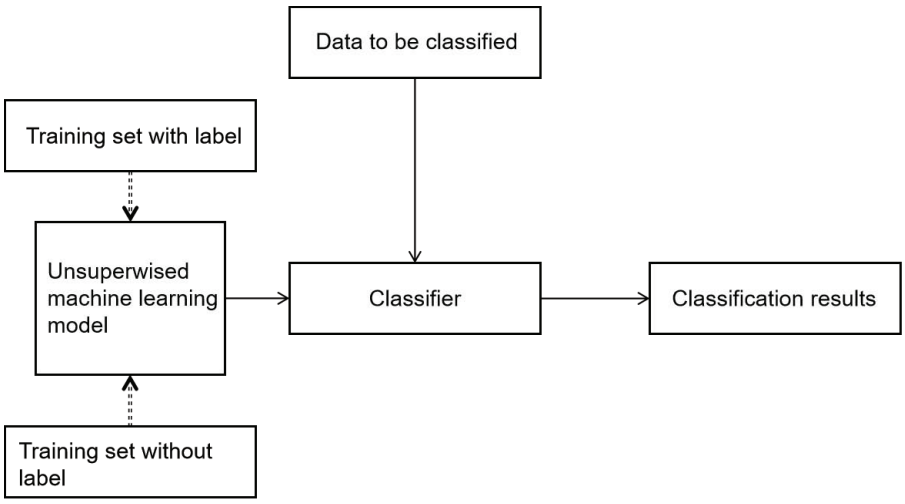


Figure 3.3: Semi-supervised machine learning

From the above introduction of the three machine learning models, it can be seen that the establishment of each model requires a complete training process. In general, the wider the coverage of training data, the more comprehensive the rules of model learning and the higher the accuracy obtained in prediction. So for a successful learning model, it must have enough training samples and a large enough span. However, it will slow down the training of the model correspondingly. This is a trade-off between accuracy and time, which constraints the development of machine learning at the beginning. But, Benefit from the advances in computing power, this is no longer an unsolvable problem. Once the training is finished and the parameters converge, the model enters the prediction stage. For new data features, classification and prediction only need simple numerical operations. This enables machine learning to solve more practical problems. The basic idea and starting point of this thesis is to use the probabilities of machine learning, solve the resource allocation optimization problem. In the model prediction stage, It can return results with a certain accuracy, thus can replace the SDP approach.

## 3.2 Design of Architecture Based on Machine Learning

The resource allocation decision structure proposed in this paper is illustrated in the following Figure 3.4. It contains a SDP approach layer and a machine learning-based decision layer. The architecture is deployed as a whole network controller, completing

network analysis and final allocation decisions. From the perspective of the process of processing the request, we can divide it into two phases.

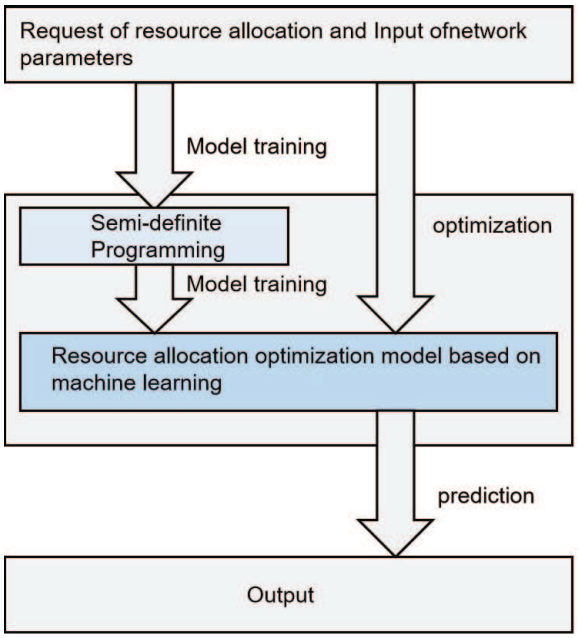


Figure 3.4: Resource allocation optimization model based on machine learning

1) In the first phase, under different network conditions, we use the SDP to calculate the optimal resource allocation results according to different parameter requirements and noise constraints. Then, The current network states input by the SDP approach are used as the features of training sample. the optimal result via applying SDP is used as a label for the training sample.

2) The second phase is resource allocation optimization. In this process, the SDP approach will no longer work. When the controller receives a new resource allocation request, the corresponding machine learning model will use the current network state and the constraints in the request as the input of the model. Since the network model has converged, the prediction will be directly returned as a result of an optimal allocation, the time required for this process will be greatly shortened compared to SDP. The results are very similar because the machine learning model already obtains the ability to optimally allocation like SDP approach selection during the training process.

In the first step of system construction, we will collect enough data and build a database of resource allocation schemes for the model. During the data collection process, we randomly sample $n$ groups different network state parameters according to the

real-time state of the network request and different target user SINR constraints while keeping the number of antenna and users unchanged. Then, the state information and the constant system parameters are used as the input of the SDP approach, the optimal allocation scheme for the resources under the current network state and constraints can be calculated. For example, if the controller receives a resource allocation request, SDP is applied to minimize the total power consumption and jointly find the BF distribution for each active user and artificial noise subject to the constraints defined in Chapter 2. When the calculation of all groups is finished, we store the input and output into a database correspondingly, taking 80% as the training set and the rest as the test set.

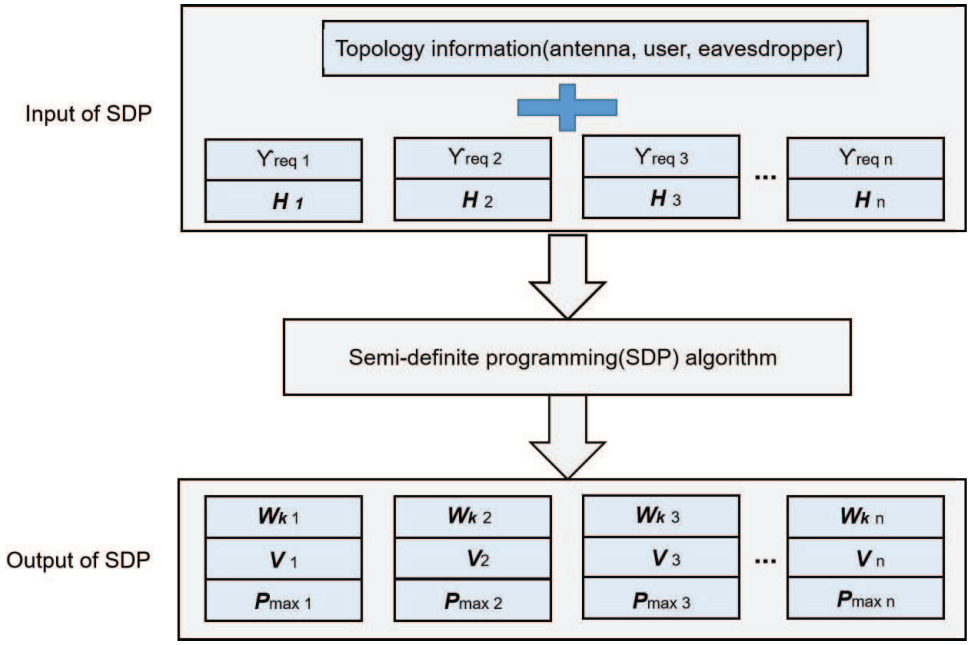


Figure 3.5: Training data collection

We adopt a supervised training model here. According to the previous introduction, the training sample needs to include two parts, sample features and sample labels. Based on the assumption of the reliability of the network environment, the structural information of the wireless communication network remains unchanged. So for SDP, the network parameter that truly affects the calculation result is the current channel matrix $\mathbf{H}$ which is related to the number of antennas, the total users of the network and the minimum required SINR for active users $\Gamma_{\text{req}}$ or the constraints set by the network administrator. Theoretically, if one of these two types of parameters changes, the results of SDP may result in a different solution. Figure 3.5 is a schematic diagram of the training process. We need to train all the nodes in the model until converge.

Therefore, to obtain all the useful parameters that represent the state of the network, the machine learning model takes all current network state parameters, as well as all

constraints, as features of its samples. As for the label of the sample, we can already get the optimal user's BF matrix **W** and AN covariance matrix **V** using SDP approach. At this point, the construction of the training sample has completed. The machine learning model can be properly trained given sufficient samples obtained by employing SDP approach, Figure 3.6 shows the model training process.

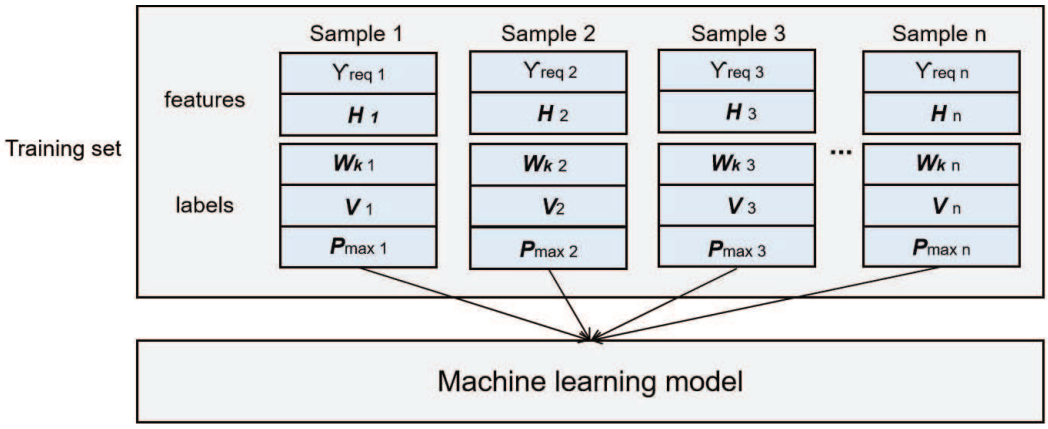


Figure 3.6: Training process

We use a 6-hidden layer FCN architecture to compute the active user BF distribution matrix **W** and artificial noise matrix **V**, we choose the mean absolute error as loss function $L$:

$$L = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x_i}| \tag{3.1}$$

where $x_i$ denotes the $i$-th prediction value of training sample and $\bar{x_i}$ denotes the $i$-th true label of training sample. By minimizing this function, we can update the variables of each layer until the model is converged. The specific training method will be described in the next chapter.

## 3.3 Resource Allocation Optimization

After the training process is finished, all the parameters of the machine learning-based model are converged. Then, we can utilize the obtained model to solve the considered problem. After receiving the relevant request, it can give a similar prediction to SDP based on the current network states and constraints independently without spending too much time.

The system collects the real-time network status information while receiving the connection request with the constraint. Since the supervised machine learning model has established a mapping between the input data and the label, in the prediction stage,

after new data are input, the model will quickly obtain the results and the controller will perform the BF allocation of the user and update artificial noise according to the result.

The prediction process obtains the solution of the considered problem in an efficient manner. If the accuracy of the machine learning algorithm tends 100%, then we can say machine learning-based model has the same ability to obtain the optimal solution of the resource allocation optimization problem as SDP. In terms of running time, SDP requires a huge calculation during the parameters improve, which means not enough time is available to return the optimal results. This is not acceptable under actual network operating conditions. On the contrary, a machine learning-based model only needs to complete the training process in advance. In the prediction phase, only simple numerical operations are needed. The required time is extremely short, which makes the practical application possible, this is the core problem that we are eager to solve.

## 3.4 Limitations of Machine Learning

In the machine model design process, we try to replace the function of the SDP, which means that we need to obtain the same input information as the SDP as the training set, so that it is possible to obtain similar predictions. In addition to these obvious network state parameters, the machine learning model also needs to have the ability to choose and extract parameters that may be truly utilized by the SDP approach, as useful training features. Furthermore, the data processing rules within the SDP approach are also important potential laws affecting the machine learning model prediction. So in summary, to finally achieve the goal of replacing the SDP approach, the following critical issues need to be considered.

First, in a FCN architecture, the neurons of next layer can be connected to all the neurons of the previous layer. A potential problem brought by this is the expansion of the total number of parameters. Suppose we have 1000 samples, each sample has 1000 features that need to be trained, then the hidden layer should have at least $1 \times 10^6$ nodes, only in this layer has $1 \times 10^{12}$ weights to be updated step by step. This not only increases the complexity of calculation but also leads to overfitting problems.

Second, for neural networks, since weight and bias matrices and back propagation algorithm are used to update parameters. The number of neurons of each layer is constant, which means the number of parameters can not change after the model training process. The shape of all samples for training and testing must be the same. It is a big constraint in the application. With the mathematical derivation from the previous section, we can notice that only one dimension data like $n \times 1$ can be used as input

data. If the data has more than one dimension, it must be reshaped first and then put in the model. This operation may change or destroy some important structure of input data such as a semi-definite matrix or the relationship between image pixels. And we need to pre-process the training data to ensure that the data format of each sample is uniform and uncorrelated. Otherwise, the shape of input and weights are not equal. However, in small wireless communication networks, the shape of channel allocation of signals will change with the number of antennas and users. This results in unequal numbers of features that can be extracted from each sample, which limits our use of neural networks. Filtering or changing the features may affect the training results and increase the calculation amount of pre-processing.

Third, the changes in network status will also affect the final system resource allocation. System parameters such as channel distribution and noise constraints should be trained together as features. For a certain channel environment, some parameters in the network may be more important than others, but if we use FCN model or other machine learning algorithms, we cannot judge the priority and importance of the features. However, if we manually specify the features of the input through previous knowledge and let the machine train. This will undoubtedly increase the complexity of manually processing data and we may ignore some potential important inner relationships. As the types and number of parameters of the network structure increase, if we have no methods to distinguish the important features, then all parameters should put into the model for training. As a result, the speed of training will slow down dramatically, the advantages of machine learning algorithms will vanish.

Finally, the SDP approach is based on the method of gradient calculation to obtain the optimal solution. This is a purely mathematical solution to the convex problem. For each input, we must calculate the result of SDP separately. However, in practical applications, we know that each input is not completely random and irrelevant. On the contrary, most of the input data follow the same data distribution. As a result, we can construct the corresponding data by exploring the intrinsic relationship of the data. Thus we can accelerate the learning process.

# Chapter 4

# Resource Allocation Based on Deep Learning

So far, this thesis has established a machine learning decision-making architecture for dealing with resource allocation problems. The design principles and processing flow have been explained. In Section 3.4, we have discussed the potential disadvantages of the machine learning model in solving such optimization problems without prior knowledge. Besides, some machine learning algorithms cannot fit in this architecture and achieve optimal or near-optimal results. Therefore, we give the advantages of deep learning algorithms applied to network resource allocation modeling. Then, in Section 4.1 we introduce some important concepts and classic models in deep learning and explain back propagation method in Section 4.2. In the actual application process, we need a powerful supervised deep learning model to solve the optimization classification problem with different constraints. Based on the original model, this chapter proposes a CNN model in Section 4.3. Through mathematical analysis, it is proved that the CNN model can independently consider the influence of constraints in the training and prediction process. Finally, the specific algorithm for training the model and the parameter update rules are given.

Based on the analysis of the above points, we especially need to pay attention to the feature selection of the data before processing. By screening out the features closely related to the model and eliminating other "noise", we can improve the training speed and accuracy. It is also possible to increase the interpretability robustness of the model. In fact, in machine learning research, a large part focuses on the data pre-processing process. How to select and process effective features determines whether the performance of a machine learning model can be improved. In the field of common speech signal recognition and image segmentation, we have a series of mature feature extraction methods, which facilitate us to quickly and standardize the extraction of high-quality data. However, much of these pre-processing methods are since people are already

familiar with the relevant fields, not by machine learning actively extracting their discoveries. For the considered resource allocation problem, how to get rid of the artificial constraints and find high-quality features to facilitate our subsequent training has become a key issue. And this is also the core idea of the recently proposed deep learning theory. Another issue to consider is the drawback of FCN architecture comparing with deep learning. Based on these considerations, we focus on deep learning models.

## 4.1 Introduction of Deep Learning

Neural network technology originated in the 1960s. Rosenblatt promoted the production of perceptron [32]. The model of the perceptron consists of an input layer, an output layer and a hidden layer. The input feature vector reaches the output layer through the hidden layer transformation, the classification result is obtained at the output layer. Figure 4.1 shows the structure of one perceptron.

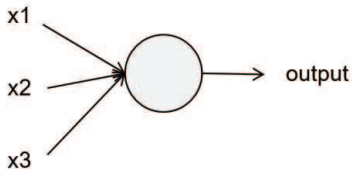


Figure 4.1: Perceptron

Suppose we have $m$ inputs, the inputs and output have a linear relationship,

$$z = \sum_{i=1}^{m} u_i x_i + b \tag{4.1}$$

then follows the activation function.

$$sign(x) = \begin{cases} -1 & z < 0 \\ 1 & z > 0 \end{cases} \tag{4.2}$$

However, perceptron cannot solve complex functional problems, such as XOR problems, which is a big constraint in practical. With the development of mathematics, this disadvantage is known to be overcome by the multi-layer perceptron in the 1980s [33].

The multi-layer perceptron has three changes based the perceptron:

1) The model has multiple hidden layers.

   By using multiple layers, it greatly improved the generalization ability of the model, enabling the model to describe more complex real-world function prob-

lems. Specifically, when there are more hidden layers, we can have more activation functions. At the same time, the ability to extract different features will be better. These explain why we add layers instead of neurons.

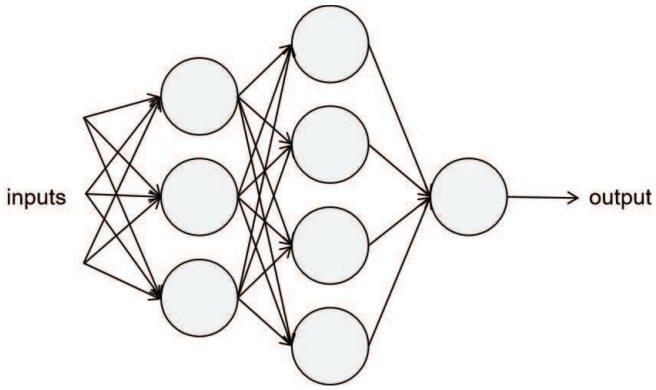


Figure 4.2: Perception with hidden layer

2) The model can have multiple outputs.

This makes the model more widely applicable, rather than limited to classification problems. This model can be flexibly applied to classification and regression, as well as other machine learning fields. Such as dimension reduction and clustering.

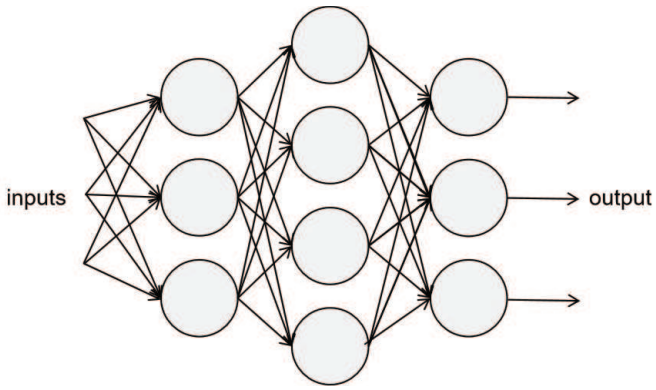


Figure 4.3: Perception with multiple outputs

3) Changes of the activation function

The third is to expand the selection range of the activation function. The activation function of the perceptron is $sign(x)$. Although it is simple to use, the application range is limited. Because this activation function is discrete and can only have two outputs. Therefore, the sigmoid, softmax and ReLU function are popularly used.

We can apply multi-layer perception for different applications by using different activation functions, the expressive power of the multi-layer perceptron has been further enhanced.

Through these changes, a multi-layer perceptron can get rid of the constraints of the early discrete transfer function. By using the back propagation algorithm during training, multi-layer perception became the current neural network. The neural network can solve the disadvantage of not being able to simulate XOR problems. Besides, multiple hidden layers can describe more complicated real-world problems.

In general, the number of layers of a neural network determines its ability to characterize complex functions, so the key point is to add more layers. However, this may cause some problems. On the one hand, as the number of layers increases, the optimization function becomes more and more easily trapped in the local optimal solution and deviates far from the global optimal solution. On the other hand, the problem of vanishing gradients also plagues us. The gradients tend to 0 when layers go deeper, parameters of the previous layers will have no changes or small changes in the training process. In 2006, Hinton used a pre-training method to alleviate the local optimal solution problem [34]. To overcome the vanishing of gradients, activation functions such as ReLU have replaced sigmoid, formed the basic structure of Deep Neural Network (DNN) today. The number of hidden layers can now reach 100. DNN can be understood as a neural network with multiple hidden layers.

Since the model consists of multiple layers, we also have many linear relationship weight $u$ and bias variables $b$. Take a three-layer neural network as an example, the specific definitions are as follows:

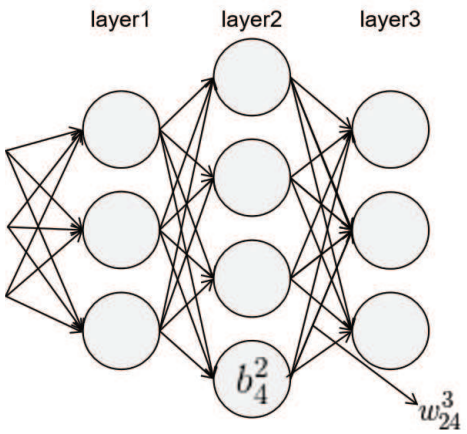


Figure 4.4: A three-layer neural network

$u_{24}^{3}$ denotes the linear relationship weight $u$ between the $4^{th}$ neuron in the second layer to the $2^{nd}$ neuron in the $3^{rd}$ layer, the superscript 3 denotes the number of layer, the subscript denotes the output index 2 of layer 3 and input index 4 of layer 2. $u_{jk}^{l}$ is the weight from the $k^{th}$ neuron in the $(l-1)^{th}$ layer to the $j^{th}$ neuron in the $l^{th}$ layer. There is no weight in the input layer.

$b_{4}^{2}$ denotes the bias of the $4^{th}$ neuron in the $2^{nd}$ layer, the superscript 2 denotes the number of layers, the subscript 4 denotes the index of the neuron. There is no bias in the output layer.

The development of machine learning can be summarized in Figure 4.5. From the original manually defining features method to model learning, to let the machine learn by itself and now to deep learning that reduces human intervention. We can find that the automatic extraction of features becomes more and more important.

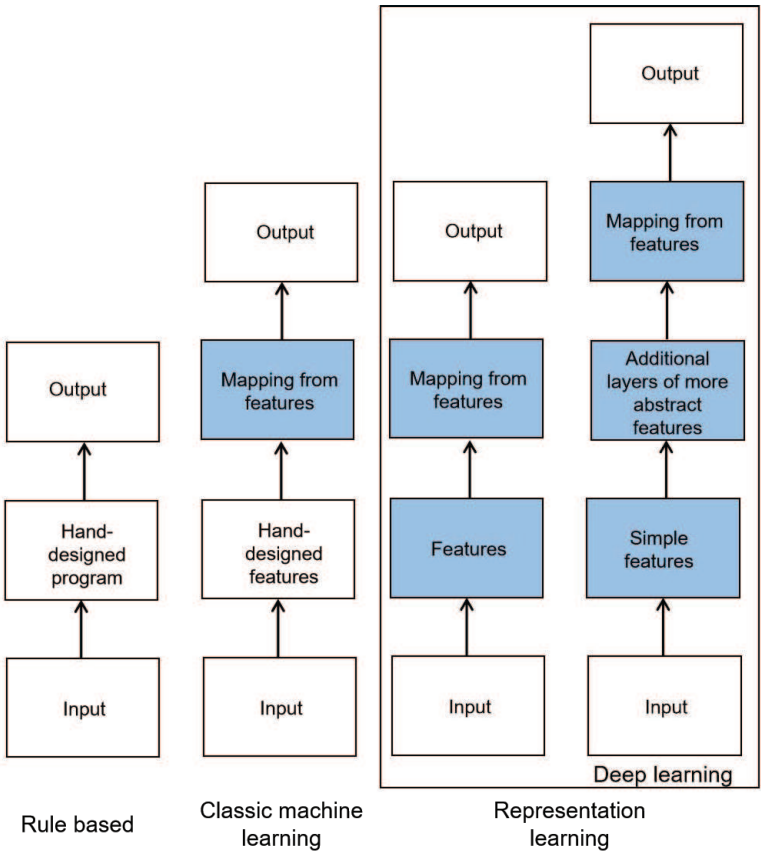


Figure 4.5: Development of machine learning

**Algorithm 1** Forward Propagation Algorithm

1: Initialize $\mathbf{a}^1 = \mathbf{x}$
2: **repeat**
3: Set $l = l + 1$
4: Calculate $\mathbf{a}^l = \sigma\left(\mathbf{z}^L\right) = \sigma\left(\mathbf{U}^l\mathbf{a}^{l-1} + \mathbf{b}^l\right)$
5: **until** $l = L$
6: Output $\mathbf{a}^L$

## 4.2 Data Propagation Algorithm

### 4.2.1 Forward propagation

If the activation function is $\sigma(z)$, we can calculate the input of next layer by the output from the previous layer like multi-layer perceptions. Suppose there are m neurons in the $(l-1)$ layer, the output of the $j^{th}$ neuron in the $l^{th}$ layer can be represented by,

$$a_j^l = \sigma\left(\sum_{k=1}^{m} u_{j,k}^l a_k^{l-1} + b_j^l\right) \tag{4.3}$$

This is the forward propagation algorithm of deep learning. For convenience, we can use matrix to represent, if there are $m$ neurons in the $l-1$ layer and n neurons in the $l-1$ layer, the linear weights in the $l$ layer form a $n \times m$ matrix $\mathbf{U}^l$, the bias of the $l$ layer form an $n \times 1$ matrix $\mathbf{b}^l$, the output of the $l-1$ layer form an $m \times 1$ matrix $\mathbf{a}^{l-1}$, the input of the $l$ layer form an $n \times 1$ matrix $\mathbf{a}^l$:

$$\mathbf{a}^l = \sigma\left(\mathbf{z}^L\right) = \sigma\left(\mathbf{U}^l\mathbf{a}^{l-1} + \mathbf{b}^l\right) \tag{4.4}$$

The forward propagation algorithm uses several weight matrices $\mathbf{U}$ and a bias vector $\mathbf{b}$ to perform a series of linear operations and activation operations with the input value vector $\mathbf{x}$. Starting from the input layer, the out of each layer is the input of the next layer. The forward calculation of the layer is performed until the operation reaches the output layer, then the output prediction is obtained.

### 4.2.2 Backward propagation

Suppose we have $m$ training samples $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ...., (\mathbf{x}_m, \mathbf{y}_m)\}$, where $\mathbf{x}$ is the input vector, the feature dimension is $n_{in}$, $\mathbf{y}$ is the output vector and the feature dimension is $n_{out}$. We need to use these $m$ samples to train a model. When there is a new test sample $(\mathbf{x}_{\text{test}}, ?)$, we can predict the output $\mathbf{y}_{\text{test}}$. If we use the DNN model, we have $n_{in}$ neurons in the input layer and $n_{out}$ neurons in the output layer, including several

neurons of hidden layers. At this time, it is necessary to find suitable linear weight matrices **U** and bias vectors **b** corresponding to all hidden layers and output layers, so that the calculated outputs of all training sample inputs is as close to the sample labels as possible.

We can use a suitable loss function to measure the results of the training samples. By optimizing this loss function to minimize the extreme value, the corresponding series of linear weight matrices **U** and bias vectors **b** can be updated. In DNN, the most common process of solving the extreme value of the loss function is generally done step by step through the gradient descent method.

Here we choose the mean absolute error function (MAE) as the loss function. For each sample, we want to minimize the following:

$$J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y}) = \left\|\mathbf{a}^L - \mathbf{y}\right\| \tag{4.5}$$

Then, we apply gradient descent and back propagation algorithm to find the suitable parameters **U** and **b** of each layer.

In the output layer, the loss function of the $L^{th}$ layer is:

$$J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y}) = \left\|\sigma\left(\mathbf{U}^L\mathbf{a}^{L-1} + \mathbf{b}^L\right) - \mathbf{y}\right\| \tag{4.6}$$

The gradient of **U** and **b**:

$$\frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{U}^L} = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^L}\frac{\partial \mathbf{z}^L}{\mathbf{U}^L} = \left(\mathbf{a}^L - \mathbf{y}\right)\left(\mathbf{a}^{L-1}\right)^T \odot \sigma'(\mathbf{z}) \tag{4.7}$$

$$\frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{b}^L} = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^L}\frac{\partial \mathbf{z}^L}{\mathbf{b}^L} = \left(\mathbf{a}^L - \mathbf{y}\right) \odot \sigma'\left(\mathbf{z}^L\right) \tag{4.8}$$

We can see that there is a same part $\frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^L}$, which we can calculate first:

$$\delta^L = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^L} = \left(\mathbf{a}^L - \mathbf{y}\right) \odot \sigma'\left(\mathbf{z}^L\right) \tag{4.9}$$

Then we calculate the gradient of each layer step by step, for the $l^{th}$ layer, gradient $\delta^l$ can be represented by:

$$\delta^L = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^L} = \left(\mathbf{a}^L - \mathbf{y}\right) \odot \sigma'\left(\mathbf{z}^L\right)\frac{\partial \mathbf{z}^L}{\mathbf{z}^{L-1}}\frac{\partial \mathbf{z}^{L-1}}{\mathbf{z}^{l+1}}\ldots\frac{\partial \mathbf{z}^l}{\mathbf{z}^{L-1}} \tag{4.10}$$

Based on the forward propagation algorithm,

$$\mathbf{z}^l = \mathbf{U}^l\mathbf{a}^{l-1} + \mathbf{b}^l \tag{4.11}$$

**Algorithm 2** Back Propagation Algorithm

1: Set the total number of layers $L$, the number of neurons of the hidden layer and output layer, activation function, loss function, iteration times max, threshold $\epsilon$, $m$ training samples, linear weight matrices $\mathbf{U}^1$, and bias vectors $\mathbf{b}^1$ of each hidden layer and output layer.
2: **repeat**
3: $\mathbf{a}^l = \mathbf{x}$
4: Apply forward propagation algorithm $\mathbf{a}^l = \sigma\left(\mathbf{z}^L\right) = \sigma\left(\mathbf{U}^l\mathbf{a}^{l-1} + \mathbf{b}^l\right)$
5: Calculate $\delta^{i,L}$ by loss function
6: Apply back propagation algorithm $\delta^{i,L} = \delta^l = \delta^{l+1}\frac{\partial \mathbf{z}^{l+1}}{\partial \mathbf{z}^l} = \left(\mathbf{U}^{l+1}\right)^T \delta^{l+1} \odot \sigma'\left(\mathbf{z}^l\right)$
7: Update $\mathbf{U}^l$, $\mathbf{b}^l$ based on $\mathbf{U}^l = \mathbf{U}^l - \mathbf{a}\sum_{i=1}^m \delta^{i,l}\left(\mathbf{a}^{\mathbf{i,l-1}}\right)^T$ and $\mathbf{b}^l = \mathbf{b}^l - \mathbf{a}\sum_{i=1}^m \delta^{i,l}$
8: **until** Changes of $\mathbf{U}$ and $\mathbf{b}$ smaller than $\epsilon$
9: Output linear weight matrices $\mathbf{U}$ and bias vectors $\mathbf{b}$ of each hidden layer and output layer

The gradient of weight and bias of the $l^{th}$ layer $\mathbf{U}^l$, $\mathbf{b}^l$ follows:

$$\frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{U}^l} = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^l}\frac{\partial \mathbf{z}^l}{\partial \mathbf{U}^l} = \delta^l\left(\mathbf{a}^{l-1}\right)^T \tag{4.12}$$

$$\frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{b}^l} = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^l}\frac{\partial \mathbf{z}^l}{\partial \mathbf{b}^l} = \delta^l \tag{4.13}$$

Until now, the key process left is to calculate $\delta^l$, we notice:

$$\delta^l = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^l} = \frac{\partial J(\mathbf{U},\mathbf{b},\mathbf{x},\mathbf{y})}{\partial \mathbf{z}^{l+1}}\frac{\partial \mathbf{z}^{l+1}}{\partial \mathbf{z}^l} = \delta^{l+1}\frac{\partial \mathbf{z}^{l+1}}{\partial \mathbf{z}^l} \tag{4.14}$$

The connection between $\mathbf{z}^{l+1}$ and $\mathbf{z}^l$:

$$\mathbf{z}^{l+1} = \mathbf{U}^{l+1}\mathbf{a}^{\mathbf{l}} + \mathbf{b}^{l+1} = \mathbf{U}^{l+1}\sigma\left(\mathbf{z}^l\right) + \mathbf{b}^{l+1} \tag{4.15}$$

$$\frac{\partial \mathbf{z}^{l+1}}{\partial \mathbf{z}^l} = \left(\mathbf{U}^{l+1}\right)^T \odot \sigma'\left(\mathbf{z}^l\right) \tag{4.16}$$

Put 4.16 in 4.14, we can get:

$$\delta^l = \delta^{l+1}\frac{\partial \mathbf{z}^{l+1}}{\partial \mathbf{z}^l} = \left(\mathbf{U}^{l+1}\right)^T \delta^{l+1} \odot \sigma'\left(\mathbf{z}^l\right) \tag{4.17}$$

By employing forward propagation algorithm and back propagation algorithm, we can calculate the gradient of each layer and update the linear weight matrices $\mathbf{U}$ and bias vectors $\mathbf{b}$ of each hidden layer and output layer during the training process. After the model converges, we can use the model to predict new samples.

## 4.3 Convolution Neural Network

CNN is an efficient identification method that has been developed in recent years and has attracted widespread attention. In the 1960s, when Hubel and Wiesel [35] studied the neurons in the cat's cerebral cortex for local sensitivity and direction selection, they found that their unique network structure can effectively reduce the complexity of the feedback neural network, then they proposed a CNN model. Now, CNN has become one of the research hotspots in many scientific fields, especially in the field of pattern recognition. Since the network avoids complex pre-processing of the data and can directly input the original samples, it has been more widely used. The use of CNN can overcome the mentioned limitations in Section 3.4, so we make further improvements with CNN model.

1) For CNN, the architecture shows in Figure 4.6, because it uses convolution layer and pooling layer, not all neurons in the upper and lower layers are connected, but operate through a convolution kernel instead. Also, neurons in the same layer can share the convolution kernel, which obviously reduces the number of weights we need to update in the training process. Weight sharing reduces the complexity of the network, and this makes the processing of high-dimensional data possible.

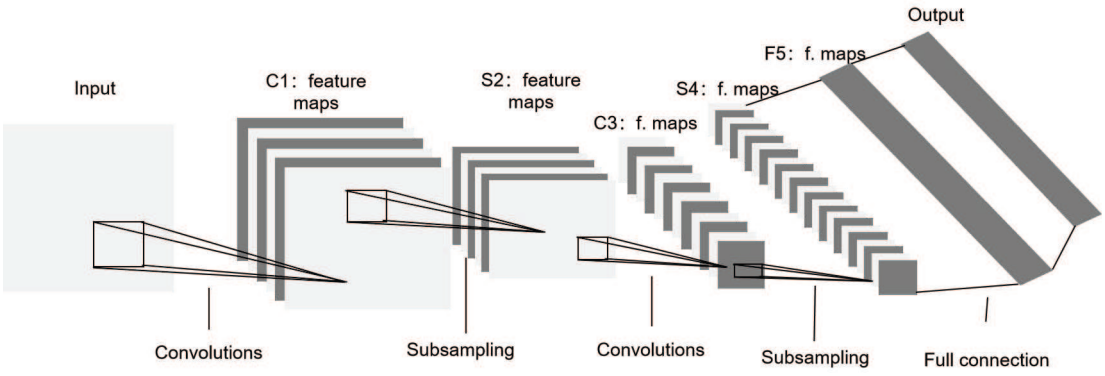


Figure 4.6: A Convolution neural network

2) The convolution operation can preserve the matrix of the input data because it has no constraint of dimension and shape, thereby we can discover potential learning rules and have no limitations about the shape. Especially the advantage that multi-dimensional input vectors can be directly inputted to the network avoids the complexity of data reconstruction during feature extraction and classification. Besides, using the pooling layer for sampling can accommodate a larger range of

feature values instead of manual screening and reduce the total number of parameters at the same time. All these properties ensure during CNN processing complicated data, there is no significant increase in time cost. Our input data and output data are all in the form of the matrix, which can be treated as different sizes to solve the parametric problem with multiple dimensions, such as the number of antennas and total users. This makes CNN model more extensive and practical.

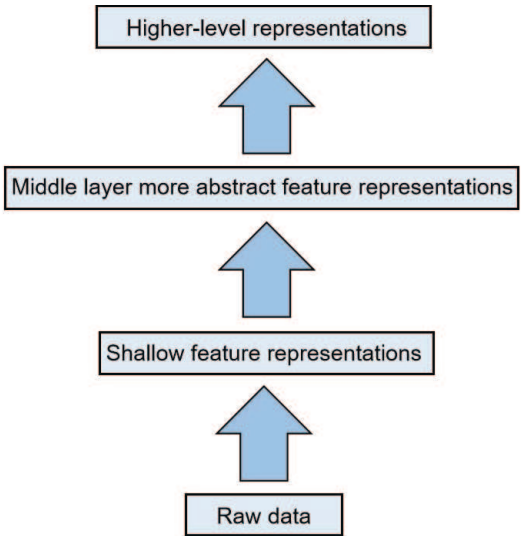


Figure 4.7: Model extraction feature process

3) A deep learning model consists of multiple non-linear transformation modules. We can extract and generate more abstract and useful features layer by layer. Specifically, deep learning can automatically learn hierarchically structured data and can simultaneously use both shallow feature representations and higher-level abstract feature representations generated by non-linear transformations. In this way, the model can learn the complex mapping between input and output. The model extraction feature process is shown in Figure 4.7.

A typical CNN model consists of three parts:

1) Convolution layer – Feature extraction

   In this process, we use a convolution kernel to perform a convolution operation with the original input to obtain the features of some small regions. In practical applications, we can use convolution kernels of different sizes. Each convolution kernel represents a feature extraction method. According to the experience obtained from experiments, a small number of convolution kernels will be set in the convolution layer closer to the input layer, more complicated convolution kernels will be set in the backward convolution layer. The size of the input becomes

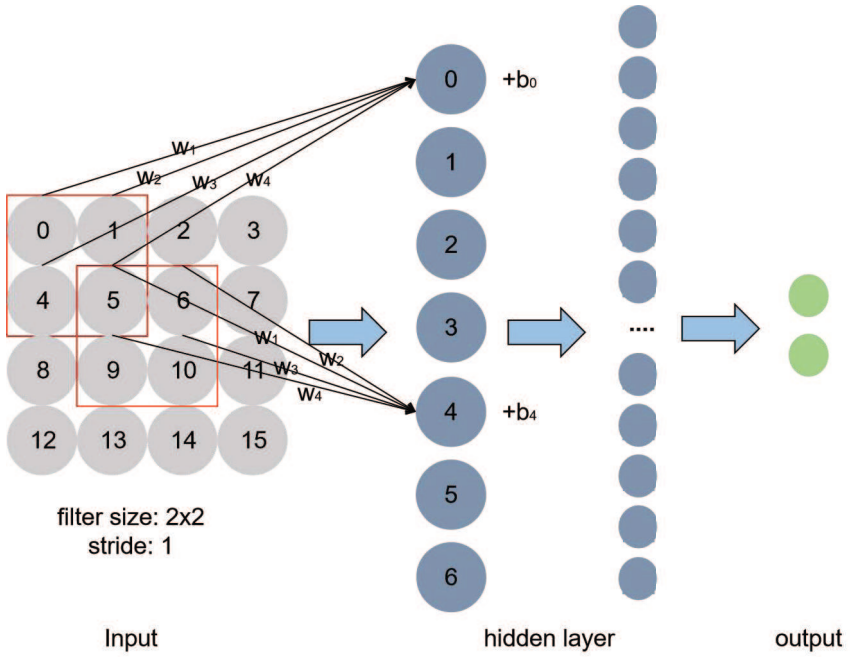


Figure 4.8: Operation of convolution layer

smaller after using the convolution kernel, which is convenient for subsequent calculations. We don't need to manually select the features. Instead, we only need to design the size, number and sliding step of the convolution kernel to let it train itself. The structure of the convolution layer shows in Figure 4.8. Suppose the size of the convolution kernel is $2 \times 2$ and stride is 1, the different point is that the neurons are partially connected, only data in the area of feature detector are connected. We only need to perceive the small parts, then combine the local information at a higher level to get the global information. The idea of partial connectivity is also inspired by the visual system structure in biology. Furthermore, the weights are shared for the whole features. This also means that the features we learn in this part can also be used in another part, so we can use the same learning features for all input data. By using the convolution layer, the total number of weights are dramatically decreased.

The output $y_0$ of convolution layer can be represented by,

$$y_0 = \left[m_1, m_2, m_3, m_4\right] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + b_0 \tag{4.18}$$

$m_i, i \in (0,..,1)$ denotes the weight of neuron and $x_i, i \in (0,..,1)$ denotes the features of input data.

2) Pooling layer – reduce parameters and avoid overfitting

The pooling layer is a sampling operation without affecting quality. Through the pooling layer, the data dimension can be greatly reduced, which not only reduces the amount of calculations, but also avoids overfitting problem. Common methods are average pooling and max pooling, cf. Figure 4.9 and 4.10.

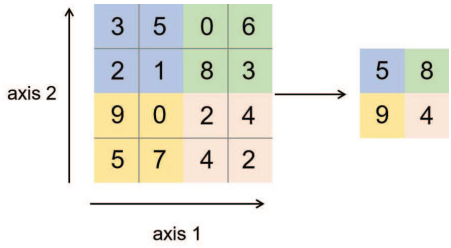


Figure 4.9: Max pooling

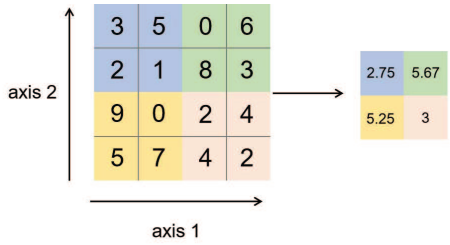


Figure 4.10: Average pooling

3) Fully connected layer – output

The data processed by the convolution layer and the pooling layer are input in the fully connected layer and then restored in the required output shape which is the same operation as in FCN.

## 4.4 Model Training

The training process and data extraction process for CNN are the same as in FCN. The difference is that we keep all input data and output data in a matrix form without doing reshaping. The specific model parameters and simulation results will be introduced in the next chapter.

# Chapter 5

# Simulation Results

This chapter conducts simulation experiments on two aforementioned resource allocation optimization models. We compare traditional SDP approach with machine learning models in terms of accuracy of results and algorithm run-time. Later, we compare the algorithm performance of the two models horizontally and analyze their advantages and disadvantages and application scenarios. In particular, we first, we apply the SDP approach using Matlab programming. Then we employ the Tensorflow module on Python to build a machine learning model. Tensorflow is an open-source software library that uses data flow graphs for numerical calculations. In Tensorflow, nodes represent mathematical operations in the graph and lines represent multidimensional data arrays or tensors, that are interconnected between the nodes. It supports parallel computation of multiple CPUs or GPUs. Tensorflow was originally developed by Google for its learning and research on deep neural networks. Figure 5.1 below shows the specific flow of the entire simulation experiment.

The simulation experiment is roughly divided into three stages. In the first stage, Matlab is used to implement the SDP approach. In particular, we set different parameters, the corresponding input and output are collected to form the training set and test set. The performance of the SDP approach such as running time is recorded at the same time. In the second stage, we separately train the FCN model and the CNN model based on the obtained data and pre-process the data according to the different requirements of the models. In the third stage, we compare the two models trained on the new test set in terms of prediction accuracy, performance and computing time, then compare with the SDP approach. Each phase is described in detail below.

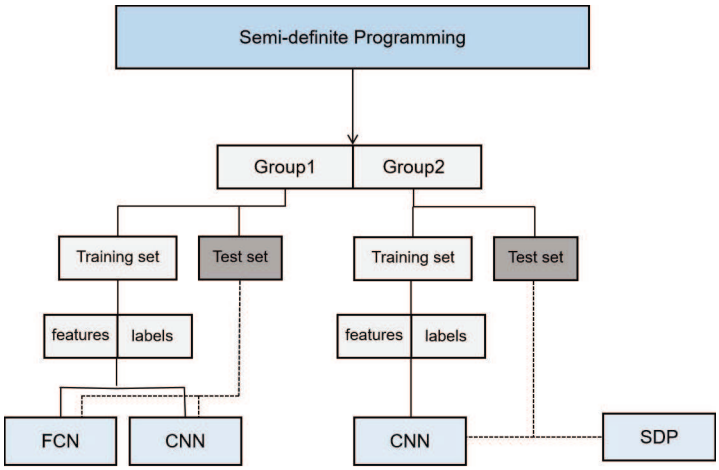


Figure 5.1: Simulation experiment flow

## 5.1 Data Preparation

### 5.1.1 Training data collection and database construction

The adopted simulation parameters are listed in Table 4.1. For the randomly generated channel matrix **H**, we apply SDP approach to obtain the optimization results, i.e. the BF matrix **W** and AN covariance matrix **V**. We study the impact of different parameters and set several groups.

Group 1, only one parameter: $\Gamma_{\text{req}}$. We keep the number of antennas equal to 8 and active users equal to 5, the eavesdroppers are 3. Then we set the threshold $\Gamma_{\text{req}}$ in the range between 10 dB to 30 dB with a step size of 1 dB.Given each $\Gamma_{\text{req}}$, the optimization problem is solved 1000 times. The number of total samples in Group 1 is 20000.

Table 5.1: System Parameters in SDP Simulations

| | |
|---|---|
| Target SINR of eavesdropper | −5 dB |
| Rician factor | 6 dB |
| Frequency bandwidth | 2 MHz |
| Reference distance | 30 meters |
| Active user noise power | −110 dBm |
| Potential eavesdropper noise power | −110 dBm |
| Base station antenna gain | 10 dBi |
| Samples of Group 1 | 1000*20 |
| Samples of Group 2 | 1000*20*2 |

Group 2, we choose some combinations ($N$,$K$) of antennas and active users: 8 antennas and 5 active users; 16 antennas and 9 ative users. For each combination, we repeat the steps in Group 1. The number of total samples in Group 2 is 40000.

We randomly select 20% data pairs of Group 1 and Group 2 as the test set and the rest as the training set.

### 5.1.2 Model Training

In the simulation experiments, we adopt the FCN model and the CNN model as independent supervised learning models to train and learn the data of Group 1. Then we use the CNN model to train the data of Group 2. Then, we compared the performance and accuracy of the two models with the same test set.

For the FCN model, because the input data can only be one-dimensional, we need to combine and flatten the input data and constraints into a one-dimensional row vector. Then calculate the rows as the number of neurons in the first hidden layer of the FCN model. For the CNN model, because it can process multi-dimensional data, we can keep the dimension of the data unchanged. But the convolution operation cannot handle complex numbers, we need to extract the real and imaginary parts of the input data separately and compose a matrix correspondingly. Then expand the constraints in the third dimension of the matrix. Put these three dimensions matrix into the CNN model for training. Finally, after prediction, the input data will be extracted accordingly and recombined into complex forms.

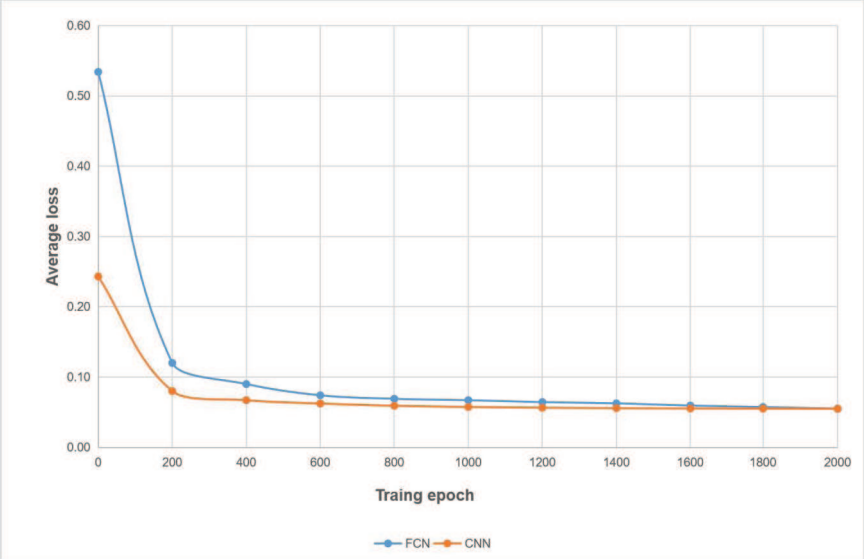


Figure 5.2: Training epoch vs. average loss of FCN and CNN

After the assignment of input data is completed, we set the hyperparameters of the two network models according to the number of training data and the number of features in the sample and conducted multiple comparative experiments on different combinations of hyperparameters. Specifically, the selection of batch size and training epoch depend on the downtrend of average loss. We choose the fast-falling parameters. In Figure 5.2, we study training epoch versus the average loss of the FCN model and the CNN model. It is expected that, as the training epoch increases, the average loss of both models are rapidly decline. This means both models converge quickly and the training process is accelerated.

## 5.2 Result of Prediction Using Group 1

In the prediction stage, the test set of Group 1 in Section 5.1.1 is utilized to verify the accuracy of the two models. We use MAE as loss function, and normalize the average loss. To better evaluate the prediction performance of the model, we calculate the average predictions of 100 samples. Because SDP approach yields the optimal solution, it is not listed here.

| Model | CNN | FCN | SDP |
|---|---|---|---|
| Nodes of first layer | 16 | 65 | \ |
| Number of hidden layers | 4 | 8 | \ |
| Learning rate | 0.0001, decay 0.95 | 0.001 | \ |
| Drop out | 0.8 | \ | \ |
| Nodes of output layer | 768 | 384 | \ |
| Training iteration | 20 | 20 | \ |
| Batch size | 64 | 64 | \ |
| Training epoch | 2000 | 2000 | \ |
| Accuracy | 90.82% | 88.97% | \ |
| Prediction time | 0.03 | 0.037 | 0.02 |

Figure 5.3: Result of prediction using Group 1

We compare the degree of loss between the predicted results and the true labels and calculate the accuracy. As can be seen from Figure 5.3, the accuracy of both models

is higher than 80%. This reveals that both models can get approximately accurate prediction results. The number of the neurons in the FCN model are much more than in the CNN model, this is decided by the architecture of models. We set the drop out rate equals 0.8 in the CNN model, which help us not only reduce probability of overfitting but also increase the robustness of the model.

In conclusion, when we solve simple problems, e.g. fewer variables and constraints, both the FCN model and the CNN model can get near-optimal predictions. The FCN model has simple architecture and less hyperparameters to choose from. Besides, compared with the FCN model, the CNN model has fewer model layers and shorter training time.

## 5.3 Parametric Prediction Based on CNN Using Group 2

It can be seen from the above comparison that, given a specific system setup, the prediction results of the FCN model are close to optimal, the run-time has also been reduced. But when we have different data from systems with different setups, the data shape are not the same such as data in Group 2, the FCN model can not handle such a parametric problem. Therefore, we can only evaluate the prediction results on the CNN model. We select Group 2 in Section 5.1.1 for training on the CNN model and use the test set to evaluate performance and accuracy.

| Model | CNN | SDP |
|---|---|---|
| Nodes of first layer | 16 | \ |
| Number of hidden layers | 4 | \ |
| Learning rate | 0.0001,decay0.95 | \ |
| Drop out | 0.8 | \ |
| Nodes of output layer | 1152 | \ |
| Training iteration | 20 | \ |
| Batch size | 64 | \ |
| Training epoch | 2000 | \ |
| Accuracy | 88.7% | \ |
| Prediction time(s) | 0.03 | 0.83 |

Figure 5.4: Result of prediction using Group 2

In Figure 5.4, even if the complexity of the input data increase, the prediction accuracy of the CNN model is still very high. If we collect enough training samples with different

parameters, the prediction accuracy will not be disturbed. However, for complex input data, more training samples means the neural network learning time will increase.

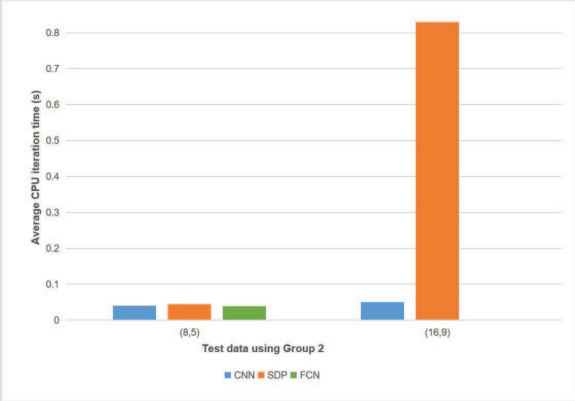


Figure 5.5: Input data vs. average CPU time per iteration for different resource allocation schemes

In Figure 5.5, we calculate the average CPU iteration time of 100 samples with two combinations of antennas and active users. The FCN model cannot solve parametric problem, so we only test it using 8 antennas and 5 active users. With the increase of antennas and users, the average iteration time of the SDP approach rises rapidly. Compared with the SDP approach, the CNN model has no obvious change on iteration time, which means the CNN model is able to tackle complicate parametric problems in a efficient manner.

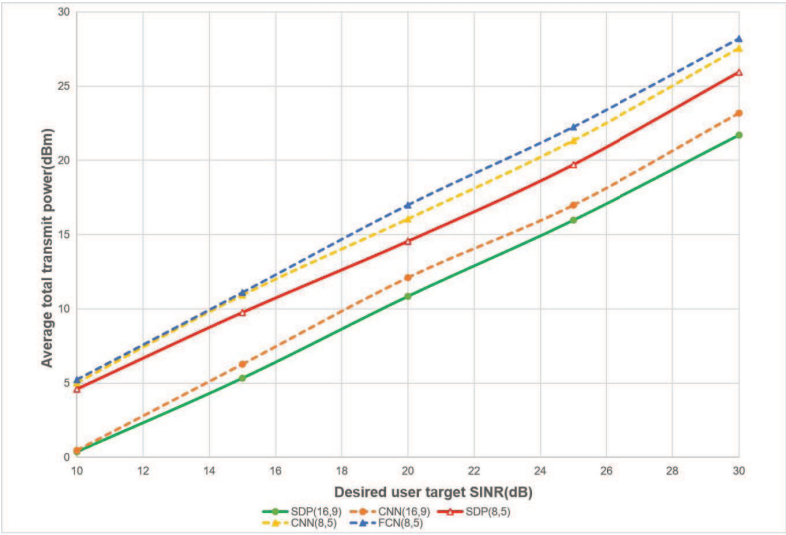


Figure 5.6: Desired user target SINR(dB) vs. average total transmit power(dBm) for different resource allocation schemes

In Figure 5.6, we separately use two combinations of antennas and active users: 8 antennas and 5 active users; 16 antennas and 9 active users as test data. We investigate the desired user target SINR(dB) of 100 samples versus the average total transmit power(dBm) for different resource allocation schemes. Before calculate the total transmit power. We removed all the predictions that did not meet the constraints of BF matrix **W** and AN matrix **V**. The prediction results of the CNN model and the FCN model have a similar trend to SDP while desired user target SINR changing. When desired user target SINR increases, the average total transmit power will improve. Meanwhile, with the increase of antennas, the average total transmit power decreases.

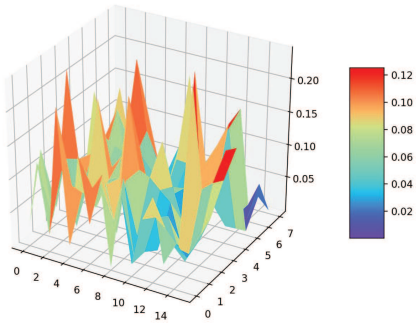

Figure 5.7: Average loss matrix of **W**

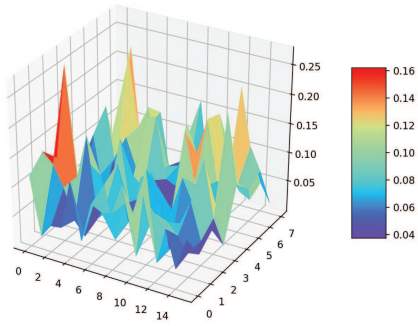

Figure 5.8: Average loss matrix of **V**

For the CNN model prediction of single sample, we calculate the average absolute loss of 100 samples. The losses of BF matrix **W** and AN matrix **V** are illustrated in Figure 5.7 and Figure 5.8, respectively. Because each matrix has different magnitude, we divide the loss matrix by the max value of **W** and **V**, respectively. Most values of the loss metrics are between 0 and 0.2. This shows that the CNN network has fully learned the mapping between different data.

# Chapter 6

# Conclusion

In this thesis, we investigated the resource allocation optimization problems for a multiuser wireless communication system. In particular, we proposes two different neural network structures based on machine learning theory – FCN and CNN. Then we introduce the realization process and the results of the two models. Finally, they were applied to the resource optimization problem and the advantages and disadvantages of the two models and SDP approach were compared horizontally. The proposed deep learning model can independently solve prediction problems with multiple constraints in a supervised learning model. After the model is trained, the prediction process becomes extremely simple and efficient which meets time requirements for practical problems. Compared with the widely used SDP algorithm, the proposed schemes have specific advantages in the optimization of network resource allocation, because it fully considers the service quality requirements of data and the resource optimization goals of network managers. On the other hand, the results given by our model are nearly the same as the optimization results of the SDP approach, but the calculation time of the model is much shorter than the SDP approach, so as to meet the necessary conditions for rapid decision-making during the actual network operation.

Via simulation, we show that FCN can already reach the accurate value of approximate SDP approach in an efficient manner. In the face of parametric and multiple constraints problems, CNN is much better than FCN in performance, mainly because CNN can process high dimension data and maintain the features of the input data structure. With multiple hidden layers, the model can mine more abstract features and improve the accuracy of the final results. Compared with FCN, the disadvantage of CNN is that many hyperparameters need to be initialized in advance. Finding suitable hyperparameters has a great impact on the prediction results, which requires us to take time to compare.

For future work, because the optimal solution cannot be obtained for non-convex problems, only approximate solutions exist, so this thesis does not cover it. However, the logic and basic algorithms of neural networks to solve such problems are the same

as the convex optimization problem. How to use a neural network for training while being able to obtain approximate solutions for a large number of non-convex problems is an important task in the future. Furthermore, this thesis mainly considers resource optimization in one single time slot. The comparison of several methods is also based on this premise. But resource optimization is always a dynamic problem. The input data is a function that changes over time, it is not completely random. Therefore, the performance of neural networks and traditional methods is also an issue that we need to consider under long-term conditions.

# Bibliography

[1] M. Khan and V. Niemi, "Aes and snow 3G are feasible choices for a 5G phone from energy perspective," in *Proc. Intern. Conf. 5G for Future Wireless Networks*. Springer, 2017, pp. 403–412.

[2] V. W. Wong, R. Schober, D. W. K. Ng, and L.-C. Wang, *Key Technologies for 5G Wireless Systems*. Cambridge University Press, 2017.

[3] J. Zhang, E. Björnson, M. Matthaiou, D. W. K. Ng, H. Yang, and D. J. Love, "Multiple antenna technologies for beyond 5G," *arXiv preprint arXiv:1910.00092*, 2019.

[4] D. W. K. Ng, E. S. Lo, and R. Schober, "Robust beamforming for secure communication in systems with wireless information and power transfer," *IEEE Trans. Wireless Commun.*, vol. 13, no. 8, pp. 4599–4615, Apr. 2014.

[5] D. Xu, Y. Sun, D. W. K. Ng, and R. Schober, "Robust resource allocation for UAV systems with UAV jittering and user location uncertainty," in *Proc. IEEE Global Commun. Conf. (GLOBECOM) Workshops, Abu Dhabi, United Arab Emirates*, Dec. 2018, pp. 1–6.

[6] D. W. K. Ng, M. Shaqfeh, R. Schober, and H. Alnuweiri, "Robust layered transmission in secure MISO multiuser unicast cognitive radio systems," *IEEE Trans. Veh. Tech.*, vol. 65, no. 10, pp. 8267–8282, Dec. 2015.

[7] J. E. Beasley, *Advances in linear and integer programming*. Oxford University Press, Inc., 1996.

[8] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The linear complementarity problem*. SIAM, 1992, vol. 60.

[9] S. Burer and R. D. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Mathematical Programming*, vol. 95, no. 2, pp. 329–357, 2003.

[10] E. B. Bajalinov, *Linear-fractional programming theory, methods, applications and software*. Springer Science & Business Media, 2013, vol. 84.

[11] X. Yu, D. Xu, and R. Schober, "Optimal beamforming for MISO communications via intelligent reflecting surfaces," *arXiv preprint arXiv:2001.11429*, 2020.

[12] Y. Sun, D. W. K. Ng, J. Zhu, and R. Schober, "Robust and secure resource allocation for full-duplex MISO multicarrier NOMA systems," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 4119–4137, 2018.

[13] D. Xu, Y. Sun, D. W. K. Ng, and R. Schober, "Multiuser MISO UAV communications in uncertain environments with no-fly zones: Robust trajectory and resource allocation design," *IEEE Trans. Commun.*, 2020.

[14] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and R. Schober, "Optimal 3D-trajectory design and resource allocation for solar-powered UAV communication systems," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4281–4298, 2019.

[15] D. Xu, X. Yu, Y. Sun, D. W. K. Ng, and R. Schober, "Resource allocation for secure IRS-assisted multiuser MISO systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM) Workshops*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[16] Y. Sun, D. W. K. Ng, D. Xu, L. Dai, and R. Schober, "Resource allocation for solar powered UAV communication systems," in *Proc. IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. Kalamata, Greece: IEEE, Jun. 2018, pp. 1–5.

[17] X. Yu, D. Xu, Y. Sun, D. W. K. Ng, and R. Schober, "Robust and secure wireless communications via intelligent reflecting surfaces," *submitted for publication, arXiv preprint arXiv:1912.01497*, 2019.

[18] X. Yu, J.-C. Shen, J. Zhang, and K. B. Letaief, "Alternating minimization algorithms for hybrid precoding in millimeter wave MIMO systems," *IEEE J. Sel. Areas Commun.*, vol. 10, no. 3, pp. 485–500, Feb. 2016.

[19] D. Xu, X. Yu, and R. Schober, "Resource allocation for intelligent reflecting surface-assisted cognitive radio networks," *arXiv preprint arXiv:2001.11729*, 2020.

[20] X. Yu, D. Xu, and R. Schober, "Enabling secure wireless communications via intelligent reflecting surfaces," in *Proc. IEEE Global Commun. Conf. (GLOBECOM), Waikoloa, HI, USA*, Dec. 2019, pp. 1–6.

[21] ——, "MISO wireless communication systems via intelligent reflecting surfaces," in *Proc. IEEE Int. Conf. Commun. China (ICCC)*, Changchun, China, May 2019, pp. 1–6.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[23] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[24] A. G. Ivakhnenko and V. G. Lapa, "Cybernetics and forecasting techniques," 1967.

[25] C. Insights, "Corporations working on autonomous vehicles," *Retrieved on March*, vol. 16, p. 2017, 33.

[26] D. Schutzer, "CTO corner: Artificial intelligence use in financial services-financial services roundtable," 2015.

[27] L. Braden-Harder, S. H. Corston, W. B. Dolan, and L. H. Vanderwende, "Apparatus and methods for an information retrieval system that employs natural language processing of search results to improve overall precision," Aug. 1999, uS Patent 5,933,822.

[28] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[29] T. K. Ho, "Random decision forests," in *Proc. of 3rd international conference on document analysis and recognition*, vol. 1, Montreal, Quebec, Canada, Aug. 1995, pp. 278–282.

[30] D. Gershgorn, "The inside story of how AI got good enough to dominate Silicon Valley," *Quartz. Retrieved*, vol. 5, 2018.

[31] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[32] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[35] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.