Technische Universität Berlin

Institut für Telekommunikationssysteme

Fachgebietes Netzwerk-Informationstheorie

Fakultät IV

Einsteinufer 25

10587 Berlin

http://www.netit.tu-berlin.de

Master Thesis

Deep learning for channel representation and user clustering in radio access networks

Yasser Saeid

Matriculation Number: 376700

28.12.2018

Referent: Prof. Dr.-Ing. Slawomir Stanczak

Chair of Network Information Theory

Co-Referent: Prof. Dr. Setareh Maghsudi

Institute of Telecommunication Systems

Supervisors: Dr.-Ing. Zoran Utkovski,

Fraunhofer Heinrich Hertz Institute (HHI)

Acknowledgment

I would like to express my grateful thanks to my first reviewer, Prof. Dr.-Ing. Slawomir Stanczak, for his guidance and support during my master studies.

Furthermore, I would like to thank my second reviewer, Prof. Dr. Setareh Maghsudi, for her time and effort dedicated to review and discuss my work progress.

I owe deepest gratitude to my direct supervisor, Dr.-Ing Zoran Utkovski. He offered me the adequate time, concern, and advice to help me grow in the research field. This work would not have been done without his continues guidance and generous support.

I want to dedicate this thesis to my loving mother, father and my sisters (Areeg and Afrah). Their words of encouragement pushed me further and helped me get through the hardest times.

I also dedicate this thesis to my lovely friends:Abd,Adem Ali Mhd, Ali Hadidi, Amer, Wissam, Sameh and Abdullah. I hope, it will motivate and encourage them to get through difficulties in pursuing their dreams.

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 28.12.2018

.....

Yasser Saeid

Abstract

In massive MIMO systems, base stations (BSs) may deploy a large number of antennas. When the system operates in a frequency-division-duplex (FDD) mode, the extensive number of antennas poses a significant challenge to the channel estimation in the downlink of the communication system. To deal with this issue, one option is to explore the possible underlying channel structure whereby the high-dimensional channel vector admits a low-dimensional (sparse) representation in some basis or *dictionary*. Under this model, estimation of the downlink channel vectors translates to solving a sparse inverse linear problem. Hence, the compressed sensing framework can be leveraged to obtain accurate channel estimation with reduced training and feedback overhead, proportional to the sparsity level of the channel.

Motivated by the work in [1], in this thesis, we adopt a dictionary learning based channel model (DLCM) according to which a *learned, overcomplete dictionary* is used to represent the channel vectors. The motivation behind DLCM is that the dictionary, due to the learning process, is able to adapt to the cell characteristics as well as insure a sparse representation of the channel. Additionally, since no structural constraints are placed on the dictionary, the approach is applicable to an arbitrary array geometry and also does not need accurate array calibration.

To perform channel estimation in the DLCM model, it is necessary to extend the CS-based framework to be able to handle signals that have sparse representations in a learned redundant dictionary, rather than a predefined orthogonal basis. Focusing on this problem, we investigate the performance of two classes of algorithms. The first class comprises of adaptations of "classical" CS reconstruction algorithms (LASSO and Signal-Space COSSAMP), to account specifically for the learned overcomplete dictionary in the channel estimation step. The other class of algorithms leverages the *deep learning* (DL) framework which has been recently successfully adopted in many applications such as speech recognition and image classification. In the context of high-dimensional channel estimation, we use the deep learning approach to mimic, using a neural network, iterative algorithms such as approximate message passing

(AMP) and iterative shrinkage-thresholding algorithms (ISTA).

Besides the use of the CS and DL frameworks to perform channel estimation, we investigate the possibility to provide compressed channel representation by performing *unsupervised clustering*. For this purpose we evaluate the performance of two algorithms. The first one is an adaptation of the classical K-means algorithm. The second one is a non-parametric method based on *deep embedded clustering* in the spirit of [2]. The compressed channel representation may be seen as an alternative to methods based on compressed channel feedback, with the difference that the channel compression takes place directly in the domain of the low-dimensional channel representation.

Zusammenfassung

In Massive-MIMO-Systemen, können Basisstationen (BSs) eine große Anzahl an Antennen besitzen. Wenn das System in einem Frequenz-Division-Duplex (FDD) Modus arbeitet, führt die große Antennenanzahl zu einer signifikanten Herausforderung bei der Kanalschätzung im Downlink des Kommunikationssystems. Eine Option um dieses Problem zu lösen ist, die zugrunde liegende Kanalstruktur zu erforschen, wobei der hochdimensionale Kanalvektor eine niedrigdimensionale (sparse/ spärliche) Darstellung in einer Basis oder einer Dictionary zulässt. Unter diesem Modell führt die Schätzung der Downlink-Kanalvektoren zur Lösung eines sparse inverse linear problem. Daher kann die komprimierte Erfassung genutzt werden um eine genaue Kanalschätzung, mit reduziertem Trainings- und Feedback-Overhead, proportional zum Sparsamkeitsgrad des Kanals, zu erhalten.

Motiviert durch die Arbeit in [1], wird in dieser Abschlussarbeit ein Dictionary Learning Based Channel Model (DLCM) verwendet, dass ein gelerntes und überkomplettes Dictionary besitzt, um die Kanalvektoren darzustellen. Die Motivation hinter DLCM ist, dass das Dictionary durch den Lernprozess in der Lage ist, sich an die Zellmerkmale anzupassen und eine spärliche Darstellung des Kanals zu gewährleisten. Da dem Dictionary keine strukturellen Einschränkungen auferlegt werden, ist der Ansatz auf eine beliebige Matrix-Geometrie anwendbar und erfordert auch keine genaue Array-Kalibrierung. Um die Kanalschätzung im DLCM-Modell durchzuführen ist es notwendig, das CS-basierte Framework zu erweitern, um die Signale mit spärlichen Darstellungen in einem gelernten redundanten Dictionary und nicht in einer vordefinierten orthogonalen Basis verarbeiten zu können. Mit Blick auf dieses Problem wird die Leistung von zwei Klassenalgorithmen untersucht. Die erste Klasse umfasst Anpassungen von "klassischen" CS-Rekonstruktionsalgorithmen (LASSO und Signal-Space COSAMP), um speziell das gelernte überkomplette Dictionary im Kanalschätzungsschritt zu berücksichtigen. Die andere Klasse nutzt das Deep Learning (DL)-Framework, das in der heutigen Zeit erfolgreich in vielen Anwendungen wie Spracherkennung und Bildklassifikation eingesetzt wird. Im Rahmen der hochdimensionalen Kanalschätzung wird der Deep-Learning-Ansatz zur Nachahmung von iterativen Algorithmen, wie Approximate Message Passing (AMP) und Iterative Shrinkage-Thresholding-Algorithmen

(ISTA) unter Benutzung eines neuronalen Netzwerks, verwendet.

Neben der Verwendung der CS- und DL-Frameworks zur Durchführung von Kanalschätzungen wird mithilfe von unsupervised Clustering Methoden die Möglichkeit untersucht, eine komprimierte Kanaldarstellung bereitzustellen. Aus diesem Grund werden die Leistungen von zwei Algorithmen bewertet. Die erste ist eine Anpassung des klassischen K-Mittel-Algorithmus. Die zweite ist eine nicht parametrische Methode, die auf tief eingebettete Clustering Methoden im Sinne von [2] basiert. Die komprimierte Kanaldarstellung kann als Alternative zu Verfahren, welche auf komprimierter Kanalrückführung basieren, gesehen werden. Mit dem Unterschied, dass die Kanalkompression direkt in der Domäne der niederdimensionalen Kanaldarstellung stattfindet.

Contents

List of Figures

List of Tables

1 Introduction

By deploying a large antenna array at the base station (BS), the base station is able to perform both receive and transmit beamforming, thus reducing multiuser interference and thereby increasing the cell throughput. For effective downlink beamforming, it is necessary to have accurate knowledge of the downlink channel state information at the transmitter (CSIT). In a time-division duplexing (TDD) system, downlink CSI can be obtained by exploiting uplink/downlink channel reciprocity. The common assumption in Massive MIMO is that each user terminal (UT) only has a small number of antennas and that the BS can use uplink channel information, obtained from the (relatively easy) uplink training, for downlink beamforming [15]. On the other hand, most cellular systems today employ FDD, as frequency-division duplexing (FDD) is generally considered to be more effective for systems with symmetric traffic and delay-sensitive applications [15] [16]. Channel reciprocity is no longer valid in FDD systems and in order to obtain CSIT, the BS has to perform downlink training. Subsequently, the user needs to estimate, quantize and feedback the channel state information.

To deal with the limited downlink channel training interval in a FDD Massive MIMO system, one option is to explore possible underlying channel structure whereby the high dimensional channel vector has a low dimensional representation [17] [18]. Motivated by the framework of *compressive sensing* (CS), if the desired signal (channel response) admits a sparse representation in some basis or *dictionary*, then the number of measurements (downlink training period) is proportional to the number of nonzero entries and the signal can be robustly recovered using sparse signal recovery algorithms [19] [20]. In this regard, there are two important questions: first, does the underlying physical propagation environment support a sparse channel representation? Second, can we find a good sparsifying dictionary $\mathbf{C}$ such that the mathematical model is valid?

The sparse channel representation is supported by the observations made in [1],

[21], [3], [22] for the Geometry-Based Stochastic Channel Model (GSCM) [3]. To summarize, the assumptions are that for a specific cell, the locations of the dominant scattering clusters are determined by cell specific attributes such as the buildings, terrain, etc. and are common to all users independent of user position. Given that the scattering clusters are far away from the BS, the subpaths associated with a specific scattering cluster will be concentrated in a small range around the line of sight (LOS) direction between the BS and the scattering cluster, i.e. having a small angular spread (AS). Further, by assuming that the user terminals are far away from the BS, subpaths associated with the user-location dependent scattering cluster also have small angular spread. For each link between the base station and the user, the number of scattering clusters that contributes to the channel responses is typically small. For users at different locations, if they can see the same scattering cluster, their channel responses will contain subpaths with similar angle-of-arrival (AOA)/angle-of-departure (AOD) which all concentrate around the LOS direction between the base station and that scattering cluster. This phenomenon is known as "shared scatterers" or "joint scatterers".

Motivated by these observations, in this thesis we exploit the idea of a low-dimensional representation for the massive MIMO channel and investigate the performance of algorithms for channel estimation which explicitly account for the observed low-dimensionality in the representation. Conceptually, we closely follow the dictionary learning based channel model (DLCM) proposed in [1], where a learned overcomplete dictionary, rather than a predefined basis or dictionary, is used to represent the channel. The motivation behind the dictionary learning based approach is that the dictionary, due to the learning process, is able to adapt to the cell characteristics as well as insure a sparse representation of the channel. Since no structural constraints are placed on the dictionary, the approach is applicable to an arbitrary array geometry and also does not need accurate array calibration. Being cell specific, a further obvious advantage of the dictionary learning process is that it adapts the channel model to the channel measurements collected in a cell. The sparse representation is also encouraged during the learning process. The learned overcomplete dictionary has the potential to exploit underlying low dimensionality through the learning process, and is robust to antenna array uncertainties and non-ideal propagation schemes. In particular, as the learned dictionary does not have any predefined structural constraints,

it is robust towards imperfections in the underlying physical generation scheme of the channel, and is expected to also work in the case when antenna gains and locations are different from the nominal values, or there exist near-field scattering clusters.

Once the channel has a sparse representation with respect to a learned dictionary, the problem of channel estimation can be defined as a sparse linear inverse problem. This allows us to utilize the compressed channel estimation framework to perform channel estimation. However, compared to the "more classical" problems of this type where the sparse representation is with respect to a known orthogonal basis, in this case we deal with the problem of estimating a vector which admits a sparse representation with respect to a learned, overcomplete dictionary. As result, it is required to modify existing algorithms such that they fit this particular estimation setup. As a particularly promising, we identify the approaches based on the framework of *deep learning*. Deep learning's success has been mostly driven by the state-of-the-art results it has accomplished in research areas such as computer vision or machine translation. When it comes to communication applications, deep learning has also focused on problems covering signal compression, compressed sensing and denoising [23] [24] [25] [26] [27] [28] Here we apply the deep learning framework to the problem of channel estimation in the dictionary learning based setup. In this context, neural networks will here be used to learn the unfolded versions of iterative algorithms such as approximate message passing (AMP) and iterative shrinkage-thresholding algorithms (ISTA) [14] [27]from data.

The rest of the thesis is organized as follows. After the Introduction, in Chapter 2 we describe the system model and introduce the dictionary learning based channel model. Furthermore, in this Chapter we provide the connection to general sparse linear inverse problems. In Chapter 3 we describe the Geometry-based Stochastic Channel Model (GSMC) used to generate channel samples in the FDD massive MIMO setup. In Chapter 4, we provide the theoretical background of compressed sensing and describe some classical algorithms for solving sparse linear inverse problems. In Chapter 5 we introduce the principles of deep learning algorithms with focus on the relevant aspects for our channel estimation problem at hand. In Chapter 6, we describe methods and techniques for unsupervised clustering. In Chapter 7 we experimentally evaluate different algorithms for channel estimation and representation in the dictionary learning based model. This chapter, which provides the core of the thesis, summarizes the

investigated performance of all approaches under consideration. Chapter 8 concludes the thesis.

2 System Model and Problem Description

By deploying a large antenna array at the base station (BS), the base station is able to perform both receive and transmit beamforming with narrow beams, thus eliminating multiuser interference and thereby increasing the cell throughput. For effective downlink beamforming, it is essential to have accurate knowledge of the downlink channel state information at the transmitter (CSIT). In a time-division duplexing (TDD) system, downlink CSI can be obtained by exploiting uplink/downlink channel reciprocity. The common assumption in Massive MIMO is that each user terminal (UT) only has a small number of antennas and that the BS can use uplink channel information, obtained from the relatively easy uplink training, for downlink beamforming [15]. On the other hand, as frequency-division duplexing (FDD) is generally considered to be more effective for systems with symmetric traffic and delay-sensitive applications, most cellular systems today employ FDD [15] [16]. Channel reciprocity is no longer valid in FDD systems and in order to obtain CSIT, the BS has to perform downlink training. Subsequently, the user needs to estimate, quantize and feedback the channel state information.

To deal with the limited downlink channel training interval in a FDD Massive MIMO system, one option is to explore the possible underlying channel structure whereby the high dimensional channel vector has a low dimensional representation [17] [18]. Motivated by the framework of Compressive Sensing (CS), if the desired signal (channel response) can be sparsely represented in some basis or dictionary, then the number of measurements (downlink training period) is proportional to the number of nonzero entries and the signal can be robustly recovered using sparse signal recovery algorithms [19] [20].

2.1 Sparse Channel Representation

The insight behind the dictionary learning approach is that the high dimensional data (channel response in our case) usually has some structure correlated in some dimensions, and the true degrees of freedom that generate the data is usually small. So by learning from large amount of data, we are able to recover useful underlying structures or models, thus making representation of the data more efficient for the desired application.

Consider a single cell downlink multiuser MIMO system with N antennas at the base station and K single-antenna users. The motivating assumption underneath the dictionary-learning based approach is that the channel vectors $\mathbf{h}_k \in \mathbb{C}^{N\times 1}$, $k \in [K]$, admit a *sparse* representation with respect to a dictionary $\mathbf{C}$, i.e.

$$\mathbf{h}_k = \mathbf{C}\mathbf{x}_k, \ \text{where} \|\mathbf{x}_k\|_0 < N. \tag{2.1}$$

In this regard, there are two important questions: first, does the underlying physical propagation environment support a sparse representation? Second, can we find a good sparsifying dictionary $\mathbf{C}$ such that the mathematical model is valid?

The sparse channel representation is supported by the observations made in [1], [21], [3], [22] for the Geometry-Based Stochastic Channel Model (GSCM) [3]. To summarize, the assumptions are that for a specific cell, the locations of the dominant scattering clusters are determined by cell specific attributes such as the buildings, terrain, etc. and are common to all the users irrespective of user position. Given that the scattering clusters are far away from the base station, the subpaths associated with a specific scattering cluster will be concentrated in a small range around the line of sight (LOS) direction between the base station and the scattering cluster, i.e. having a small angular spread (AS). Further, by assuming that the user terminals are far away from the base station, subpaths associated with the user-location dependent scattering cluster also have small angular spread. For each link between the base station and the user, the number of scattering clusters that contributes to the channel responses is typically small. For users at different locations, if they can see the same scattering cluster then their channel responses will contain subpaths with similar AOA/AOD which all concentrate around the LOS direction between the base station and that scattering cluster, a phenomenon known as "shared scatterers" or "joint scatterers".

These considerations, if valid, support the idea of a low dimensional representation for the large Massive MIMO channel.

2.2 Dictionary Learning for Channel Representation

Dictionary learning in the context of channel estimation in FDD massive MIMO systems has been addressed in [1]. Specifically, in [21], [18] and [29] the discrete Fourier transform (DFT) matrix has been used to sparsely represent the channel. The utilization of DFT basis is compatible with theoretical results of signal recovery in compressive sensing [19], and has been proposed as the virtual channel model [30] or angular domain channel representation [31]. However, the DFT basis is only valid for a uniform linear array (ULA), and can only lead to an approximate sparse representation with limited scattering and sufficiently large number of antennas [29], [30]. For practical channels, the DFT basis will often result in a large number of nonzero entries in the channel representation, which in turn requires a large number of training symbols for reliable channel estimation, thus losing the benefits of CS based channel estimation.

In order to accurately and sparsely represent the channel, a dictionary learning based channel model (DLCM) was proposed in [1], where a learned overcomplete dictionary, rather than a predefined basis or dictionary, is used to represent the channel. The dictionary, due to the learning process, is able to adapt to the cell characteristics as well as insure a sparse representation of the channel. Since no structural constraints are placed on the dictionary, the approach is applicable to an arbitrary array geometry and also does not need accurate array calibration. Being cell specific, a further obvious advantage of the dictionary learning process is that it adapts the channel model to the channel measurements collected in a cell. The sparse representation is also encouraged during the learning process. The learned overcomplete dictionary has the potential to exploit underlying low dimensionality through the learning process, and is robust to antenna array uncertainties and non-ideal propagation schemes. In particular, as the learned dictionary does not have any predefined structural constraints, it is robust towards towards imperfections in the underlying physical generation scheme of the channel, and is expected to also work in the case when antenna gains and locations are different from the nominal values, or there exist near-field scattering clusters.

The dictionary-learning problem has been formalized in [1]. For completeness, we

provide a summary in the following. Assuming we collect L channel measurement vectors as training samples in a specific cell, the goal is to learn the dictionary $\mathbf{C} \in \mathbb{C}^{N\times M}$ (where $N < M$ to benefit from overcompleteness), such that for all channel responses $\mathbf{h}_i, i = 1, \ldots, L$, they can be approximated as $\mathbf{h}_i \approx \mathbf{C}\mathbf{x}_i$, $\mathbf{x}_i \in \mathbb{C}^{M\times 1}$. The dictionary learning algorithm should in general be able to address model fitting $\|\mathbf{h}_i - \mathbf{C}\mathbf{x}_i\|_2$ (accuracy), and encourage small $\|\mathbf{x}_i\|_0$ (efficiency) for sparse representation. If one constrains the model mismatch error, then the dictionary learning problem can be formulated as

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_L} \frac{1}{L}\sum_{i=1}^{L} \|\mathbf{x}_i\|_0 \tag{2.2}$$

$$\text{s.t. } \|\mathbf{h}_i - \mathbf{C}\mathbf{x}_i\|_2 \leq \eta, \ \forall i \tag{2.3}$$

and $\|\mathbf{C}_{.j}\|_2 \leq 1, \ \forall j = 1, \ldots, M$ in order to prevent ambiguity between $\mathbf{C}$ and $\mathbf{x}$. The solved $\mathbf{C}$ in (2.3) leads to the sparsest representation in the sense of all channel measurements, given the model mismatch tolerance η.

Two similar formulations could alternatively be used. If one knows beforehand or wants to constrain the sparsity level of each coefficient $\mathbf{x}_i$, then one solves:

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_L} \frac{1}{L}\sum_{i=1}^{L} \frac{1}{2}\|\mathbf{h}_i - \mathbf{C}\mathbf{x}_i\|_2^2 \tag{2.4}$$

$$\text{s.t. } \|\mathbf{x}_i\|_0 \leq T_0, \ \forall i \tag{2.5}$$

where T_0 constrains the number of non-zero elements in each $\mathbf{x}_i$. In other words, one expects that every channel measurement can be represented using T_0 atoms from the learned dictionary, and one solves for the dictionary that minimizes model mismatch using channel response samples. If no explicit constraints are posed on the model fitting error or sparsity level, the dictionary learning process can be formulated in a general form as

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_L} \frac{1}{L}\sum_{i=1}^{L} \frac{1}{2}\|\mathbf{h}_i - \mathbf{C}^d\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{x}_i\|_0 \tag{2.6}$$

where λ is the parameter that trades off the data fitting and sparsity. To solve the dictionary learning problem, in 7.3 block coordinate descent framework has been applied where each iteration includes alternatively minimizing with respect to either $\mathbf{C}$ or $\mathbf{x}_i$, $\forall i$, while keeping the other fixed. In 7.3 it has been shown experimentally that starting from a reasonable dictionary, e.g. an overcomplete DFT dictionary, the

learning algorithm will lead to a dictionary that improves the performance in terms of both sparse representation and sparse recovery.

2.3 Dictionary Learning for Channel Estimation

Once the channel has the sparse representation, the compressed channel estimation framework can be utilized to reduce the amount of downlink training. Here we address a particular implementation of downlink channel estimation with *explicit feedback*. According to this approach, the users feed back (quantized) received training signals to the base station and channel is recovered at the base station. This is different from the conventional channel estimation schemes where the users estimate the channel (or calculate precoding matrices), and report feedback to the base station in the form of channel quality indicator(CQI) and precoding matrix indicator (PMI) (as e.g. in LTE). Such scheme has several advantages: firstly the sparse recovery process requires complicated computation so it is preferably done at the base station thus saving energy for user terminals. Secondly, the dictionary is learned and stored at the base station, making it available to all users which otherwise involves significant overhead in storage at the user equipment and conveyance of dictionary. Furthermore, such overhead will be incurred every time user enters a new cell. By assigning the recovery operation to the base station, one avoids such overhead. In addition, since dimension of received symbol T is less than the channel dimension N in a Massive MIMO system, the scheme also reduces feedback overhead.

3 Geometry-Based Stochastic Channel Model (GSCM)

3.1 Introduction and motivation

In this chapter, we summarize the *geometry-based stochastic channel model* (GSCM) [3] [4] used to generate the channel observations.

GSCM is widely adopted because it allows us to specify the average power delay profile (PDP), the angular power spectrum (APS) and their statistical distributions. Moreover, GSCM exhibits several additional advantages, specifically: i) it has an obvious relation to the real environment e.g., locations and parameters of scatters; ii) it shows any possible correlation between PDP and APS; iii) the stochastic model includes the movement of mobile stations and scatterers (if they are moving objects); iv) it easily implements shadowing, and the appearance and disappearance of transmission paths (e.g. because of blocking by obstacles);

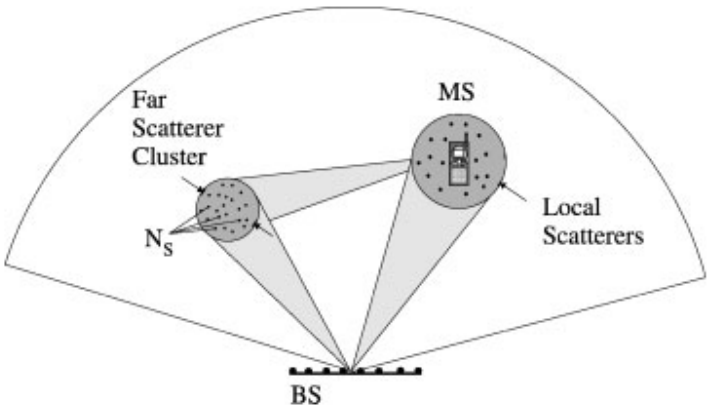


Figure 3.1: Principle of the GSCM [3]

3.2 Generating User Parameters for the GSCM

In GSCM, the channel model is generated by dividing the simulation in two levels: *link-level* and *system-level* simulations. In the link-level simulations one considers a single base station transmitting to a signal mobile station. In the system level simulations one considers a multiple BSs with multiple sectors and multiple MSs. The interface between the two is provided by a look-up table.

The overall procedure for generating the channel matrices consists of three basic steps, see figure 3.2:

1. Specify an environment, either suburban macro, urban macro or urban micro:
   a) Suburban macrocell (approximately 3 km distance BS to BS);
   b) Urban macrocell (approximately 3 km distance BS to BS) (We choose this scenario for our channel model);
   c) Urban microcell (less than 1 km distance BS to BS);
2. Obtain the parameters to be used in simulations, associated with that environment;
3. Generate the channel coefficients based on the parameters;

Generating user parameters for urban macrocell

In table 3.1, we can find matching parameters for our model [4]:

1. Placement of the MS with respect to each BS in order to compute path loss.
2. Determine random delays for each of the N multipath components. N=6 for macrocell environments.
3. Determine random average powers for each of the N multipath components.

$$P'_n = \exp \frac{(1 - r_{DS}).(\tau'_n - \tau'_1)}{r_{DS}.\sigma_{DS}}.\ 10^{\frac{-\xi}{10}}\ , n = 1, ..., 6 \tag{3.1}$$

where ξ_n are i.i.d Gaussian random variables with standard deviation $\sigma_{RND} = 3dB$, which is which is a shadowing randomization effect on the per-path powers.

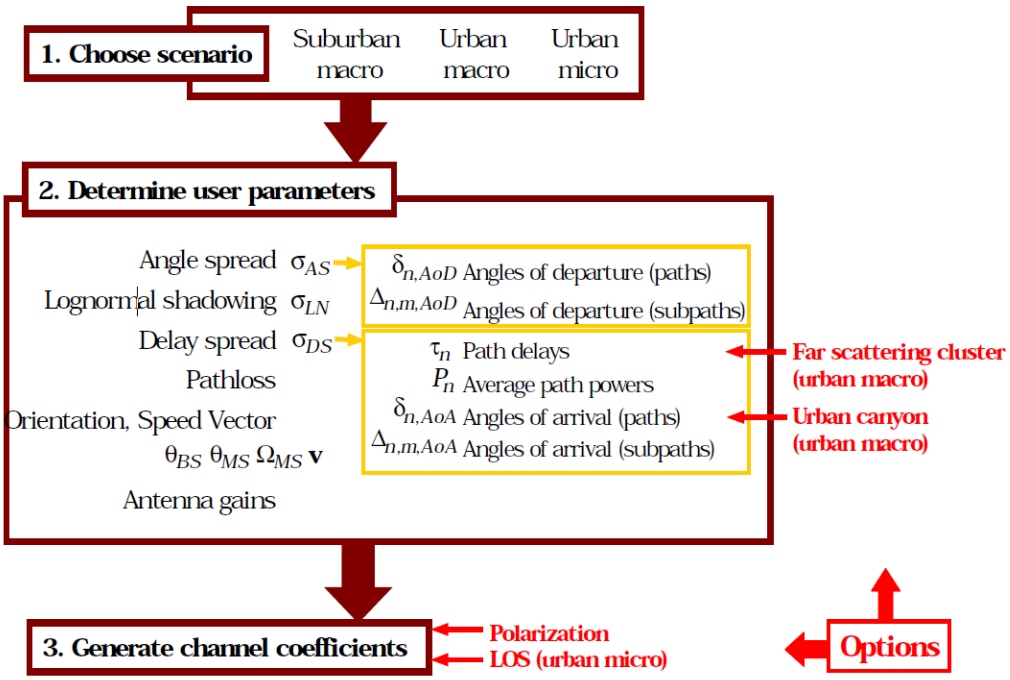


Figure 3.2: Channel model overview for simulations [4]

4. Determine AoDs for each of the N multipath components. First generate i.i.d. zero-mean Gaussian random variables:

$$\delta' = r_{AS}.\sigma_{AS}, \ n = 1, ..., N, \tag{3.2}$$

where $\sigma_{AoD} = r_{AS}\sigma_{AS}$. The value rAS is given in table 3.1

5. Associate the multipath delays with AoDs. The nth delay τ_n generated in Step 3 is associated with the nth AoD $\delta_{n,AOF}$ generated in step 6.

6. Determine the powers, phases and offset AoDs of the $M = 20$ sub-paths for each of the N paths at the BS.

7. Determine the AoAs for each of the multipath components. The AoAs are i.i.d. Gaussian random variables:

$$\delta_{n,AoA} \sim \eta(0, \sigma^2_{n,AoA}) \tag{3.3}$$

where $\sigma^2_{n,AoA} = 104.12(1 - exp(-0.2175\,|10log_{10}(P_n)|)$

8. Determine the offset AoAs from the table 3.1

Channel Scenario	Suburban Macro	Urban Macro	Urban Micro
Number of paths (N)	6	6	6
Number of sub-paths (M) per-path	20	20	20
Mean AS at BS AS at BS as a lognormal RV $\sigma_{AS} = 10^{\epsilon_{AS}x+\mu_{AS}}, x \sim \eta(0,1)$	$E(\sigma_{AS}) = 5^0$ $\mu_{AS} = 0.69$ $\varepsilon_{AS} = 0.13$	$E(\sigma_{AS}) = 8^0$, 15^0 8^0 $\mu_{AS} = 1.18$ $\epsilon_{AS} = 0.34$ 15^0 $\mu_{AS} = 1.18$ $\epsilon_{AS} = 0.210$	$NLOS : E(\sigma_{AS}) = 19^0$ N/A
$r_{AS} = \frac{\sigma_{AoD}}{\sigma_{AS}}$	1.2	1.3	N/A
Per-path AS at BS (Fixed)	2 deg	2 deg	5 deg (OS and NLOS)
BS per-path AoD $\eta(0, \sigma^2_{AoD})where$ Distribution standard distribution	$\eta(0, \sigma^2_{AoD})$	$U(-40deg, 40deg)$ $\sigma_{AoD} = r_{AS}\sigma AS$	 $\sigma_{AoD} = r_{AS}\sigma AS$
Mean AS at MS	$E(\sigma_{AS,MS}) = 68^0$	$E(\sigma_{AS,MS}) = 68^0$	$E(\sigma_{AS,MS}) = 68^0$
Per-path AS at MS (fixed) MS Per-path AoA Distribution Delay spread as a lognormal RV	35^0 $\eta(0, \sigma^2_{AoA}(Pr))$ $\mu_{DS} = -6.18$ $\epsilon_{DS} = 0.288$	35^0 $\eta(0, \sigma^2_{AoA}(Pr))$ $\mu_{DS} = -6.18$ $\epsilon_{DS} = 0.288$	35^0 $\eta(0, \sigma^2_{AoA}(Pr))$ N/A
Mean total RMS Delay Spread	$E(\sigma_{DS} = 0.17\mu S)$	$E(\sigma_{DS} = 0.65\mu S)$	$E(\sigma_{DS} = 0.251\mu S)$
$r_{DS} = \frac{\sigma_{delays}}{\sigma_{DS}}$	1.4	1.7	N/A
Distribution for path delays			$U(0, 1.2\mu)$
Lognormal shadowing standard deviation,σ_{SF}	$8db$	$8db$	NLOS: 10 dB, LOS :4dB
Pathloss model (dB), d is in meters	$31.5 + 35\log_{10}(d)$	$31.5 + 35\log_{10}(d)$	NLOS: 34.53+ $38\log_{10}(d)$ LOS: 30.18+ $26\log_{10}(d)$

Table 3.1: Environment parameters [4]

9. Determine the antenna gains of the BS and MS sub-paths as a function of their respective sub-path AoDs and AoAs. For the nth path, the AoD of the mth sub-path (with respect to the BS antenna array broadside) is

$$\Theta_{n,m,AoA} = \Theta_{BS} + \delta_{n,AoA} + \Delta_{n,m,AOD}\Theta_{n,m,AoA} = \Theta_{MS} + \delta_{n,AoA} + \Delta_{n,m,AOD} \tag{3.4}$$

10. Apply the path loss based on the BS to MS distance from Step 2, and the log normal shadow fading determined in step 3 as bulk parameters to each of the sub-path powers of the channel model.

4 Compressed Sensing and Linear Inverse Problems

4.1 Introduction

In this chapter we summarize the preliminaries related to the compressed sensing framework used in the thesis. The summary and the notation are based on [5].

The work of Nyquist, Shannon, and Whittaker on sampling continuous-time band-limited signals leaded to a digital revolution [5]. According to that framework, we can recover signals, images and videos if samples are taken from a set of uniformly spaced samples, known as *Nyquist rate*. However, in some certain applications, we need a huge amount of data ends up with far too many samples. That means in certain domain, applying Nyquist rate is too costly or physically impossible.

Compressed sensing (CS) could surpass the limit of sampling theory, and recently found its application in a wide range of areas, including electrical engineering and applied mathematics. According to the compressed sensing framework, only a few non-zero coefficients might be enough to reconstruct reliably a signal (video, image etc.), given that the signal has a sparse representation in a certain basis, or a dictionary. CS enables a potentially large reduction in the sampling and computation costs for sensing signals that have a sparse or compressible representation. By means of CS, it is possible to reduce the sampling and computation costs for any signal that has a sparse representation.

4.2 Preliminaries

In the following, we present some preliminaries necessary for the understanding of the CS framework, and for the description of the algorithms applied for the communication problems addressed in this thesis.

4.2.1 Vector Spaces:

A vector space X is a non-empty collection of elements (vectors) on which two operations are defined.

Let X be a vector space and F scalar field $X \times X \to X,\ F \times X \to X$:

1- Addition : $\forall\ \mathbf{x}, \mathbf{y} \in X$ there is $\mathbf{x}+\mathbf{y} \in X$.

2- Scalar Multiplication: $\forall\ \mathbf{x} \in X$ and $\forall\ \alpha \in \mathbb{F}$, $\alpha \cdot \mathbf{x} \in X$.

So that for each pair of elements $\mathbf{x}$,$\mathbf{y}$ in X there is a unique element $\mathbf{x}+\mathbf{y}$ is defined as the sum of $\mathbf{x}$ and $\mathbf{y}$ in X, and for each scalar multiplication $\alpha \cdot \mathbf{x}$ exists a unique element $\alpha\mathbf{x}$ defined as the scalar multiple of $\mathbf{x}$ by α in X.

The two vector space operations satisfy common axioms

1. Commutativity of addition: for each pair of elements: $\mathbf{x}+\mathbf{y} \in X, \mathbf{x}+\mathbf{y} = \mathbf{y}+\mathbf{x}$.
2. Associativity of addition: $\forall\ \mathbf{x}, \mathbf{y}, \mathbf{z} \in X,\ (\mathbf{x}+\mathbf{y})+\mathbf{z} = \mathbf{x}+(\mathbf{y}+\mathbf{z})$.
3. Identity element of addition: there exists an element $0 \in X$, called the zero vector, so that: $\mathbf{x}+0 = \mathbf{x},\ \ \forall\ \mathbf{x} \in X$.
4. Inverse elements of addition: for each element $\mathbf{x} \in X$ there exists an element $\mathbf{y} \in X$ such that $\mathbf{x}+\mathbf{y} = 0$.
5. Compatibility of scalar multiplication with field multiplication: for each scalar α, β and each $\mathbf{x} \in V$: $(\alpha(\beta)\mathbf{x}) = \alpha((\beta\mathbf{x}))$.
6. Identity element of scalar multiplication: for each element $\mathbf{x} \in X,\ 1\mathbf{x} = \mathbf{x}$ 1 is defined as the multiplicative identity in F.
7. Distributivity of scalar multiplication with respect to vector addition: for each pair of scalars α, and each $\mathbf{x} \in X$: $\alpha(\mathbf{x}+\mathbf{y}) = \alpha\mathbf{x}+\alpha\mathbf{y}$.
8. Distributivity of scalar multiplication with respect to field addition: for each pair of scalars α,β and each $\mathbf{x} \in X$: $(\alpha+\beta)\mathbf{x} = (\alpha\mathbf{x}+\beta\mathbf{x})$.

Normed vector Spaces: Let X be a vector space, a norm over X is a real-valued function that maps each vector $\mathbf{x} \in X$ into a real number and is denoted by $\|\mathbf{x}\|$, the norm satisfies the following axioms

1. The zero vector, 0, has a length of zero; every other vector has a positive length: $\|\mathbf{x}\| \geq 0$ and $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = 0$.

2. Multiplying a vector by a positive number changes its length without changing its direction: $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ for any scalar α and any $\mathbf{x} \in X$.

3. The triangle inequality is taking norms as distances: $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\|+\|\mathbf{y}\|$ $\forall \mathbf{x}, \mathbf{y} \in X$.

The normed linear vector space on X is the vector space X together with its norm $\|.\|$.

Inner Product Spaces: Let X be a vector space and let $\mathbf{x}, \mathbf{y}$ be arbitrary elements in X, and F is either the field of real numbers $\mathbb{R}$ or the field of complex numbers $\mathbb{C}$. The *inner product* on X is a function that assigns to every ordered pair of vectors $\mathbf{x}$ and $\mathbf{y}$ a scalar value which is a map X x $X \rightarrow F$ and denoted by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i. \tag{4.1}$$

The inner product satisfies the following axioms $\forall$ $\mathbf{x}, \mathbf{y}, \mathbf{z} \in X$ and $\alpha \in F$:

1. $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ and $\langle \mathbf{x}, \mathbf{x} \rangle = 0$ if and only if $\mathbf{x}$=0.

2. $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x} + \mathbf{z} \rangle + \langle \mathbf{y} + \mathbf{z} \rangle$

3. $\langle \alpha\mathbf{x}, \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle$

4. $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle}$ where ($\overline{\mathbf{x}}$ is a complex conjugate of $\mathbf{x}$).

The requirement 1 defines the positive definiteness, 2 and 3 define the linearity in the first component and 4 defines the conjugate symmetry.
A vector space together with its inner product is an *inner product space*.

Complete Vector Spaces: A sequence $\{x_n\}$ in a normed space is defend as a *Cauchy sequence* if

$$\|\mathbf{x}_m - \mathbf{x}_n\| \rightarrow 0 \text{ as } n, m \rightarrow \infty.$$

A Banach space is a vector space $X \in \mathbb{R}$ or $X \in \mathbb{C}$, which is equipped with a norm and is complete with respect to that norm, that is to say, for every Cauchy sequence $\{\mathbf{x}_n\}$ in X, there exists a limit in X,

$$\sum_{n=1}^{\infty} \|x_n\|_X < \infty \quad \Rightarrow \quad \sum_{n=1}^{\infty} x_n \text{ converges in } X.$$

A vector space is called complete if every Cauchy sequence has a limit (and therefore convergent) in this vector space.

A *Hilbert space* denoted as $\mathcal{H}$ is a complete linear vector space with respect to the norm produced by the inner product which is defend on $X \times X$

For all $\mathbf{x}, \mathbf{y} \in \mathcal{H}$ we have:

$$\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| + \|\mathbf{y}\|, \tag{4.2}$$

where, $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ is a norm on $\mathcal{H}$ and the equality holds if and only if $\mathbf{x} = \alpha \mathbf{y}$ for some $\alpha \in \mathbb{F}$ or $\mathbf{y} = 0$, The inner product is bounded if both $\|\mathbf{x}\|$ and $\|\mathbf{y}\|$ are bounded, and several normed vector spaces can be converted to Hilbert spaces by defining an appropriate inner product.

Examples:

1. the Euclidean spaces R^n:

   The inner product is defined on the Euclidean space as:

   $$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i, y_i, \tag{4.3}$$

   $\forall\ \mathbf{x} = (x_1, x_2, ...x_n)$, $\mathbf{y} = (y_1, y_2, ...y_n) \in \mathbb{R}^{\kappa}$.

   The resulting Euclidean norm is defined as:

   $$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{\infty} |\mathbf{x}_i|^2} \ \ \forall\ \mathbf{x} \in \mathbb{R}^{\kappa}. \tag{4.4}$$

   The space $\mathbb{R}^{\kappa}$ is complete with respect to the Euclidean norm, thus the Euclidean space $\mathbb{R}^{\kappa}$ is a Hilbert space.

2. l^2 spaces: The inner product is defined on the l^2 space as:

   $$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i, \overline{y_i}, \tag{4.5}$$

   where $\langle \mathbf{x}, \mathbf{y} \rangle < +\infty$ as $\|\mathbf{x}\| < +\infty$ and $\|\mathbf{y}\| < +\infty$.

   The resulting norm in l^2 is:

   $$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{\infty} |\mathbf{x}_i|^2} \ \ \forall \mathbf{x} \in l_2. \tag{4.6}$$

   The l^2 space is also complete with respect to the l^2 norm thus the l_2 space is a Hilbert space.

3. L^2 spaces: let $\mathbf{f} = f(t),\ \mathbf{g} = g(t) \in L^2$ .The inner product is defined on the L^2 space as:

$$\langle f(t), g(t) \rangle = \int_a^b f(t)\overline{g(t)}dt, \tag{4.7}$$

where $\langle f, g \rangle < +\infty$.

The resulting norm in L^2 is:

$$\|\mathrm{f}\|_2 = \Big(\int_{-\infty}^{+\infty} |f(t)|^2 \mathrm{d}t\Big)^{1/2} \forall\ \mathrm{f} = f(t) \in L^2. \tag{4.8}$$

The L^2 space is also complete with respect to the L^2 norm, thus the L^2 space is a Hilbert space.

4.2.2 Basis:

A set $\phi_i{}_{i=1}^n$ is called a basis for R^n if the vectors in the set span R^n and linearly independent. This implies that each vector in the space has a unique representation as a linear combination of these basis vectors. Specifically, for any $x \in R_n$, there exist coefficients $x = \sum_{i=1}^n c_i\phi_i$. Where $\phi \in R^{nxn}$. We formally formulate this description using a matrix representation, so we can write:

$$x = \phi c \tag{4.9}$$

Each orthonormal basis fulfills the following:

$$\langle \phi_i, \phi_j \rangle = \begin{cases} 1, & i = j. \\ 0, & i \neq j. \end{cases} \tag{4.10}$$

We can easily calculate the coefficients c, if there is an orthonormal basis is known:

$$c = \phi x \tag{4.11}$$

That is easy to proof because $\phi^T\phi = I$, where I is the identity matrix.

4.2.3 Frames:

A frame is a set of vectors $\phi_{i=1}^n$ in R^d, $d < n$, which fulfills:

$$A\|x\|_2^2 \leqslant \left\|\phi^T x\right\|_2^2 \leqslant B\,\|x\|_2^2 \tag{4.12}$$

With $0 < A \leqslant B < \infty$, the equation (4.12) represents the frame bounds.

$$\phi = \begin{cases} \text{is A-tight} & if A = B \\ \text{is Parseval frame} & if A = B = 1 \\ \text{is equal-norm} & if \lambda > 0 and \left\|\phi\right\|_2 = \lambda \end{cases}$$

Frames can provide richer representations of data since, due to their redundancy for a given signal x, there exists infinitely many coefficient vectors c such that $x = \phi c$. In order to obtain a set of feasible coefficients, we exploit the dual frame. Specifically, any frame satisfying $\Phi\widetilde{\Phi}^T = \widetilde{\Phi}\Phi^T = I$ is called dual frame. The particular choice $\widetilde{\Phi} = (\Phi\Phi^T)^{-1}\Phi$ is referred to as the canonical dual frame (known as Moore-Penrose pseudoinverse).

4.2.4 Low-Dimensional Signal Models

Sparsity and nonlinear approximation: Signals are usually approximated a linear combination of just a few elements from a known basis or dictionary. Sparse signal models provide a mathematical framework where little information (compared to the original high-dimensional representation) suffices to reliably represent a known signal. Mathematically, we say that a signal x is k-sparse when it has at most k non-zeros, i,e $\left\|x\right\|_0 \leq k$

$$\sum_k = x : \left\|x\right\|_0 \leq k \tag{4.13}$$

However, we face signals that are not sparse, but they have a sparse representation in a certain basis Φ. In this case, we still say that x is a k-sparse under the following assumption: $x = \Phi c$ where $\left\|c\right\|_0 \leq k$.

The application of sparsity cover a wide spectrum of domains in signal processing and approximation theory for tasks such as compression and denoising, and in statistics and learning theory as a method for avoiding over-fitting.

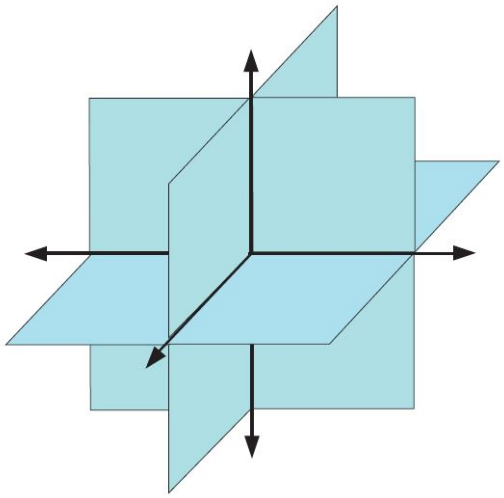


Figure 4.1: Union of sub-spaces defined by $\sum_2 \subset R^2$, i.e., the set of all 2-sparse signals in R^3. [5]

Geometry of sparse signals: Sparsity is non linear model because each dictionary changes from signal to another. We can observe it by considering two k-sparse signals $x, z \in \sum_k$. The linear combination of $x + z$ maybe not belong to $\sum_k$ because their Support could not coincide. Therefore, we say that set of sparse signals $\sum_k$ does not form a linear space. Instead it consists of the union of all possible $\binom{n}{k}$. In 4.1 figure , there is an illustration of this example .

Compressible signals: In practice, only few signals are truly sparse but they are compressible. Each compressible signal can be well approximated to a sparse one. We call these signals "approximately or relatively sparse". Moreover, we can quantify the compressibility by calculating the error incurred by approximating a signal x by some $\hat{x} \in \sum_k$:

$$\sigma_k(x)_p = \min_{\hat{x} \in \sum_k} = \|x - \hat{x}\|_p \,, \tag{4.14}$$

if $x \in \sum_k$ then clearly $\sigma_k(x)_p = 0, \ \forall p$.

4.2.5 Convexity

In many cases we would like to optimize a given function f where f is defined as $f : \mathbb{R}^n \to \mathbb{R}$. We can optimize f by finding $x \in \mathbb{R}^n$ that minimizes or maximizes f. It is framed as optimization problems. It turns out that, minimizing or maximizing a

function in order to find the global optimum is not an easy task. But we can, in many cases, efficiently find the global optimum for a special class of optimization problems known as convex optimization [6].

Convex sets: We say a set C is a convex set, if $x, y \in C$, $\forall\ x$ and $\forall\ y$, and $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$, $\theta x + (1-\theta)y \in C$. This means that if we take any two elements in C, and draw a line segment between these two elements, then every point on that line segment also belongs to C, see figure 4.2.

(a) (b)

Figure 4.2: Examples of a convex set (4.2a), and a non-convex set (4.2b. [6]

Convex functions: A function f is convex if its domain (denoted $\mathbb{D}$) is a a convex set, and if, for all $x, y \in \mathbb{D}(f)$ and $\theta \in \mathbb{R}$, $0 \leq \theta \leq 1$,

$$f(\theta x + 1(1-\theta)y) \leq \theta f(x) + (1+\theta)f(y).$$

We interpret figure 4.3 as following:

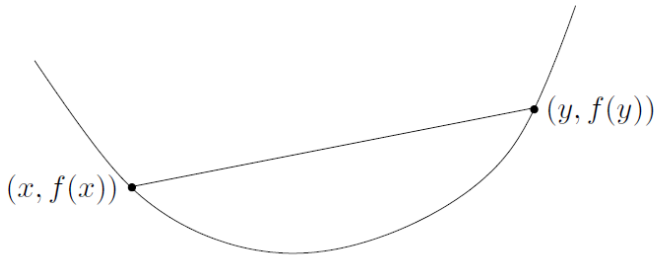


Figure 4.3: Graph of a convex function [6]

if we pick any two points on the graph of a convex function and draw a straight line between then, then the portion of the function between these two points will lie below this straight line.

We say a function g is strictly convex if its domain (denoted $\mathbb{D}$) is a a convex set, and if, for all $x, y \in \mathbb{D}(f)$ and $\theta \in \mathbb{R}$, $0 < \theta < 1$ and $x \neq y$,

$$f(\theta x + 1(1-\theta)y) \leq \theta f(x) + (1+\theta)f(y).$$

If f is (strictly) convex then $-f$ is (strictly) concave [6].

4.3 Sparse Linear Inverse Problems

4.3.1 Introduction

Consider the situation where we want to recover a signal $S^0 \in \mathbb{R}^N$ from a noisy linear measurement $y \in \mathbb{R}^M$ of the form

$$y = \Phi\ S^0 + w \tag{4.15}$$

where $w \in \mathbb{R}^M$ is additive white Gaussian noise (AWGN) and $M \ll N$. Moreover, we assume that the signal vector S^0 has a sparse representation in a known orthonormal basis $\Psi \in \mathbb{R}^{N \times N}$. Then we can write down

$$y = Ax^0 + w, \tag{4.16}$$

where $A \stackrel{\triangle}{=} \Phi\Psi$ and Ψ is an orthonormal basis. Then we seek to recover a sparse vector x^0 from y. x^0 is a sparse vector. Furthermore, MSs need to estimate x^0 based on the downlink training measurement. This problem is known as the sparse linear inverse problem. There are several methods to solve it by deploying the convex optimization framework. We will introduce a couple of them in the following sections.

4.3.2 Least Absolute Shrinkage and Selection Operator (LASSO) Regression

In the following we describe the details of the LASSO algorithm. The exposition follows [7].

Definition: assume we have training data of the form $(\boldsymbol{x}^i, y_i)$ where $i = 1, 2, ..., N$ and $\boldsymbol{x}^i = (x_{i1}, ..., x_{ip})^T$ are the predictor variables, y_i are the responses and the observations(y_is) given (x_is) are independent.
LASSO minimizes the residual sum of squares subject to the sum of the absolute value

of coefficients being less than constant. And $\sum_i \frac{x_{ij}}{N} = 0$,$\sum_i \frac{x_{ij}^2}{N} = 1$. LASSO estimate $(\hat{\alpha}, \hat{\beta})$, by Letting $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_p)^T$. LASSO are defined by:

$$(\hat{\alpha}, \hat{\beta}) = \text{argmin}\left\{ \sum_{i=1}^{N} (y_i - \alpha - \sum_j \beta_j x_{ij})^2 \right\} \qquad \text{subject to} \sum_j |\beta_j| \leq t. \tag{4.17}$$

Where $t \geq 0$ is a tuning parameters and controls the amount if shrinkage that is applied to the estimates. We can assume without loss of generality that $\bar{y} = 0$ [7]. Equation 4.17 is a quadratic programming problem with linear inequality constraints. We introduce a method to compute it in the following section.

Geometry of the LASSO: The elliptical contours of this function are shown by the red solid curves in 4.4 in case β_1, β_2. The constraint region is the rotated blue square. The LASSO solution is the first place where the contours touch the square and sometimes touches at the corners corresponding to a zero coefficient.

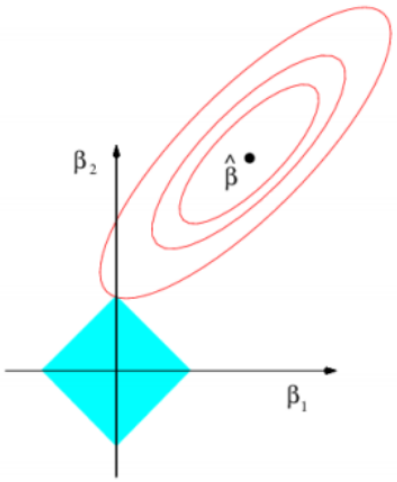


Figure 4.4: Estimation for Lasso [7]

LASSO Algorithm: Let $g(\beta) = \sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2$, and let $\delta_i, i = 1, 2, ..., 2^p$ be the $p-$tuples of the form $(\pm 1, \pm 1, ..., \pm 1)$. Then the condition $\sum |\beta_j| \leq t$ is equivalent to $\delta_i^T \beta \leq t$ for all i. For a given β, let $E = \{i : \delta_i^T \beta = t\}$ and S is an equality set and $S = \{i : \delta_i^T \beta < t\}$ [7]

LASSO Algorithm [7]

- Start with $E = \{i_0\}$ where $\delta_{i_0} = \text{sign}(\hat{\beta}^0)$, where $\hat{\beta}^0$ overall least squares estimate.
- Find $\hat{\beta}$ to minimize $g(\beta)$ subject to $G_E\beta \leq t$.
- While $\left\{ \sum \left|\hat{\beta}_j\right| > t \right\}$
- add i to the set E where $\delta = \text{sign}(\hat{\beta})$.
- Find $\hat{\beta}$ to monimze $g(\beta)$ subject to $G_E\beta \leq t$.

4.3.3 Iterative Algorithms

In this section, we propose different methods to solve linear inverse problem through solving convex optimization problem which has the shape:

$$\hat{x} = \underset{x}{\text{argmin}} \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1 \tag{4.18}$$

Where $\lambda > 0$ is a tunable parameter that controls the trade-off between sparsity and measurement fidelity in $\hat{x}$. The convexity of 4.18 leads to provably convergent algorithms and bounds on the performance of the estimate $\hat{x}$. We refer 4.18 as ℓ_1 problem.

ISTA one of the simplest approaches to solve 4.18 which itertes the steps (for $t = 0, 1, 2, ...$ and $\hat{x}_0 = 0$)

$$v_t = y - A\hat{x}_t \tag{4.19}$$

$$\hat{x}_{t+1} = \eta(\hat{x} + \beta A^T v_t; \lambda), \tag{4.20}$$

where β is a step-size, v_t is the iteration t residual measurement error and $\eta(\cdot, \lambda) : \mathbb{R}^N \to \mathbb{R}^N$. Moreover, η is a soft thresholding shrinkage function, defined as :

$$[\eta(r; \lambda)] \stackrel{\triangle}{=} sgn(r_j) max \{|r_j| - \lambda, 0\} \tag{4.21}$$

Approximate Message Passing (AMP): Approximate message passing (AMP) [32] is a message passing algorithm for linear inverse problems that are casted as factor graphs. Due to its lower computational complexity, AMP and its variants have become pragmatic alternatives to belief propagation (BP) and expectation (EP) algorithms for large-scale estimation and detection problems that can be cast as linear inverse problems, e.g., [14] [26]. AMP and its variants are based on the loopy BP algorithm for factor-graphs, in which factor measurements depend only weakly on a large number of random variables. In addition, AMP methods allow a rigorous analysis in the high-dimensional system limit.

AMP has initially been introduced in the context of reconstruction of sparse vectors $x \in \mathbb{R}^N$ from a (small) vector of linear observations. For the linear mixing model (4.16), [33] suggested the following first order AMP algorithm for reconstructing x given A and y. Start with an initial guess x^0 and proceed by

$$\begin{aligned} x^{t+1} &= \eta_t(A^* z^t + x^t), \\ z^t &= y - Ax^t + \frac{1}{\delta}\delta \left\langle \eta'_{t-1}(A^* z^{t-1} + x^{t-1}) \right\rangle \end{aligned} \tag{4.22}$$

for an appropriate sequence of nonlinear functions $\{\eta_t\}_{t\geq 0}$. For a large class of random matrices $\mathbf{A}$, the behavior of the AMP algorithm can be accurately described using the "state evolution" (SE) formalism. The sparsity-undersampling tradeo-ff of AMP for an appropriate choice of the functions η_t can compete with the one of convex optimization approaches, however at a lower computational complexity which is in the order of the simplest greedy algorithms. The above mentioned equations may be considered as a modification of the iterative thresholding algorithm for compressed sensing reconstruction. The equations can also be cast as messages from the Sum-Product formulation of loopy BP for factor graphs [34]. For all $i, j \in [N]$ and $a, b \in [n]$ (here $[N] = \{1, 2, ..., N\}$) start with messages $x^0_{j\to a} = 0$ and proceed by

$$\begin{aligned} z^t_{a\to i} &= y_a - \sum_{j\in[N]\setminus i} A_{aj} x^t_{j\to a} \\ x^{t+1}_{i\to a} &= \eta_t \left(\sum_{b\in[n]\setminus a} A_{bi} z^t_{b\to i} \right). \end{aligned} \tag{4.23}$$

The important difference between AMP and iterative thresholding in general is the last term on the right-hand side of the AMP equations, which in statistical physics is known as the "Onsager reaction term"

5 Deep Neural Networks (DNN):

5.1 Introduction Of Deep Neural Networks:

This section discusses the motivations and principles regarding learning algorithms for deep architectures. And in this essay, the following synonyms for the deep learning will be used interchangeably "deep structured learning" and "hierarchical learning". **Deep learning** is a sub branch of machine learning to predict, classify or make a decision based on data without hand-crafted programming for each details. If the neural networks (NN) that have a sufficient number of hidden layers can be regraded as deep learning, whereby a complex function is approximated through a composition of simple and predefined operations of units (or neurons). The operations performed are usually defined by a weighted combination of a specific group of hidden units with a **non-linear activation function**, depending on the structure of the model. Such operations along with the output units are named "layers" [35]. Experimental proce-

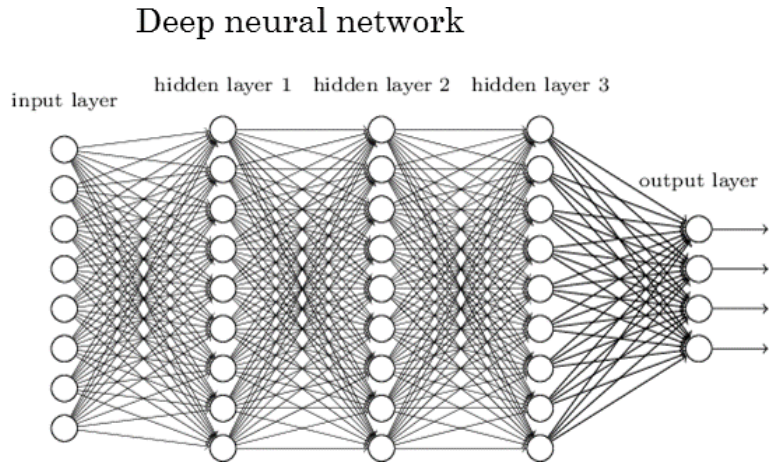


Figure 5.1: Deep neural network [8]

dures shows that in order to computers to mimic the human perceptions (for instance vision) which naturally deploy hierarchical structures. (e.g. seeing implies an analysis of the perceptive object, mouth, hair... etc. for each object there is a different layer).

To put it in other words, imitation of the biological observation and AI-level tasks demand hierarchical structure [36]. A breakthrough in 2006 had been made by professor Hinton who introduced back then the concept of Deep belief networks which can be trained without labeled data (unsupervised learning), the improvement of GPUs abilities (processing power) and decrease of its prices had allowed as well deep learning to make its comeback. Nowadays deep learning is used in different applications: as an example classification, regression, dimensionality reduction, robotics... etc [36].

5.2 A Three-Way Categorization

Depending on how the architectures and techniques are intended for use, e.g., synthesis/generation or recognition/classification, one can broadly categorize most of the work in this area into three major classes [37]:

1. **Deep networks to deal with unsupervised learning or generative learning:**
   Building deep nets depends only on the input signals (training sets without any class) without taking into the consideration the supervision information e.g. of RBMs, DBNs And auto-encoders. This class can be represented graphical model.

2. **Deep networks to deal with supervised learning or discriminative learning:**
   Due to the availability of the labels (supervised learning $\Leftrightarrow$ labels are known). Then it is possible to characterize posterior distributions (discriminative). For instance the convolutions neural network.

3. **Hybrid deep networks:**
   When the estimation of the parameter of unsupervised deep networks depends on discriminative criteria, we can achieve Hybrid deep networks, e.g. Restricted Boltzmann Machines (generative model), which learns their parameters using Contrastive Divergence (discriminative criterion).

5.3 Activation Functions

In this section, we introduce three types of activation function, namely: 1. Sigmoid or Logistic 2. Tanh-Hyperbolic tangent 3. ReLu -Rectified linear units

Sigmoid Activation function ranges from 0 between 1 and has a S-shaped curve.

$$f(x) = \frac{1}{1 + exp(-x)} \tag{5.1}$$

The reasons of popularity of Sigmoid Activation function are :

i) Vanishing gradient problem ii) Its output is not zero centered. It makes the gradient updates go too far in different directions.$0 < output < 1$, and it makes optimization harder. iii) Sigmoids saturate and kill gradients. iv) Sigmoids have slow convergence.

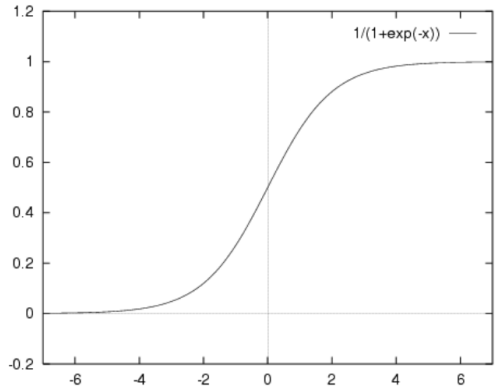


Figure 5.2: Sigmoid Activation function.

Hyperbolic Tangent Function- Tanh : Its output is zero centered because its ranges in between -1 to 1 with other words $-1 < itsoutput < 1$.

Hyperbolic Tangent function is more preferred in practice over Sigmoid function because its optimization is easier even though it suffers from vanishing gradient problem.

$$f(x) = \frac{1 - exp(-2x))}{1 + exp(-2x)} \tag{5.2}$$

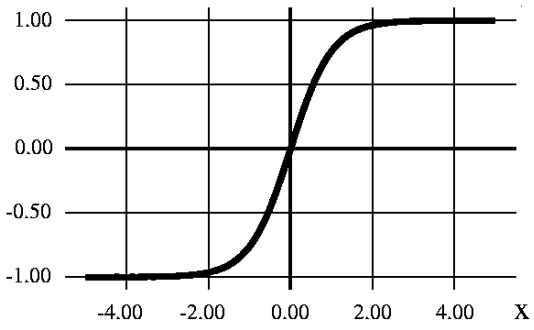
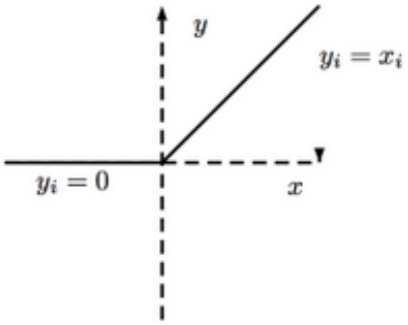
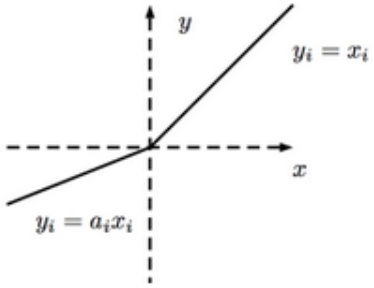


Figure 5.3: Hyperbolic Tangent function- Tanh

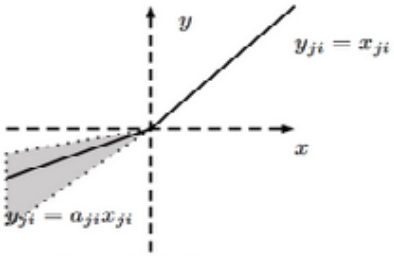
ReLu- Rectified Linear Units ReLu has become very popular in the past years. It was proved that it converges faster 6 times then *tahn* function.



(a) ReLU



(b) leaky ReLU



(c) Randomized Leaky ReLU

Figure 5.4: ReLU Types [9]

$$R(x) = \begin{cases} max(0, x) & if x > 0 \\ 0 & if x < 0 \end{cases} \tag{5.3}$$

Moreover, ReLu avoids and rectifiers vanishing gradient problem. Nowadays, almost

all deep learning models use ReLu. However, ReLu has one limitation namely, it can be used only within hidden layers .

5.4 Deep Belief Nets (DBN):

5.4.1 Definition:

DBN belongs to the first class of the deep learning classes which implies that DBN must be generative graphical models. Moreover, DBN is formed by multiple hidden layers, e.g, multiple layers of RBM or auto-encoder [37].

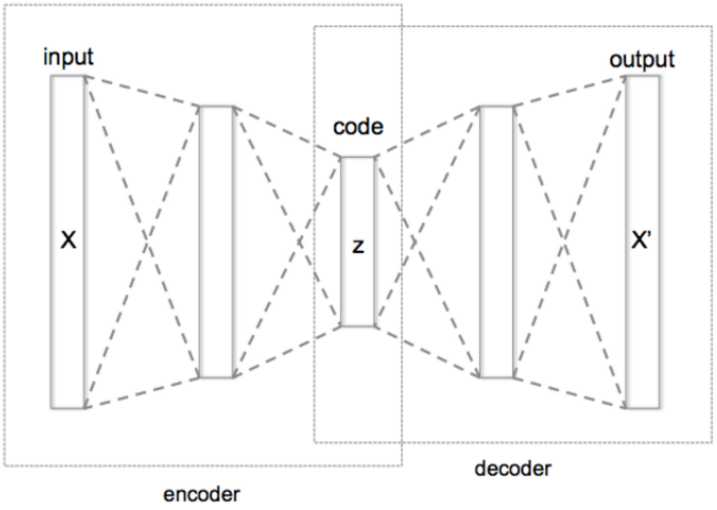


Figure 5.5: Auto-encoder [10]

5.4.2 Auto-Encoders

Auto-encoder is a neural network that implements two transformations: Encoder encode(x) : $R^n \rightarrow R^d$ and decoder(z) : $R^d \rightarrow R^n$ and fulfills the following properties: i) Feed-forward ii) non-recurrent neural network iii) The output units are equal to input nodes. iv) Hidden layers connect between the output and input layers. v) The dimensions (number of nodes) of hidden layers can smaller or larger than the input and output layers. vi) Their model treats an input data without labels (unsupervised learning). vii) Consists of an encoder and a decoder. viii) In case, there are multiple hidden layers, it is called deep auto-encoders. That implies that, we have two transition states: ix) Encoder phase: $\Phi : X \rightarrow F \Rightarrow z = sigmoid(Wx + b)$ x) Decoder

phase: $\phi : F \rightarrow X \Rightarrow x' = sigmoid(W'z + b')$ [10] [11].

Training of Auto-Encoders The aim of auto-encoder is to obtain d dimensional representation of data such that an error measure between x and $f(x) = decode(encode(x))$ is minimized [11]. Figure 5.6 shows a four layers auto-encoder network. In our model, both encoder and decoder parts of the auto-encoder consist of feed-forward neural networks with classical fully connected layers computing $l = f(W * x + b)$ where f is an activtion function. We now summary the process of the auto-encoder training in four steps: i) given a sparse x, compute dense $f(x)$ and loss using equation:

$$MMSE = \frac{m_i * (r_i - y_i)^2}{\sum_{i=0}^{i=n} m_i} \tag{5.4}$$

ii) Compute gradients and perform weight update (backward pass). iii) Treat $f(x)$ as a new example and compute $f(f(x))$. Now both $f(x)$ and $f(f(x))$ are dense and the loss from equation 5.4 has all m as non-zeros. (second forward pass). iv) Compute gradients and perform weight update (second backward pass) Steps (3) and (4) can be also performed more than once for every iteration.

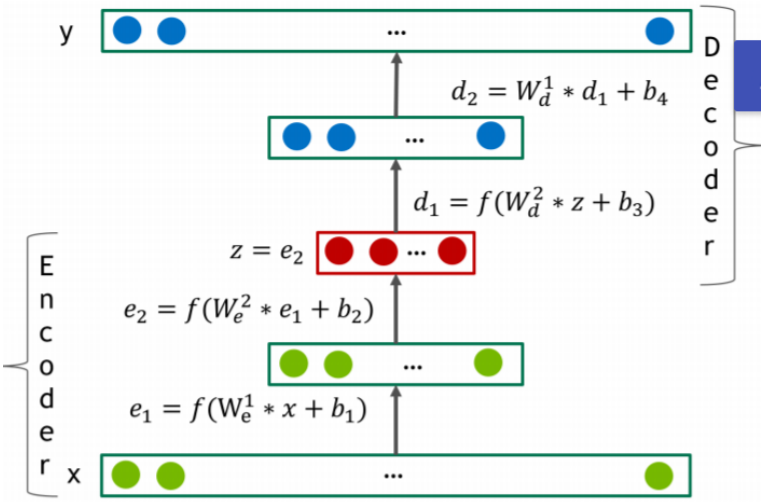


Figure 5.6: Training of Auto-encoder [11]

6 Classification Algorithms

6.1 Introduction:

Cluster analysis divides data into groups (clusters) for the purposes of summarization or improved understanding. Moreover, Cluster analysis divides data into meaningful or useful groups (clusters). If meaningful clusters are the goal, then the resulting clusters should capture the "natural" structure of the data.

6.2 Basic Concepts and Techniques of Cluster Analysis:

The goal of cluster analysis groups objects is to find some certain similarities between given data. However it is a hard task to achieve because the definition of what constitutes a cluster is not well defined, and in many applications, clusters are not well separated from one another. To illustrate this concept, how hard to decide what constitutes a cluster, we assume have data point in two dimensional space and we want to cluster them, [12].
In figure 6.5, we can see the most reasonable interpretation of the structure of these

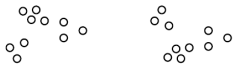


Figure 6.1: Initial points

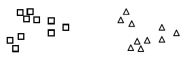


Figure 6.2: Two clusters

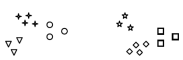


Figure 6.3: Six clusters

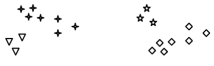


Figure 6.4: Four clusters

Figure 6.5: Data clustering [12]

points is that there are two clusters, On the other hand, we can divide the clusters into 4 clusters which may look more desirable for the human visual system. Finally, it may not be unreasonable to have four or six ones.

The Proximity Matrix The most cluster analysis methods use a similarity matrix S or a dissimilarity matrix D. Both of them are referred to as a **proximity matrix**. To understand a proximity matrix, there is an small illustration in figure 6.10 [12].

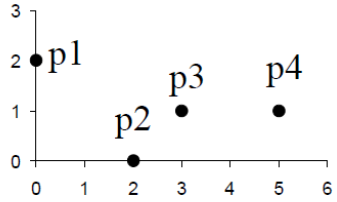


Figure 6.6: Two clusters

points	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Figure 6.7: data matrix

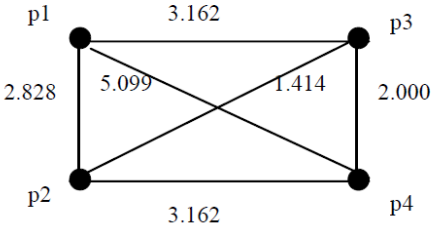


Figure 6.8: proximity graph

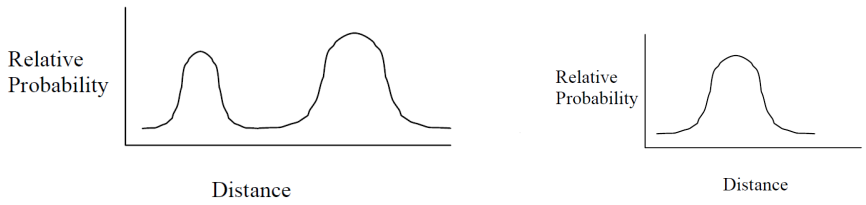
	p1	p2	p3	p4
p1	0.000	2.828	3.162	5.099
p2	2.828	0.000	1.414	3.162
p3	3.162	1.414	0.000	2.000
p4	5.099	3.162	2.000	0.000

Figure 6.9: proximity matrix

Figure 6.10: Data clustering [12]

6.3 The Curse of Dimensionality

It was Richard Bellman who came with this term **the curse of dimensionality** in his book [38]. The coming question is from his book [38], page 97: " In view of all that we have said in the forgoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days." In high dimensionality data points become increasingly "sparse".



(a) Data with clusters (b) Data without clusters

Figure 6.11: Checking the latent pattern [12]

To visualize this[1] , consider 100 points distributed with a uniform random distribution in the interval $[0, 1]$. If this interval is broken into 10 cells, then it is highly likely that all cells will contain some points. However, consider what happens if we keep the number of points the same, but distribute the points over the unit square (this corresponds to the situation where each point is two-dimensional). If we keep the unit of discretization to be 0.1 for each dimension, then we have 100 two-dimensional cells, and it is quite likely that some cells will be empty. For 100 points and three dimensions, most of the 1000 cells will be empty since there are far more points than cells. Conceptually our data is "lost in space" as we go to higher dimensions. Therefore, in order to know if a training data contains cluster is to plot probability density function of the pairwise distances of all points of the data set, see figure 6.11. As we can see in figure 6.11, if a data set contains clusters, we will have two peaks, in ideal case, or more close to each other. On the other hand, we will have only one peak if a data set does not manifest any pattern of clusters.

6.4 K-Means

6.4.1 Introduction

This section is based on [13]. K-means algorithm belongs to unsupervised learning branch. We begin by considering the problem of identifying clusters of data points in D-dimensional data.

We assume a data set $\{x_1, ..., x_2\}$ where $x_i \in N$ observations. Our aim to cluster the data set in k clusters. We assume that k is known. Moreover, we can compute inter-point distances and say, with some degree of certainty, this distance is smaller

[1] This example is taken from [7]

than those inter-points outside of i_{th} cluster. Each cluster has a center ,centroids, μ_i where $i \in K$ clusters. Therefore, we need a method to assign each point of the data set to its cluster.

6.4.2 Data Assignment

We start our work by defining binary indicator variables r_{nk} where $r_{nk} \in \{0, 1\}$ and k refers to a number of clusters. If data point x_n belongs to cluster k_{th} then $r_{nk} = 1$ if it does not then $r_{nk} = 0$. Secondly, we start defining an objective function or known as distortion function [13]:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| x_n - \mu_k \right\|^2 \tag{6.1}$$

By minimizing 6.1, we try to find out binary indicator variables $r_{nk}$and centriods of clusters μ_k.

We start by random initialization for μ_k. After that, kmeans is initialized and iterate between three steps:

- we fix μ_k and try to minimize J with respect to r_{nk}. More formally:

$$r_{nk} = \begin{cases} 1 & if k = \text{argmin}_j \left\| x_n - \mu_j \right\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

- we fix r_{nk} and try to minimize J with respect to μ_k. More formally:

$$2 \sum_{n=1}^{N} r_{nk} (x_n - \mu_j) \tag{6.3}$$

  then we solve for μ_k to give:

$$\mu_k = \frac{\sum_n r_{nk} \; x_n}{\sum_n r_{nk}} \tag{6.4}$$

  Then we update centriods values:

$$\mu_k^{new} = \mu_k^{old} + \eta_n (x_n - \mu_k^{old}) \tag{6.5}$$

  where η is a learning rate parameter.

- iterate till convergence.

These steps are explained in details in figure 6.12 .

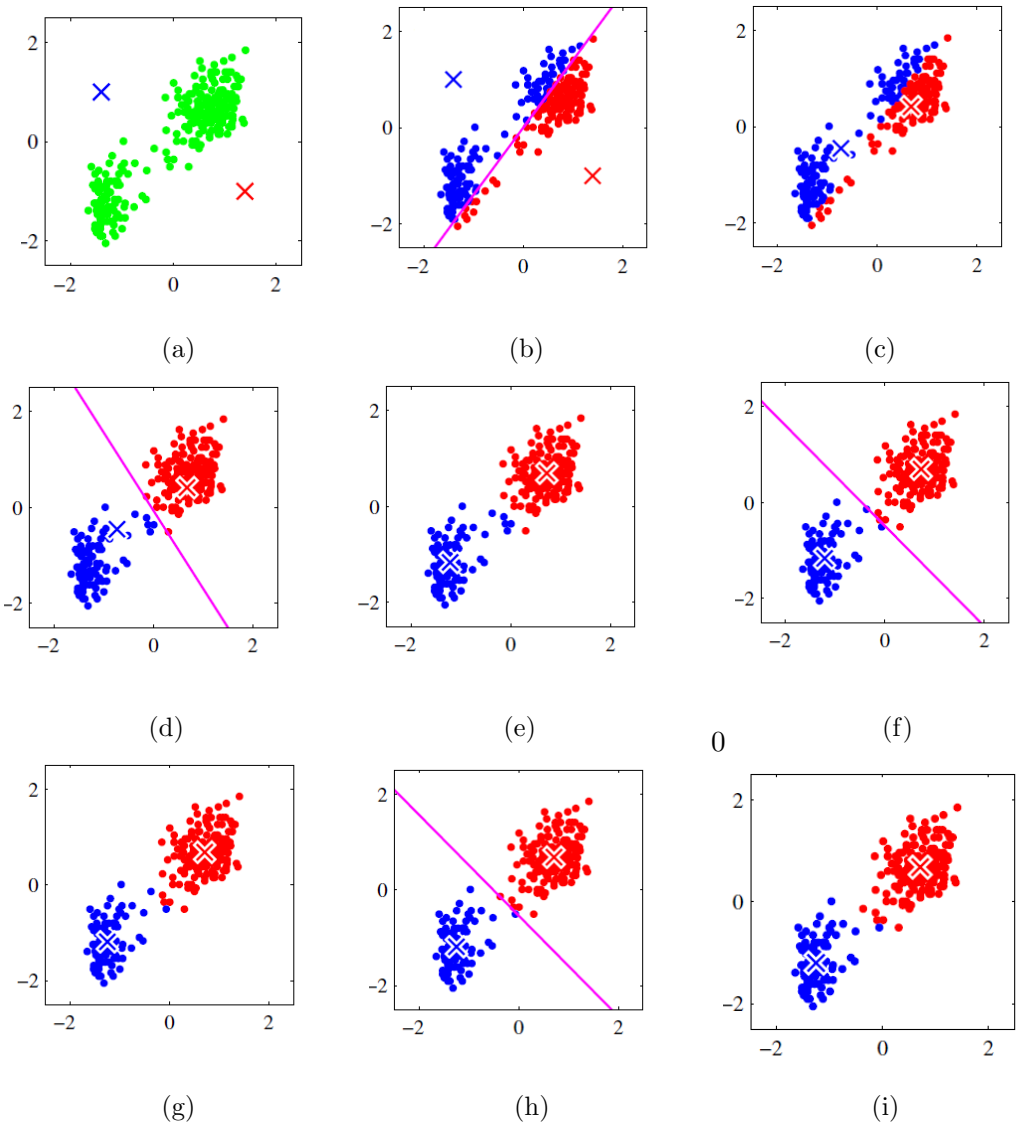


Figure 6.12: Illustration of the K-means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres μ_1 and μ_2 are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of the algorithm [13].

6.5 Deep Embedded Clustering (DEC)

6.5.1 Introduction

DEC is a method that simultaneously learns feature representations and cluster assignments using deep neural networks. This approach learns a mapping from data space to lower-dimensional feature space (unsupervised learning) in which it iteratively optimizes a clustering function-based objective. usually, we can train DNN using stochastic gradient descent or similar algorithms , this is called training phase which means finding out the required parameters (weights and biases of neural network). However, SGD is not suitable for unsupervised machine learning disciplines. In contrast to this novel (DEC) [2].
Deep beliefs network (DBN), consists of multiple layers, learns F in two phases:

1. soft assignment computation.
2. updating deep mapping.

6.5.2 Deep Embedded Clustering:

We need to cluster a set of N points into K clusters (K centroids). Furthermore, our data space manifests a higher dimensional representations. We introduce a nonlinear function F which transfers the data from higher dimensional data to lower dimensional presentation Z (known as embedded points) in order to avoid the curse of dimensionality. F transfers x (high dimensional representation) to z (lower dimensional representation). In the other words, we reduce dimension of data space by deploying F. F is defined as $f_\theta : X \to Z$, where θ are learnable parameters.

6.5.3 Soft Assignment

We will use t_distribution to measure similarity between z_i and μ_j

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'} (1 + \left\|z_i - \mu_{j'}\right\|^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{6.6}$$

where

1. $z_i = f_\theta \, \epsilon \, Z$ corresponds to $x_i \, \epsilon \, X$.
2. α the degrees of freedom of the student's t_distribution

3. q_{ij} is the probability of assigning sample i to cluster j

6.5.4 KL-Divergence Minimization

Our model is trained by defining the objective function:

$$L = KL(P \parallel Q) = \sum_i \sum_j p_{ij} \; log \; \frac{p_{ij}}{q_{ij}} \tag{6.7}$$

$$p_{ij} = \frac{q_{ij}^2 / f}{\sum_{j'} q_{ij'}^2 / f_{j'}} \tag{6.8}$$

where $f_i = \sum_i q_{ij}$ are soft cluster frequencies.

6.5.5 Optimization

we optimize the cluster centriods by using Stochastic Gradient Descent (SGD) as following:

$$\frac{\partial L}{\partial z_i} = \frac{\alpha + 1}{\alpha} \; \sum_j (1 + \frac{\|z_i + \mu_j\|^2}{\alpha})^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j) \tag{6.9}$$

$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha + 1}{\alpha} \; \sum_i (1 + \frac{\|z_i + \mu_j\|^2}{\alpha})^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j) \tag{6.10}$$

Then we backpropagate $\frac{\partial L}{\partial z_i}$ through DBN in order to compute DNN's parameters $\frac{\partial L}{\partial \theta}$.

6.5.6 Deep Beliefs Network Parameters

The first step of DEC is to build a stacked auto-encoder (SAE). The SAE consists of two layer neural defined as:

$$h = g_1(W_1 \tilde{x} + b_1) \tag{6.11}$$

$$y = g_2(W_2 h + b_2) \tag{6.12}$$

Our model parameters are $\theta = [W_1, b_1, W_2, b_2]$ that are computed by minimizing the least-sqaure loss $\|x - y\|^2$. We use ReLUs as an activation function in all encoder and decoder pairs. For more details about implementation , see chapter 7.

7 Contribution and Results

7.1 Channel Model Generation

The channel coefficients are generated following the Geometry-Based Stochastic Channel Model (GSCM), as elaborated in Chapter 3. We assume that the base station (BS) has a uniform linear array with $N = 100$ antennas, and single-antenna user terminals. We consider urban area environment and macro-cell with radius of 1200 meters centered at the BS. The azimuth angle ranges form $\frac{-\pi}{2}$ to $\frac{\pi}{2}$ with respect to broadside of the base station antenna array. The locations of the fixed scattering clusters are drawn uniformly in the range from 300 meters to 800 meters distance from the BS. Moreover, there are total of 7 fixed location scattering clusters. The user location is drawn uniformly in the range from 500 meters to 1200 meters from teh BS. For each channel measurement, we consider 4 scattering clusters, with 3 being fixed scattering clusters that are closest to the user, and the remaining cluster being at the user location in order to model user-location dependent scatterers. Each cluster contains 20 subpaths concentrated in a 4 degree angular spread. The AOA/AOD values are uniformly randomly generated within the angular spread.

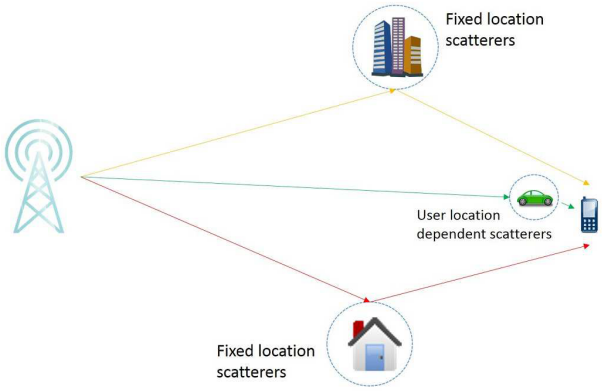


Figure 7.1: Illustration of signal propagation in a typical cell [1]

7.2 Dictionary Learning

We assume that the dictionary matrix D^d, after it is offline learned, remains constant in the channel estimation phase. We start by generating $L = 10000$ downlink channel responses ,$h_i^d, i = 1, ..., L$. We leverage the ℓ_1 framework using open toolbooks [1] for the dictionary learning process. As discussed in Chapter 2, the dictionary learning approach assumes that the downlink channel vectors admit a sparse representation with respect to the learned overcomplete (redundant) dictionary. Once we learn the dictionary, the downlink channel response h^d may be estimated from the decomposition $h^d = D^d\beta^d$, where β^d is a sparse vector, $\left\|\beta^d\right\|_0 \ll N$.

Let us consider a single cell Massive MIMO system operated in FDD mode. The BS is equipped with $N = 100$ antennas and we assume single-antenna users. Considering that the BS sends pilot (training) sequences for channel estimation, the received signal in the downlink is

$$y^d = \sqrt{\rho}Ah^d + n^d \tag{7.1}$$

where $A \in {}^{T^d \times N}$ is the matrix of pilot sequences, T^d is the length of the training sequences, and n is additive noise, $n \sim C\mathcal{N}(0, \sigma^2\mathbb{I})$. Aiming at reducing the channel estimation overhead, we are interested in the regime $T_d \leq N$. We will assume that the elements of the pilot matrix A are i.i.d Gaussian with mean 0 and variance 1. The noise vector n^d is also Gaussian, $n^d \sim C\mathcal{N}(0, \sigma^2\mathbb{I})$ where $\sigma = 1$, and for the channel vector we have $\left\|h^d\right\|_2 = 1$. ρ^d represents the combined effect of transmitted power and large scale fading. With the above, the SNR is defined as [1]:

$$SNR = \frac{\mathbb{E}\left\|\sqrt{\rho^d}Ah^d\right\|_2^2}{\mathbb{E}\left\|n^d\right\|_2^2} = \rho^d \tag{7.2}$$

Following [1], as a baseline we will use the case when the dictionary is fixed to be either a DFT basis, or an overcomplete DFT dictionary. A DFT basis F is defined as

$$\begin{aligned} F &= [f(-\frac{1}{2})\ f(-\frac{1}{2}+\frac{1}{N})\ ...\ f(\frac{1}{2}-\frac{1}{N})] \in \mathbb{C}^{N\times N}, \\ f(\psi) &= \frac{1}{\sqrt{N}}\left[1, e^{j2\pi\psi}, ..., e^{j2\pi\psi(N-1)}\right]^T \end{aligned} \tag{7.3}$$

Based on a M-dimensional DFT basis F, where $M > N$, an overcomplete $N \times M$ DFT dictionary $\tilde{F}$ may be obtained by deleting $M - N$ rows of F.

[1] 4K-SVD toolbox and SPGL1 toolbox `https://elad.cs.technion.ac.il/software/`

As a performance measure we take the normalized mean square error (NMSE), defined as:

$$NMSE = \frac{\mathbb{E}\left\|h^d - \hat{h}^d\right\|_2^2}{\mathbb{E}\left\|n^d\right\|_2^2} = \frac{1}{L}\sum_{i=1}^{L}\left\|h^d - \hat{h}^d\right\|_2^2, \tag{7.4}$$

where $\hat{h}^d$ is the downlink channel estimate estimate and L is the number of measurements. As we assume the dictionary learning phase to be performed offline, this step is not critical for the channel estimation performance and different algorithms may be used for the dictionary learning task. Following [1] the dictionary learning step may be summarized as

Dictionary Learning [1]

`Input`: model mismatch constraint η; `Output`: learned dictionary D^d

- Collect downlink channel measurements $h_i^d, i = 1, ..., L$ in differnt locations of the cell.
- Solve the dictionary learning problem.

$$\underset{\substack{D^d \in C \\ \beta_1^d,\ldots,\beta_l^d}}{min} = \frac{1}{L}\sum_{i=1}^{L}\left\|\beta_i^d\right\|_1 s.t\left\|h_i^d - D^d\beta_i^d\right\| \leq \eta, \forall i. \tag{7.5}$$

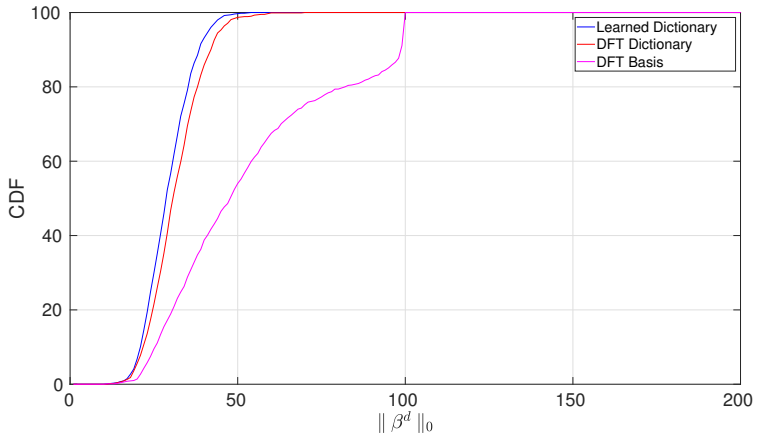
  where $\eta \in \{0.01, 0, 1\}$ and C is the contraint set defined as following:

$$C = \{D \in {}^{N\times M}, \text{subject to}\left\|D_{.j}\right\|_2 \leq 1, \forall j = 1, ..., M\} \tag{7.6}$$

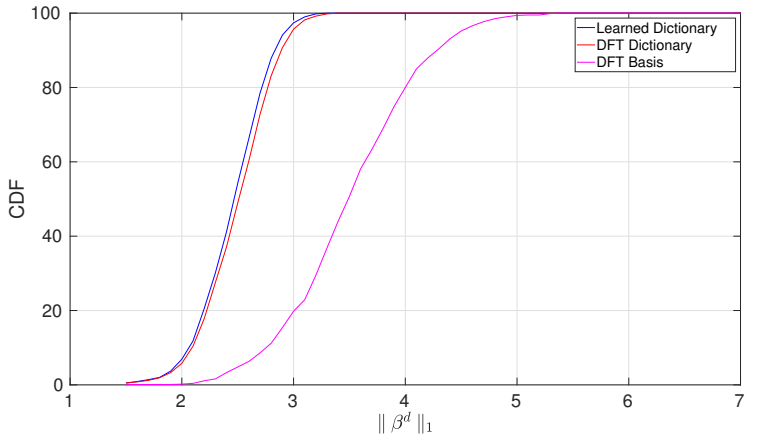
The L channel measurement vectors used to learn the dictionary in the offline phase are generated according to the GSCM channel model described in Chapter 3. To confirm the validity of the dictionary learning approach, i.e. the sparse channel representation $h^d = D\beta^d$, where $\left\|\beta\right\|_0 \ll N$, we compute the cumulative distribution function (CDF) of $\left\|\beta\right\|_0$ and $\left\|\beta\right\|_1$.

Fig. 7.2 and Fig. 7.3 depict the CDF of the l_0 and l_1 norm of β, under a model mismatch factor $\eta = 0.1$, respectively $\eta = 0.01$. We observe that results support the sparse channel assumption. In addition, the learned dictionary has the smallest l_0 and l_1 norm, followed by the overcomplete DFT dictionary. The DFT basis is the least appropriate from the perspective of sparse representation.

Fig. 7.2 indicates that approximately 40 columns of D^d are sufficient to represent the channel for $\eta = 0.1$.



(a) CDF of $\|\beta\|_0$



(b) CDF of $\|\beta\|_1$

Figure 7.2: Cumulative distribution function of $\|\beta\|_0$and $\|\beta\|_1$ for a base station with 100 antennas. Model mismatch factor $\eta = 0.1$

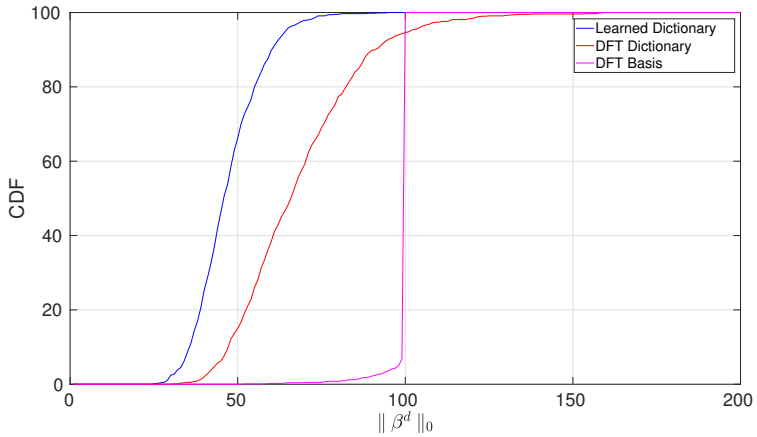
From Fig. 7.3 we see that approximately 60 columns of the learned dictionary are required to represent the channel.

7.3 Compressed Channel Estimation

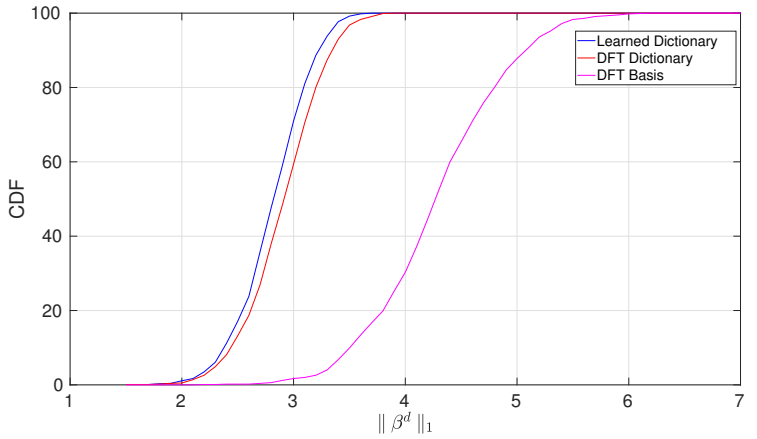
Given the dictionary D^d, in the channel estimation step the task is to estimate the vector β^d (i.e. the channel h^d) according to the model where y^d is given as

$$\begin{aligned} y^d &= \sqrt{\rho} A h^d + n^d \\ &= \sqrt{\rho} A D^d \beta^d + n^d \end{aligned}$$

We start by summarizing the approach in [1].



(a) CDF of $\|\beta\|_0$



(b) CDF of $\|\beta\|_1$

Figure 7.3: Cumulative distribution function of $\|\beta\|_0$and $\|\beta\|_1$ for a base station with 100 antennas. Model mismatch factor $\eta = 0.01$

Compressed channel estimation according to [1].

`Input`: downlink pilots A, downlink dictionary D^d, downlink recieved power ρ^d, error constraint ϵ `Output`:downlink channel estimation $\hat{h}^d$

- Base station transmits downlink pilots A.
- User recieves $y^d = \sqrt{\rho^d} A h^d + n^d$.
- User feed back y^d.

- Base station performs sparse recovery using LASSO

$$\min_{\beta^d} \left\|\beta^d\right\|_1 s.t. \left\|y^d - \sqrt{\rho^d} A D^d B^d\right\|_2 \leq \epsilon \tag{7.7}$$

- Base station estimates downlink channel as $\hat{h^d} = D^d \beta^d$

We note that the channel estimation approach in [1] uses LASSO to solve the optimization problem in (7.7).

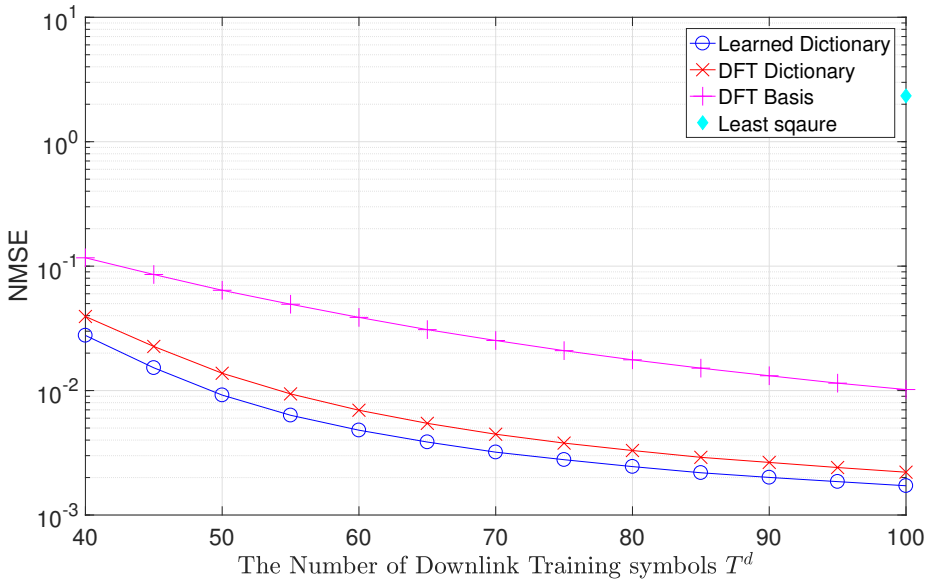


Figure 7.4: Normalized mean square error (NMSE) comparison of different sparsifying basis and dictionaries where $SNR = 30dB$.

Fig. 7.4 depicts the NMSE of the channel estimation with learned dictionary, as compared to a DFT basis and an overcomplete DFT dictionary. As shown, the dictionary learning approach from [1] delivers the best result in terms of NMSE, compared to both the DFT basis and the overcomplete DFT dictionary. We also compare to the least squares (LS) method which requires $T^d \geq N = 100$. We observe that when compressed channel estimation is performed by using the learned dictionary, less training symbols (i.e. shorter pilot sequences) are required compared to DFT dictionary and DFT basis.

7.3.1 Signal Space CoSaMP for Sparse Recovery with Redundant Dictionaries

Signal Space CoSaMP (SSCoSaMP) [39] is an extension of the CoSaMP algorithm introduced in [40]. CoSaMP is an iterative recovery algorithm that delivers the same guarantees as the best optimization-based approaches. CoSaMP is in its essence similar to greedy pursuit, and is thus of lower complexity than convex optimization approaches.

However, in its original form, CoSaMP assumes sparse representation with respect to an orthogonal basis and is thus not directly applicable for the channel estimation problem where the sparse representation is with respect to an overcomplete dictionary. Signal Space CoSaMP (SSCoSaMP) is an extension of CoSaMP to perform recovery of signals that have sparse representation in a redundant dictionary. In the following we first summarize the SSCoSaMP algorithm in it original form. Afterwards, we present an adaptation that is better suited for our channel estimation problem at hand.

Let A be a $m \times n$ matrix and let D be an $n \times d$ arbitrary matrix. Also, let y be the observed signal in the compressive sensing system model, where x has a representation in a dictionary D, $x = D\alpha$

$$\begin{aligned} y &= Ax + w \\ &= AD\alpha + w. \end{aligned}$$

We say that A satisfies the $D - RIP$ property of order k if there exists a constant $\delta \in (0, 1)$ such that:

$$\sqrt{1-\delta_k} \leq \frac{\|AD\alpha\|_2}{\|D\alpha\|_2} \leq \sqrt{1+\delta_k} \tag{7.8}$$

The equation holds for all α satisfying $\|\alpha\|_0 \leq k$. A lemma in [39] states that, for any choice of D, if A is populated with i.i.d. random entries from a Gaussian or subgaussian distribution, then with high probability, A will satisfy the D-RIP of order k as long as $m = O(k\ log(\frac{d}{k}))$.

Th original SSCosaMP algorithm is summarized in the following.

Algorithm 1 Signal Space CoSaMP

Input: A,D,y,k, stopping criterion

initialize: $r = y$, $x^0 = 0$,$\ell = 0$, $\Gamma = \varnothing$ stopping criterion

while: not converage **do**

proxy: $\tilde{v} = A * r$

identify: $\Omega = S_D(\tilde{v}, 2k)$

merge: $T = \Omega \cup \Gamma$

update: $\tilde{x} = \underset{z}{\arg\min} \|y - Az\|_2 \quad \text{s.t.} \quad z \in R(D_T)$

$\Gamma = S_D(\tilde{v}, 2k)$

$x^{\ell+1} = \mathcal{P}_\Gamma \tilde{x}$

$r = y - Ax_{\ell+1}$

$\ell = \ell + 1$

end while

output: $\hat{x} = x^\ell$

Our numerical simulations suggest that in the case when $T^d < N$, and D^d is the learned dictionary, SSCoSaMP does not always converge. For this reason, we introduce a modification in the update step of SSCoSaMP. The modified algorithm can be summarized as follows:

Algorithm 2 Modified SSCoSaMP

Input: A, D, y, k, stopping criterion

initialize: $r = y$, $x^0 = 0$,$\ell = 0$, $\Gamma = \varnothing$ stopping criterion

while: not converge **do**

proxy: $\tilde{v} = A * r$

identify: $\Omega = S_D(\tilde{v}, 2k)$

merge: $T = \Omega \cup \Gamma$

update: $\tilde{x} = \underset{z \in R(D_T)}{\arg\min} \|y - Az\|_2 \quad \text{s.t.} \quad \left\| y^d - \sqrt{\rho^d} A D^d B^d \right\|_2 \leq \epsilon$

$\Gamma = S_D(\tilde{v}, 2k)$

$x^{\ell+1} = \mathcal{P}_\Gamma \tilde{x}$

$r = y - Ax_{\ell+1}$

$\ell = \ell + 1$

end while

output: $\hat{x} = x^\ell$

Fig. 7.5 illustrates the NMSE performance of the modified SSCOSAMP and the algorithm in [1], for sparsity level $\left\|\beta^d\right\|_0 = 40$ and $SNR = 30$. We observe that

the modified SSCOSAMP requires a relatively high pilot length ($T^d \geq 80$) for a very reliable channel estimation. Both algorithms significantly outperform least squares (LS).

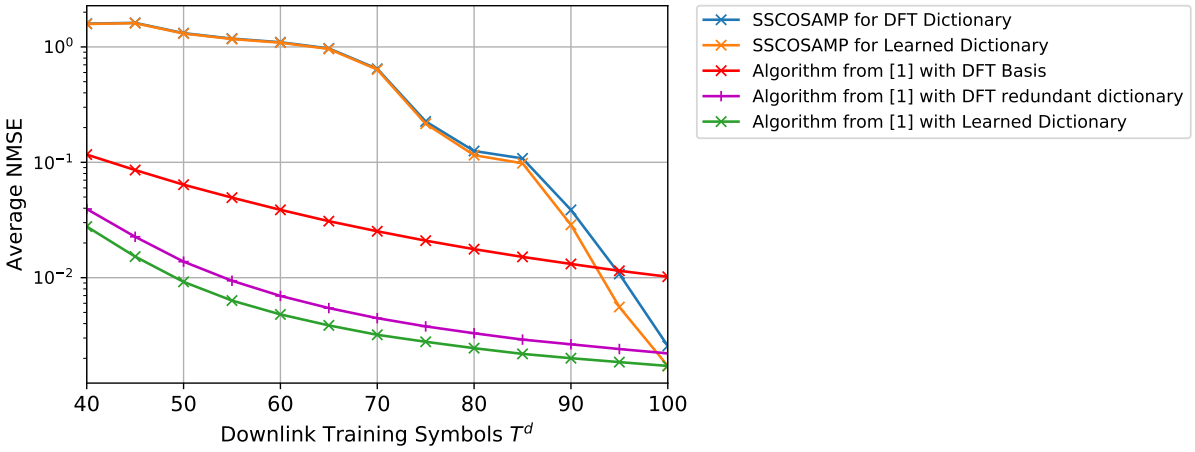


Figure 7.5: Normalized mean square error (NMSE) with $SNR = 30dB$.

7.4 Deep Learning for Sparse Channel Estimation

7.4.1 Introduction

Deep learning has been recently applied to applications in compressed sensing, where neural networks have been used to learn the unfolded versions of iterative algorithms such as approximate message passing (AMP) and the iterative shrinkage-thresholding algorithm (ISTA) [26] [27] from data. From a general point of view, iterative algorithms can be expressed as a repetition of operations in the form:

$$\mathbf{x}_{t+1} \coloneqq g\left(\mathbf{x}_t; \mathbf{y}, \boldsymbol{\Theta}\right) \tag{7.9}$$

for a certain number of iterations indexed by t. $\boldsymbol{\Theta}$ represents a set of optimization parameters, and g encapsulates the iteration step, which may include intermediate vectors, linear operations and (parametric) non-linear functions. Each iteration step can subsequently be transformed into a hidden layer with inputs $\mathbf{x}_t$ and $\mathbf{y}$, output $\mathbf{x}_{t+1}$, learnable parameters $\boldsymbol{\Theta}_t$ taking over the role of classical weights and non-linear operations acting as activation functions. Iterations are then unfolded T times to form a T-layer neural network. This approach is used in [26] [14] to design learned AMP (LAMP) and learned Vector-AMP (LVAMP) neural networks.

7.4.2 Learned Iterative Shrinkage and Thresholding Algorithm (LISTA)

LISTA is an approach that uses the deep learning framework based on the ISTA algorithm to find a solution to inverse linear problems. Under this framework, the ISTA iteration can be represented as [14]:

$$\hat{x}_{t+1} = \eta_{st}(S\hat{x}_t + By; \lambda) \text{ with } \begin{cases} B = \beta A^T \\ S = I_N - BA \end{cases} \tag{7.10}$$

Fig. 7.6 depicts a feed-forward neural network constructed by unfolding $T = 4$ iterations of ISTA. By training the ISTA network, we try to find out $\Theta = [B, S, \lambda]$ by

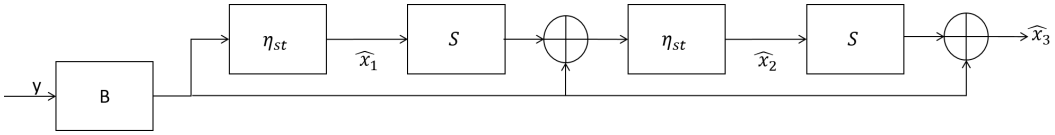


Figure 7.6: The feed-forward neural network constructed by unfolding T = 4 iterations of ISTA [14]

minimizing the following quadratic loss:

$$\mathbb{L}(\theta) = \frac{1}{D}\sum_{d=1}^{D} \left\| \hat{x}_T(y^d; \Theta) - x^d \right\|_2^2 \tag{7.11}$$

We can also illustrate Fig. 7.6 in a different way, by zooming in at the t_th layer/iteration, as depicted in Fig. 7.7

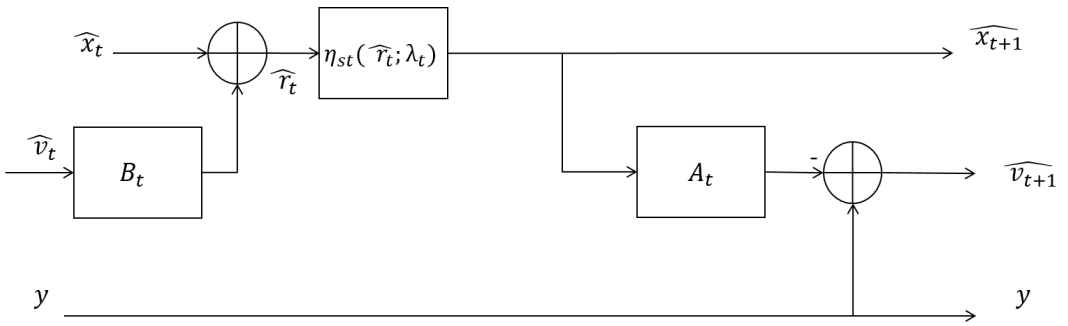


Figure 7.7: The t_{th} layer of the LISTA network, with its learnable parameters A_t, B_t and λ_t [14]

.

7.4.3 Learned Approximate Message Passing (LAMP)

In this section, we present the neural network structure associated with LAMP [14]. Fig. 7.8 shows an example for a LAMP neural network. There are two major dif-

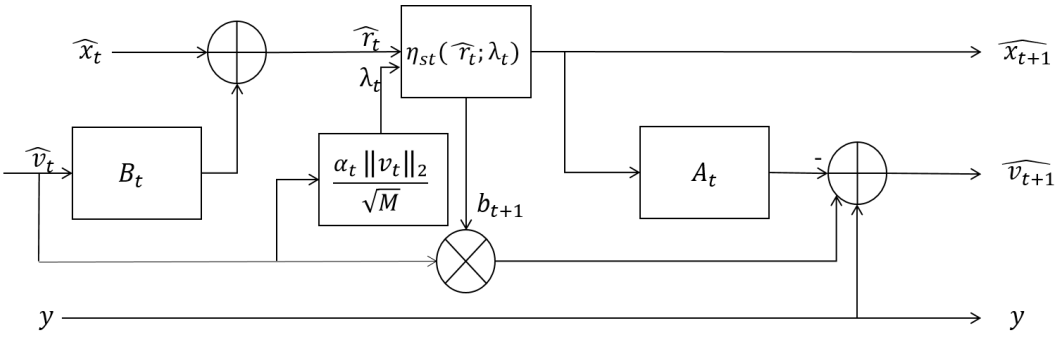


Figure 7.8: The t_{th} layer of the LAMP-ℓ_1 network, with its learnable parameters [14]

ferences between learned ISTA and learned AMP: i) AMP's residual v_t includes the Onsager-correction term "$b_t v_{t-1}$". ii) AMP's shrinkage threshold λ_t, includes the t-dependent value, which given by $\lambda_t = \frac{\alpha}{\sqrt{M}} \|v_t\|_2$.

Parameterizing of LAMP-ℓ_1: In t-th layer, the LAMP-ℓ parameters can be summarized in the following:

$$\hat{x}_{t+1} = \beta_t \eta_{st}(\hat{x}_t + B_t v_t; \frac{\alpha_t}{\sqrt{M}} \|v_t\|_2) \tag{7.12}$$

$$v_{t+1} = y - A\hat{x}_{t+1} + \frac{\beta_t}{M} \|\hat{x}_{t+1}\|_0 \, v_t, \tag{7.13}$$

where $\hat{x}_0 = 0$ and $v_0 = y$. As discussed, the main goal is to learn $\Theta_{T-1}^{tied} \triangleq \{B, \{\alpha_t, \beta_t\}_{t=0}^{T-1}\}$ by minimizing the quadratic loss function 7.11. We summarize LAMP-ℓ and its learnable parameters in the following

---

Algorithm 1 Tied LAMP-ℓ_1 parameter learning.

Initialize: $B = A^T$, $\alpha_0 = 1$,$\beta_0 = 1$

Learn: $\Theta_0^t ied = \{B, \alpha_0\}$

for $t = 1$ **to** $T - 1$ **do**

Initialize $\alpha_t = \alpha_{t-1}$, $\beta_t = \beta_{t-1}$

Learn $\{\alpha_t, \beta_t\}$ with fixed $\Theta_{t-1}^{\text{tied}}$

Re-learn $\Theta_t^{\text{tied}} = \{B, \{\alpha_i, \beta_i\}, \alpha_0\}$

end for Return $\Theta_{T-1}^{\text{tied}}$

Channel estimation with LISTA and LAMP: In the following, we present results for the channel estimation problem in the dictionary learning setup. We implement LAMP and LISTA in Python using Tensorflow. One of the important steps in the implementation is to determine the optimal number of layers for the neural network architectures based on LAMP and LIST. To that aim, we generate $L = 10^6$ pairs of the $\{y_i^d, x_i^d\}^L$ for different values of T^d. In the training phase, we need to build 13 different neural networks. Their configurations match the value of T_i^d where $i \in= [40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]$, which corresponds to the dimension of the input layer. We can illustrate the result of the training by plotting the average NMSE against layers, see Fig. 7.9. Our experimental results show that only 10 layers(iterations) are enough to get an accurate estimate. After the network is trained, we fix the number of layers to 10 and check validity, performance and accuracy of the deployment by generating new downlink signals $y_{x_i}^d$ where $i \in \{1, ..., 100000\}$. We feed them into our NN (LAMP and LISTA) to estimate the sparse channel vectors.

We note that LAMP and LISTA have already been applied to communication scenarios involving solving linear inverse problems. However, the important difference here is that the channel representation (and hence the sparse channel recovery) is with respect to a learned, redundant dictionary, in contrast to an orthogonal basis/DFT dictionary. As result, as we have seen already in the case with SSCoSaMP, this issue has important implications on the convergence of the sparse recovery algorithm. However, the expectation here is that LISTA and, in particular, LAMP, are more robust to effects of this type. The expectation is confirmed bay the numerical simulations illustrated in Fig. 7.9, which depicts the average NMSE as function of the length of the training (pilot) sequence in the downlink. First, we observe that the experimental results show that both LAMP and LISTA provide good estimation performance of β^d. Second, we observe a graceful degradation of the performance with the decrease of the length of the pilot sequence, meaning that both algorithms adapt well to the whole range of T^d values.

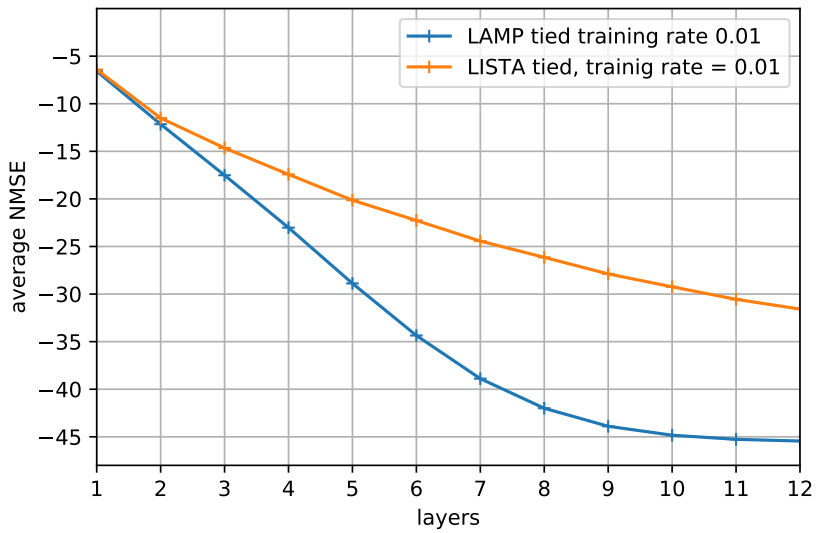


Figure 7.9: Normalized mean square error (NMSE) vs layers

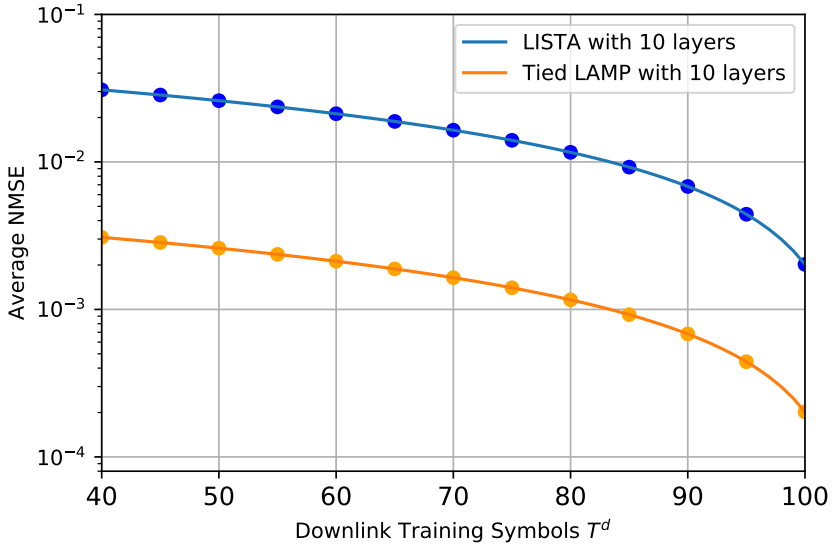


Figure 7.10: Normalized mean square error (NMSE) vs T^d

7.5 Classification Algorithms for Compressed Channel Representation

In the previous sections, the focus was on channel estimation algorithms which solve the underlying linear inverse problem. In this section we follow a different approach

and employ classification algorithms that result in a compressed channel representation. The motivation for the approach is that in practice it is usually the case that both the observations (measurements) and the channel estimates have to be quantized or compressed at some point (due to the limited amount of feedback and depending on the channel estimation protocol). For this reason, it is reasonable to investigate approaches where the channel vectors are directly compressed in the domain of their low-dimensional representation. This, for example, can be achieved by employing clustering algorithms.

Before we do this, we run some experiments to assess the potential and confirm the validity of clustering approach. We start by measuring similarity on the generated data set by plotting a histogram (approximate probability density function) of the pairwise distances of all points in the data set. For this purpose, we generate 10^6 samples ($\{h^d\}^l$ respectively $\{y^d\}^l$) according to our channel model. Fig. 7.11 depicts the approximate probability density function of the pairwise distances of the received signal vector y^d.

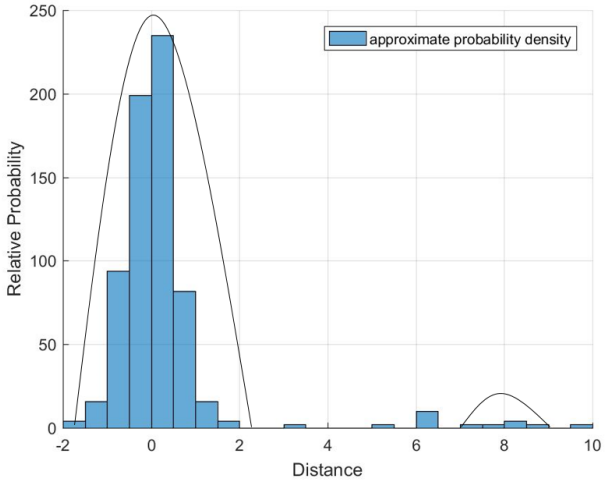


Figure 7.11: Inter-Point distances for data y^d

As illustrated, the data set manifests a latent pattern of classification. Therefore, we proceed by deploying two different classification to approximate the downlink channel response h^d, K-means (as described in Sec. 6.4(, and Unsupervised Deep Embedding for Clustering Analysis (as described in Sec. 6.5).

7.5.1 K-means

K-means consists of two phases:

- Learning phase.
- Estimation and testing phase.

Learning Phase: Since the K-means algorithm requires the number of clusters K, our first task is to find an appropriate number of clusters for channel representation with sufficient precision. For this, we first generate $L = 10^6$ observation vectors y^d according to our channel model. Determining the optimal number of clusters K is not an easy task to accomplish. In order to solve this challenge, we plot the distribution of our training data with respect to each cluster C_i where $i \in \{1, ..., K\}$, see figure 7.12 and then we compute the average mean square error for each iteration of K. It turns out that $K = 30$ returns a good compromise between the number of clusters and reliable representation.

As result of the clustering, each observation vector y^d can be approximated by the according cluster centroid, which provides an efficient way of channel compression.

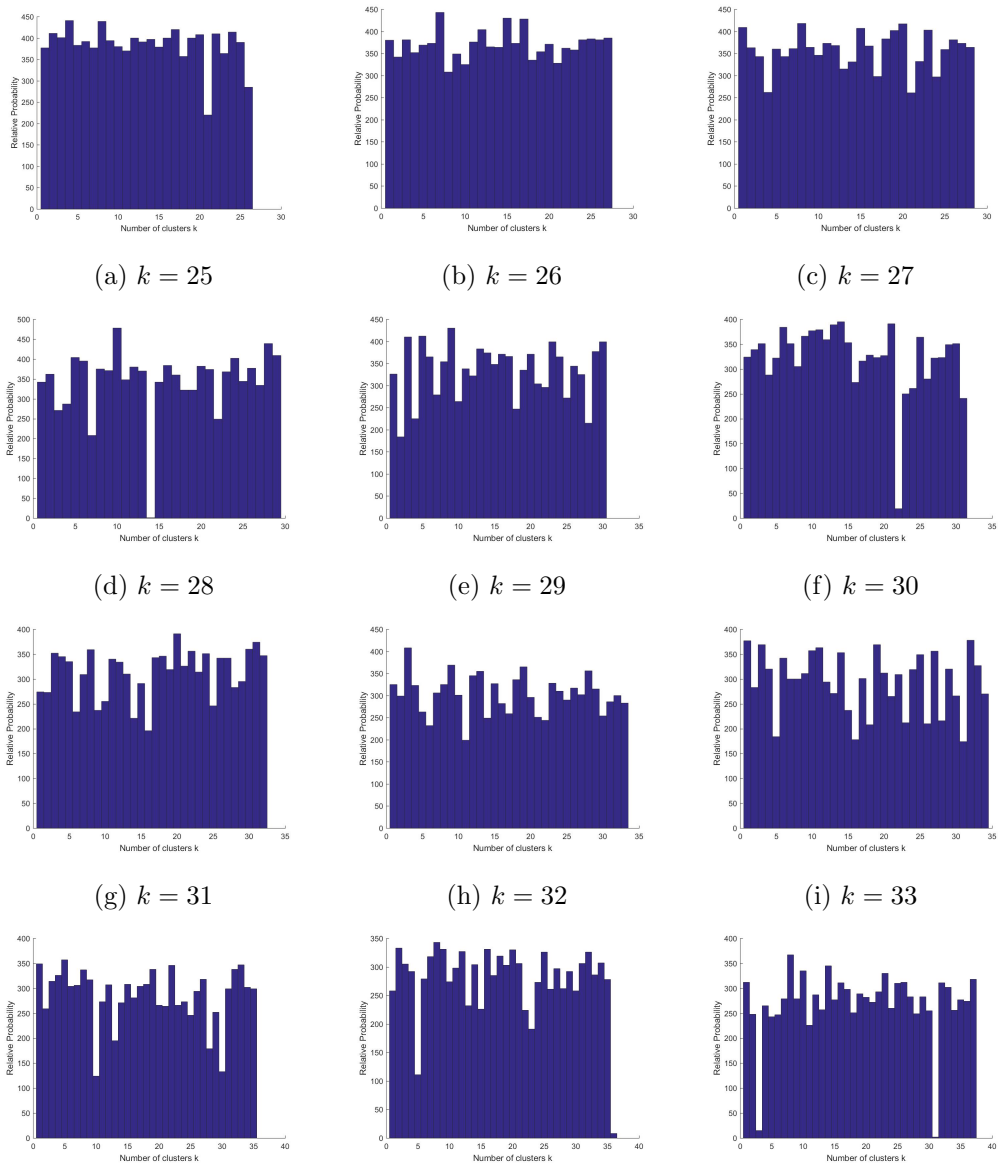
The last step of the training is to calculate a lookup-table that will be stored at both the mobile staion (MS) and at the base station (BS). For this purpose, we compute the estimated received signal $\hat{y}_K^d$ where $k \in \{1, ..., 30\}$ and deploy LASSO to find an approximation of the sparse vector by solving equation 7.14:

$$\min_{\beta^d} \left\| \beta^d \right\|_1 s.t. \left\| c_k - D^d B^d \right\|_2 \leq \epsilon \tag{7.14}$$

Estimation and testing phase: We now check the validity of our approach by generating 10^6 samples of y_i^d and their pilot length T^d varies from 40 to 100. Then we estimate the sparse vector β_i^d by computing the upcoming equation:

$$\Delta = \left\| y_i^d - \hat{y}_k \right\|_2^2 \tag{7.15}$$

Where y_i^d is the new generated values "$i \in \{1, ..., 100000\}$" and $\hat{y}_k^d$ is the estimated value by means of K-means. That implies, the mobile station computes Δ only thirty times and then MS finds the smallest value of Δ. By discovering the smallest Δ implies discovering the matching index (Cluster). Nevertheless, we already computed



(a) $k = 25$ (b) $k = 26$ (c) $k = 27$

(d) $k = 28$ (e) $k = 29$ (f) $k = 30$

(g) $k = 31$ (h) $k = 32$ (i) $k = 33$

(j) $k = 34$ (k) $k = 35$ (l) $k = 36$

Figure 7.12: Determine the best K

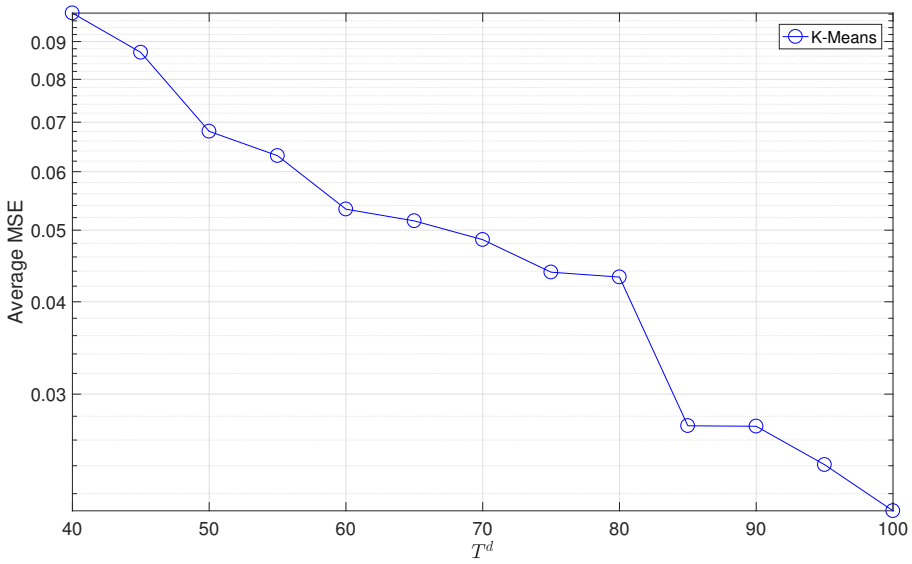


Figure 7.13: Normalized mean square error (NMSE) vs T^d

the sparse vectors for each cluster, that easily leads us to compute the sparse vector. We plot the results in Fig. 7.13.

In its original form K-means works only for real numbers, but we deploy K-means, in this context, to classify complex vectors. To solve this challenge, we resort to the representation of complex numbers via real-valued 2×2 matrices. Formally, a complex number $= a + ib$ can be represented as $\begin{bmatrix} a & b \\ -b & a \end{bmatrix}$. Fig. 7.13 depicts that K-means can approximate h^d with average NMSE which ranges between 0.1 and 0.01. That implies that each MS needs only 5 bits to inform the base station about the respective clustering index of the downlink received signal.

7.5.2 Unsupervised Deep Embedding for Clustering Analysis

In this section we investigate the performance of a clustering algorithm based on deep unsupervised embedding (DEC). We recall that details of the DEC-algorithm are provided in Section 6.5. In the deep learning architecture we set the network dimensions to 200–500–500–2000–45. All layers are densely (fully) connected. We initialize the weights to random numbers drawn from a zero mean-Gaussian distribution with standard deviation of 0.01. Each layer is pre-trained for 50000 iterations. The entire deep autoencoder is further fine-tuned for 100000 iterations. The Learning rate is set to

0.1. To initialize the centroids, we run K-means with $K = 30$. In the KL divergence minimization phase, we train with a constant learning rate of 0.01. The convergence threshold is set to $tol = 0.1\%$. The results for the NMSE are illustrated in Fig. 7.14.

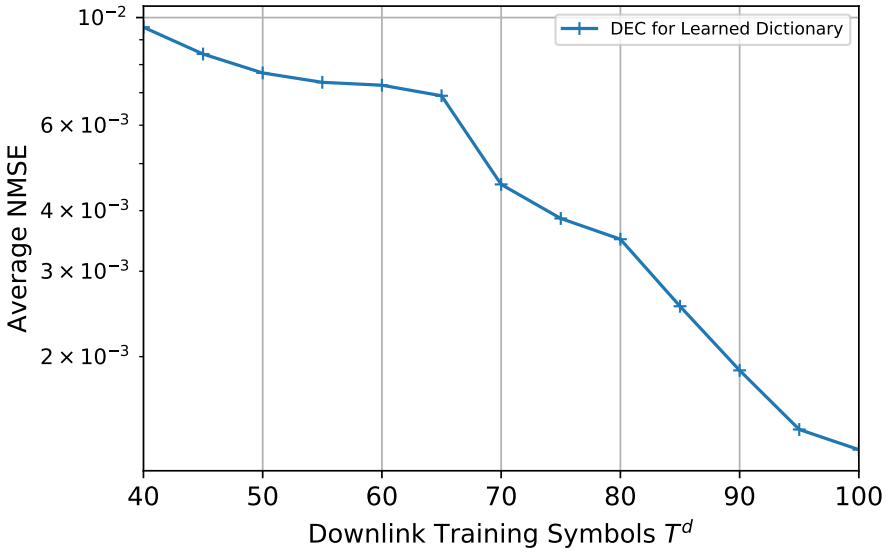


Figure 7.14: Normalized mean square error (NMSE) vs T^d

7.5.3 Overall Performance Comparison

In the following we compare the performance of all algorithms considered in this thesis. LISTA and LAMP were implemented in Python using Tensorflow with ADAM optimizer. The DLCM algorithm from [1] and K-means are implemented in Matlab. The algorithm based on deep embedded clustering (DEC) is implemented in Python using Keras.

Fig. 7.15 is illustrative as it shows the potential of the learning framework to solve channel estimation problems under the dictionary learning model. In particular, we observe the outstanding performance of LAMP which outperforms all other algorithms over the whole considered range of lengths of the pilot sequences. The performance of LAMP is followed by the performance of the algorithm in [1] which, however, is of greater complexity compared to LAMP. It is interesting to see that the clustering approach based on deep embedding clustering fares well compared to other algorithms, while simultaneously providing compressed channel representation which is very useful if we consider quantized feedback. Additionally, we observe that the NMSE performance of both LAMP and DEC degrade gracefully when we decrese the length of the

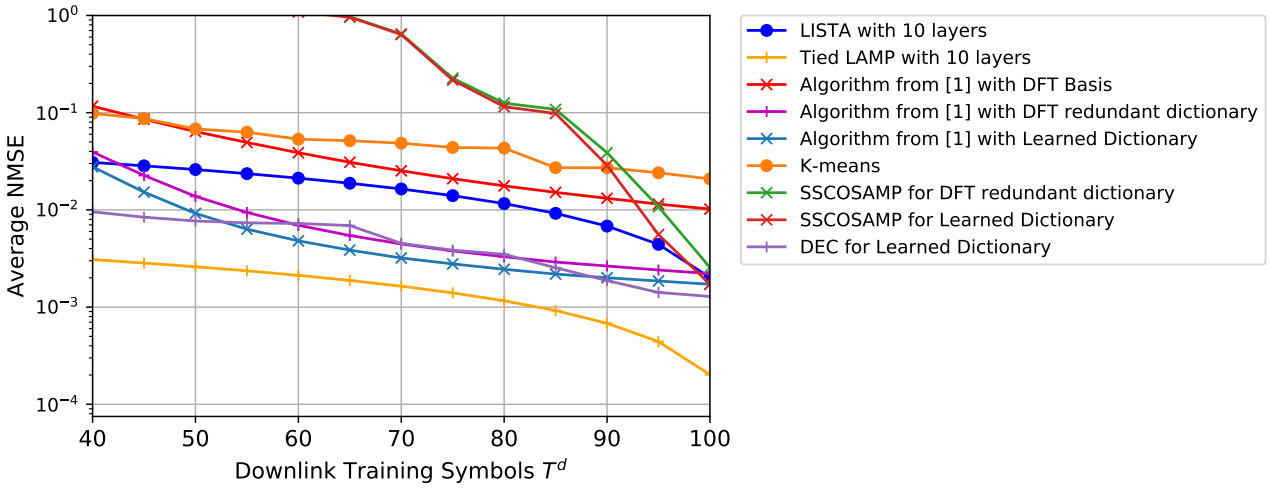


Figure 7.15: Compassion between the deployed algorithms

pilot sequence T^d for training LISTA, SSCoSaMP and DEC tend to have the same behaviour when $T^d \geq 100$. Altogether, the results indicate that the considered approaches can represent a viable alternative to more classical compressed sensing based channel estimation algorithms.

8 Conclusion

Motivated by the approach in [1], in this thesis we addressed a dictionary learning based approach to channel representation and estimation in massive MIMO systems. According to this approach to channel modelling, a learned redundant dictionary is used to represent the channel vectors. The motivation behind the approach is that the learned dictionary, due to the learning process, is able to adapt to the cell characteristics as well as insure a sparse representation of the channel. To perform channel estimation in the dictionary learning channel model, it was necessary to extend the compressed sensing (CS) framework to be able to handle signals that have sparse representations in a learned redundant dictionary, rather than a predefined orthogonal basis.

Focusing on this problem, we investigated the performance of two classes of algorithms: i) adaptations of "classical" CS reconstruction algorithms (LASSO and Signal-Space COSSAMP), to account specifically for the learned overcomplete dictionary in the channel estimation step; ii) deep learning algorithms to mimic, using a neural network, iterative algorithms such as approximate message passing (AMP) and iterative shrinkage-thresholding algorithms (ISTA). The numerical results indicate that, under the dictionary learning based channel model, the deep learning framework yields performance (in terms of normalized mean square error) which is comparable, or even superior (for a certain range of parameters), to compressive sensing based algorithms for channel estimation in the dictionary learning based model.

Besides this, we also investigated the performance of unsupervised clustering algorithms for compressed channel representation. For this purpose we evaluated both the performance of the classical K-means algorithm, and a non-parametric method based on deep embedded clustering in the spirit of [2]. The numerical results suggest that the compressed channel representation based on unsupervised clustering may be seen as a viable alternative to methods based on compressed channel feedback.

Bibliography

[1] Yacong Ding and Bhaskar D. Rao. Dictionary learning based sparse channel representation and estimation for FDD massive MIMO systems. *CoRR*, abs/1612.06553, 2016.

[2] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. *CoRR*, abs/1511.06335, 2015.

[3] Andreas Molisch, A Kuchar, J Laurila, K Hugl, and R Schmalenberger. *Geometry-based directional model for mobile radio channels -principles and implementation*. European Transactions on Telecommunications, 2003.

[4] 3GPP. Universal Mobile Telecommunications System (UMTS), Spatial channel model for Multiple Input Multiple Output (MIMO) simulations. Technical Specification (TS) ETSI TR 125 996, 3rd Generation Partnership Project (3GPP), 04 12, Sep 2014. Version TR 25.996.

[5] "Yonina C. Eldar". *"Compressed Sensing Theory and Applications"*. "Cambridge University Press ", 2012.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[7] Robert Tibshirani. Regression shrinkage and selection via the lasso. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 58:267–288, 1994.

[8] Exploring deep learning. `http://www.rsipvision.com/exploring-deep-learning/`.

[9] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[10] Schematic structure of an autoencoder with 3 fully connected hidden layers. `https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_structure.png`.

[11] Oleksii Kuchaiev and Boris Ginsburg. Training deep autoencoders for collaborative filtering. *CoRR*, abs/1708.01715, 2017.

[12] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high-dimensional data. In *In New Vistas in Statistical Physics: Applications in Econophysics, Bioinformatics, and Pattern Recognition*. Springer-Verlag, 2003.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[14] Mark Borgerding and Philip Schniter. Onsager-corrected deep networks for sparse linear inverse problems. *CoRR*, abs/1612.01183, 2016.

[15] Fredrik Rusek, Daniel Persson, Buon Kiong Lau, Erik G. Larsson, Thomas L. Marzetta, Ove Edfors, and Fredrik Tufvesson. Scaling up MIMO: opportunities and challenges with very large arrays. *CoRR*, abs/1201.3210, 2012.

[16] Christian R. Berger, Zhaohui Wang, Jianzhong Huang, and Shengli Zhou. Application of compressive sensing to sparse channel estimation. *Comm. Mag.*, 48(11):164–174, November 2010.

[17] Ansuman Adhikary, Junyoung Nam, Jae Young Ahn, and Giuseppe Caire. Joint spatial division and multiplexing. *CoRR*, abs/1209.1402, 2012.

[18] Waheed U. Bajwa, Jarvis Haupt, Akbar M. Sayeed, and Robert Nowak. Compressed channel sensing: A new approach to estimating sparse multipath channels.

[19] E.J. Candès and M.B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[20] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59, 08 2006.

[21] Xiongbin Rao and Vincent K. N. Lau. Distributed compressive CSIT estimation and feedback for FDD multi-user massive MIMO systems. *CoRR*, abs/1405.2786, 2014.

[22] Andreas Molisch and Fredrik Tufvesson. Propagation channel models for next-generation wireless communications systems. *IEICE*, E97B:2022–2034, 2014.

[23] Maximilian Arnold, Sebastian Dörner, Sebastian Cammerer, and Stephan ten Brink. On deep learning-based massive MIMO indoor user localization. In *SPAWC*, pages 1–5. IEEE, 2018.

[24] Steffen Limmer and Slawomir Stanczak. Optimal deep neural networks for sparse recovery via laplace techniques. *CoRR*, abs/1709.01112, 2017.

[25] Timothy J. O'Shea, Johnathan Corgan, and T. Charles Clancy. Unsupervised representation learning of structured radio communication signals. *CoRR*, abs/1604.07078, 2016.

[26] Mark Borgerding and Philip Schniter. Onsager-corrected deep learning for sparse linear inverse problems. *CoRR*, abs/1607.05966, 2016.

[27] Mark Borgerding and Philip Schniter. Onsager-corrected deep networks for sparse linear inverse problems. *CoRR*, abs/1612.01183, 2016.

[28] Chris Metzler, Ali Mousavi, and Richard Baraniuk. Learned d-amp: Principled neural network based compressive image recovery. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1772–1783. Curran Associates, Inc., 2017.

[29] Omar El Ayach, Sridhar Rajagopal, Shadi Abu-Surra, Zhouyue Pi, and Robert W. Heath Jr. Spatially sparse precoding in millimeter wave MIMO systems. *CoRR*, abs/1305.2460, 2013.

[30] A. M. Sayeed. Deconstructing multiantenna fading channels. *IEEE Transactions on Signal Processing*, 50(10):2563–2579, Oct 2002.

[31] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, New York, NY, USA, 2005.

[32] Mohsen Bayati and Andrea Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *CoRR*, abs/1001.3448, 2010.

[33] David L. Donoho, Arian Maleki, and Andrea Montanari. Message passing algorithms for compressed sensing. *CoRR*, abs/0907.3574, 2009.

[34] Yongyi Mao, Frank R. Kschischang, and Brendan J. Frey. Convolutional factor graphs as probabilistic models. *CoRR*, abs/1207.4136, 2012.

[35] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *CoRR*, abs/1803.04311, 2018.

[36] Yoshua Bengio. Learning deep architectures for ai, 7 June 2013.

[37] Li Deng and Dong Yu. *Deep Learning: Methods and Applications*. Now Publishers Inc., Hanover, MA, USA, 2014.

[38] "R. Bellman". *"Adaptive Control Processes: A Guided Tour, Princeton University Press"*. "The Princeton Legacy Library", 1961.

[39] Mark A. Davenport, Deanna Needell, and Michael B. Wakin. Signal space cosamp for sparse recovery with redundant dictionaries. *CoRR*, abs/1208.0353, 2012.

[40] Deanna Needell and Joel A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Commun. ACM*, 53(12):93–100, December 2010.