Research Internship Report

25/04/2016 ∼ 25/09/2015

Deep Learning and Reality Gap

Author:

FENG Siyao

A report submitted in fulfilment of the requirements for the degree of Master in Computer Science

at

Telecom Paristech

September 2016

Supervisors

Scientific Supervisors:

Michele SEBAG

Head of Équipe A-O - LRI

Tel : 01.69.15.42.33 E-mail : sebag@lri.fr

Academic Supervisor:

Albert BIFET

Associate Professor in Big Data

Télécom ParisTech

E-mail : albert.bifet@gmail.com

Acknowledgements

The internship opportunity I had with TAO group was a great chance for learning and for research experience. It's here at TAO I strengthened my willing to future research careers. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am sincerely grateful for having a chance to meet so many wonderful people who led me through this five-month internship.

First and foremost, I would like to express my sincere gratitude to my supervisor Mme. Michele SEBAG, for the continuous support of my internship, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this report. I could not have imagined having a better superadvisor and mentor for my internship.

My sincere thanks also goes to M. Gaetan Marceau Caron, who provided me his knowledge and experience and precious time, only with his help I can get better work enviroment.

Last but not the least, I would like to thank all the TAO team members. Thank you for the fabulous leisure time we had together!

Contents

List of Figures

Chapter 1

Introduction

Deep learning is part of a broader family of machine learning methods based on learning representations of data. In the recent years, Artificial Neural Nerwork has achieved a success in the domains like Computer Vision, Natural Language Processing.

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data. Some of the representations are inspired by advances in neuroscience and are loosely based on interpretation of information processing and communication patterns in a nervous system, such as neural coding which attempts to define a relationship between various stimuli and associated neuronal responses in the brain.[1]

In the field of neuroscience, the data of electroencephlogram data(EEG Data) are being used in the brain computer interface(BCI), for example, using the EEG data to control the wheelchair or the gamestick. But they are facing the problem that the EEG data has a high variability problem. The data is not only depends on the subject, but also depends on the session. Which is to say that the EEG data collected in the morning and in the evening from the same subject is different. So it asks to train a model depends on a subject and then to adjust the model for different session.

Domain Adaptation[2][3] is a field associated with machine learning and transfer learning. This scenario arises when we aim at learning from a source data distribution a well performing model on a different (but related) target data distribution. For instance, one of the tasks of the common spam filtering problem consists in adapting a model from one user (the source distribution) to a new one who receives significantly different emails (the target distribution). Note that, when more than one source distribution is available the problem is referred to as multi-source domain adaptation.[4]

In this internship, I work under the supervision of Dr. Michele Sebag and Dr. Gaetan Marceau Caron at Laboratoire de Recherche en Informatique (LRI), Gif-sur-Yvette, France. We aim at using domain adaptation to eliminate the variability of the EEG data.

1.1 Outline

In the first part(chapter 2) the contexte and the objectif of this internship will be introduced, the detail of EEG data, domain adaptation will be explained.

chapter 3 will show the state of the art on the field of deep learning or more presicely the domain adaptation.

The following chapter(chapter 4) focuses on the framework of this internship, the program I used to train the model.

chapter 5 will give you the experiments I have done and the results achieved. Some comments will also follow the results.

The final chapter 6 will be the discussion of the experiments results of using domain adaptation on the EEG data to eliminate the variability problem. Then the perspective of the subject will be given.

Chapter 2

Context and Objective

In this chapter, I will first introduce the context of this internship which is the Brain Computer Interface (BCI) and the new deep learning technique Domain Adaptation. From the problems identified for the EEG data, we propose the approach by using Domain Adaptation to eliminate the variabilty problem in the EEG data.

2.1 Context

2.1.1 BCI

Brain Computer Interface Brain-computer interface is a method of communication based on neural activity generated by the brain and is independent of its normal output pathways of peripheral nerves and muscles. The neural activity used in BCI can be recorded using invasive or noninvasive techniques. The goal of BCI is not to determine a person's intent by eavesdropping on brain activity, but rather to provide a new channel of output for the brain that requires voluntary adaptive control by the user.[5] The potential of B-CI systems for helping handicapped people is obvious, the wheelchair controled by EEG signal is shown at Figure 2.1. There are several computer interfaces designed for disabled people.[6] Most of these systems, however, require some sort of reliable muscular control such as neck, head, eyes, or other facial muscles. It is important to note that although requiring only neural activity, BCI utilizes neural activity generated voluntarily by the user. Interfaces based on involuntary neural activity, such as those generated during an epileptic seizure, utilize many of the same components and principles as BCI, but are not included in this field. BCI systems, therefore, are especially useful for severely disabled, or locked-in, individuals with no reliable muscular control to interact with their surroundings.

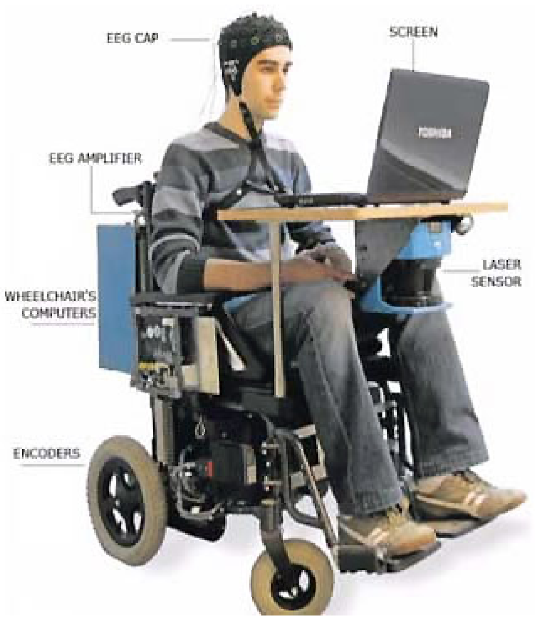

FIGURE 2.1: A wheelchair controled by EEG signal

But there's a main limitation for BCI is that the EEG data have a high inter-variability and intra-variability. EEG data are highly dependent on subject, so it requires to train the BCI on each subject. Also, it's highly dependent on session, EEG collected are varies from different time(like morning and evening), and even during a same work session, so it also requires to train on each subject for each session, which is difficult to realize and redundant. Thus we should find an approach to eliminate the subject-independent and session independent problem of EEG data.

2.1.2 Domain Adaptation

Top-performing deep architectures are trained on massive amounts of labeled data. In the absence of labeled data for a certain task, domain adaptation often provides an attractive option given that labeled data of similar nature but from a different domain (e.g. synthetic images) are available. The paper by [Ganin et al. 2015][7] propose a new approach to domain adaptation in deep architectures that can be trained on large amount of labeled data from the source domain and large amount of unlabeled data from the target domain (no labeled target domain data is necessary).

As the training progresses, the approach promotes the emergence of "deep" features that are

1. Discriminative for the main learning task on the source domain
2. Invariant with respect to the shift between the domains.

The paper[7] shows that this adaptation behavior can be achieved in almost any feed-forward model by augmenting it with few standard layers and a simple new **gradient reversal layer**. The resulting augmented architecture can be trained using standard back-propagation. Overall, the approach can be implemented with little effort using any of the deep-learning packages. The method performs very well in a series of image classification experiments, achieving adaptation effect in the presence of big domain shifts and outperforming previous state-of-the-art on Office datasets.

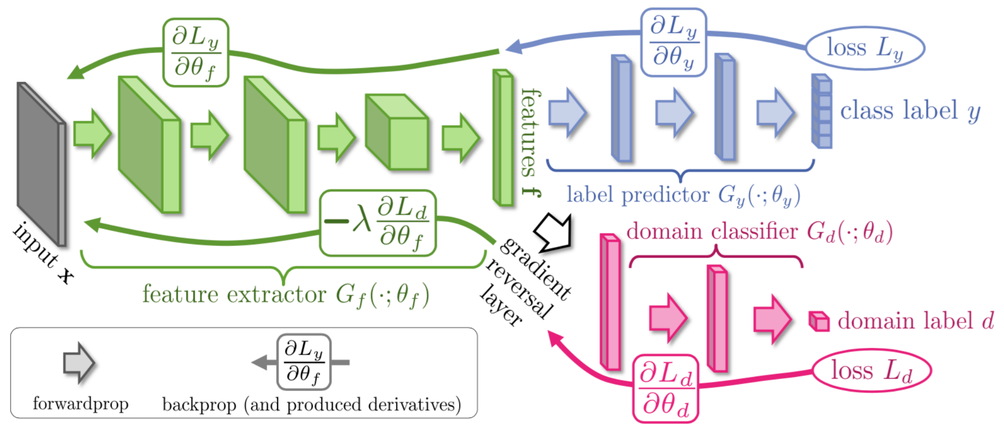
$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
domain classifier $G_d(\cdot;\theta_d)$
input $\mathbf{x}$
gradient reversal layer
feature extractor $G_f(\cdot;\theta_f)$
domain label d
$\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

FIGURE 2.2: Domain Adaptation Model Architecture

This method involves two domains, the source domain and the target domain. For example, the source domain can be MNIST dataset (handwritten digit database) and the target domain can be the SVHN dataset (The Street View House Numbers). Formally, the source is described through a training set, made of (instance, label) pairs:

$$\mathcal{S} = (x_i, y_i), x_i \in X, y_i \in Y, i \in [1, n]$$

where X is the instance space (e.g. the vectors of pixel values) and Y is the label space (e.g. the digit numbers). The target is described through a test set made of instances:

$$\mathcal{T} = x_i', x_i' \in X, i \in [1, n]$$

The general goal is to build a classifier h for the source domain and for the target domain **without** gathering the target doamin labels.

The paper has proposed a model for the domain adaptation as Figure 2.2. The proposed architecture includes a deep feature extractor (green) and a deep label predictor (blue), which together form a standard feed-forward architecture. Unsupervised domain adaptation is achieved by adding a domain classifier (red) connected to the feature extractor via a gradient reversal layer that multiplies the gradient by a certain negative constant during the backpropagationbased training. Otherwise, the training proceeds in a standard way and minimizes the label prediction loss (for source examples) and the domain classification loss (for all samples). Gradient reversal ensures that the feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.

During the learning stage, we aim to minimize the label prediction loss on the annotated part (i.e. the source part) of the training set, and the parameters of both the feature extractor and the label predictor are thus optimized in order to minimize the empirical loss for the source domain samples. This ensures the discriminativeness of the features $\mathbf{f}$ and the overall good prediction performance of the combination of the feature extractor and the label predictor on the source domain. At the same time, we want to make the features $\mathbf{f}$ domain-invariant which is we want the representation of the two domains to be similar.

At training time, in order to obtain domain-invariant features, we seek the parameters of the feature mapping that maximize the loss of the domain classifier (by making the two feature distributions as similar as possible), while simultaneously seeking the parameters of the domain classifier that minimize the loss of the domain classifier. So as we can see, the parameter λ of the Gradient Reversal Layer controls the trade-off between the two objectives that shape the features during learning.

2.2 Objective

After the problem identified of the EEG data using in the BCI (high inter-variability and high intra-variability) .the objective of this internship thus is to build a subject-dependent, session-dependent representations of the EEG data. The sought representation must achieve a lossy compression of the signal to enabling the reconstruction of the brainwave data in order to further use in the BCI, besides, the representation of the signals gathered along different sessions follows the same distribution, or distribution as similar as possible. There's also a difficulty is that this reconstruction must preserve the complex spatio-temporal-frequency structure of the EEG data.

Chapter 3

State of the Art

Recently, some scientist use deep learning network on the EEG data to do the emotion recognition, also some other domain adaptation methods have been proposed over the recent years, I will list the abstract of this papers in this chapter to give you the state of the art for this subject.

3.1 EEG-based emotion classification using deep belief networks

Abstract:[8] In recent years, there are many great successes in using deep architectures for unsupervised feature learning from data, especially for images and speech. In this paper, we introduce recent advanced deep learning models to classify two emotional categories (positive and negative) from EEG data. We train a deep belief network (DBN) with differential entropy features extracted from multichannel EEG as input. A hidden markov model (HMM) is integrated to accurately capture a more reliable emotional stage switching. We also compare the performance of the deep models to KNN, SVM and Graph regularized Extreme Learning Machine (GELM). The average accuracies of DBN-HMM, DBN, GELM, SVM, and KNN in our experiments are 87.62%, 86.91%, 85.67%, 84.08%, and 69.66%, respectively. Our experimental results show that the DBN and DBN-HMM models improve the accuracy of EEG-based emotion classification in comparison with the state-of-the-art methods.

3.2 EEG-Based Emotion Recognition Using Deep Learning Network with Principal Component Based Covariate Shift Adaptation

Abstract:[9] Automatic emotion recognition is one of the most challenging tasks. To detect emotion from nonstationary EEG signals, a sophisticated learning algorithm that can represent high-level abstraction is required. This study proposes the utilization of a deep learning network (DLN) to discover unknown feature correlation between input signals that is crucial for the learning task.The DLN is implemented with a stacked autoencoder (SAE) using hierarchical feature learning approach. Input features of the network are power spectral densities of 32-channel EEG signals from 32 subjects. To alleviate overfitting problem, principal component analysis (PCA) is applied to extract the most important components of initial input features. Furthermore, covariate shift adaptation of the principal components is implemented to minimize the nonstationary effect of EEG signals. Experimental results show that the DLN is capable of classifying three different levels of valence and arousal with accuracy of 49.52component based covariate shift adaptation enhances the respective classification accuracy by 5.55provides better performance compared to SVM and naive Bayes classifiers.

3.3 Domain Adaptation via Transfer Component Analysis

Abstract:[10] Automatic emotion recognition is one of the most challenging tasks. To detect emotion from nonstationary EEG signals, a sophisticated learning algorithm that can represent high-level abstraction is required. This study proposes the utilization of a deep learning network (DLN) to discover unknown feature correlation between input signals that is crucial for the learning task.The DLN is implemented with a stacked autoencoder (SAE) using hierarchical feature learning approach. Input features of the network are power spectral densities of 32-channel EEG signals from 32 subjects. To alleviate overfitting problem, principal component analysis (PCA) is applied to extract the most important components of initial input features. Furthermore, covariate shift adaptation of the principal components is implemented to minimize the nonstationary effect of EEG signals. Experimental results show that the DLN is capable of classifying three different levels of valence and arousal with accuracy of 49.52component based covariate shift adaptation enhances the respective classification accuracy by 5.55provides better performance compared to SVM and naive Bayes classifiers.

3.4 Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach

Abstract:[11] The exponential increase in the availability of online reviews and recommendations makes sentiment classification an interesting topic in academic and industrial research. Reviews can span so many different domains that it is difficult to gather annotated training data for all of them. Hence, this paper studies the problem of domain adaptation for sentiment classifiers, hereby a system is trained on labeled reviews from one source domain but is meant to be deployed on another. We propose a deep learning approach which learns to extract a meaningful representation for each review in an unsupervised fashion. Sentiment classifiers trained with this high-level feature representation clearly outperform state-of-the-art methods on a benchmark composed of reviews of 4 types of Amazon products. Furthermore, this method scales well and allowed us to successfully perform domain adaptation on a larger industrial-strength dataset of 22 domains.

Chapter 4

Program

The approach we proposed for this subject is using domain adaptation to eliminate the variability problem in the EEG data. In this chapter, firstly I will present the structure of EEG data we used for this internship. Then I will present the adaption of the DANN structure to our case. Thirdly is defining the criteria for our Deep Neural Network.

4.1 EEG data

The EEG data we use is provided by Dr. Fabrizio De Vico Fallani from ICM in paris(Institut du Cerveau et de la Moelle Epinière). The EEG dataset contains 18 files. They are divided into two sessions from 9 subject.

The EEG signals refer to recordings of one minute during a state of no-task with closed eyes, or say "resting state". The data is collected by a helmet on the head. The helmet contains 56 electrodes all over the head, as shown in the Figure 4.1. The sampling frequency is equal to 200 Hz, so each file contains a matrix of 56 x 12000.

The preprocessing for the data is normalization. I have linearly normalize the sensor values to [0,1] by:

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

where x_{min} and x_{max} are the minimum and maximum sensor value on the session.

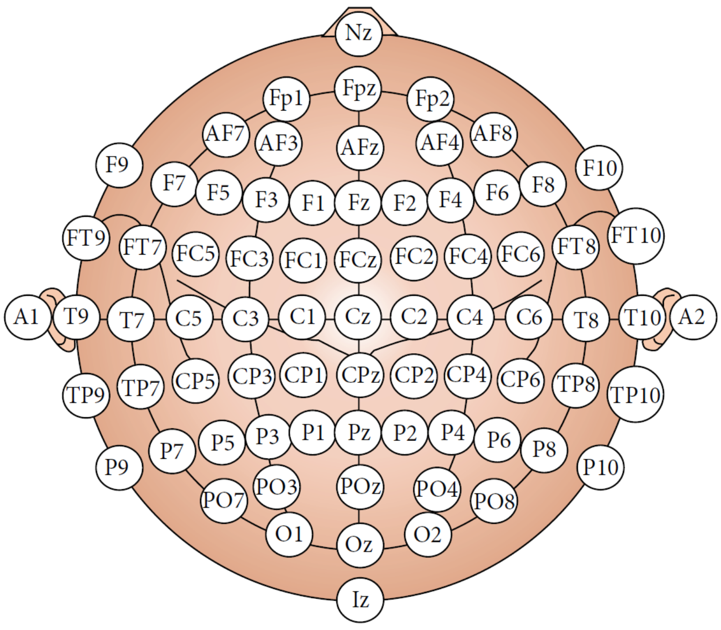

FIGURE 4.1: Location of the electrodes over the head of the subject

4.2 Architecture of Dann EEG

The principle for our adaption is that we treat session one of each subject EEG data as source domain in the Domain Adaptation, and the session two as the target domain. Once we using the DANN for training, the model will not be able to distinguish which session the input data belongs to, this is to say we can't tell the difference between session one and session two. So after reconstruction, the EEG data from two sessions become indistinguishable.

The base of our DANN is auto-encoder. An autoencoder, autoassociator or Diabolo network[12] is an artificial neural network used for unsupervised learning of efficient codings.[13] The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. Recently, the autoencoder concept has become more widely used for learning generative models of data.[14]

Considering that the input data dimension is 56, so in the input layer and output layer of our Neural Network, the size will be 56. In this internship, which is as a first approach to solve the problem, we use a simple structure with 1 hidden layer like 56 - n -56, where n varies from 1 to 56.

Above is the base structure of our neural network which is an auto-encoder. Then we need to add the gradient reversal layer which connected to the hidden layer. So the structure is shown in Figure 4.2

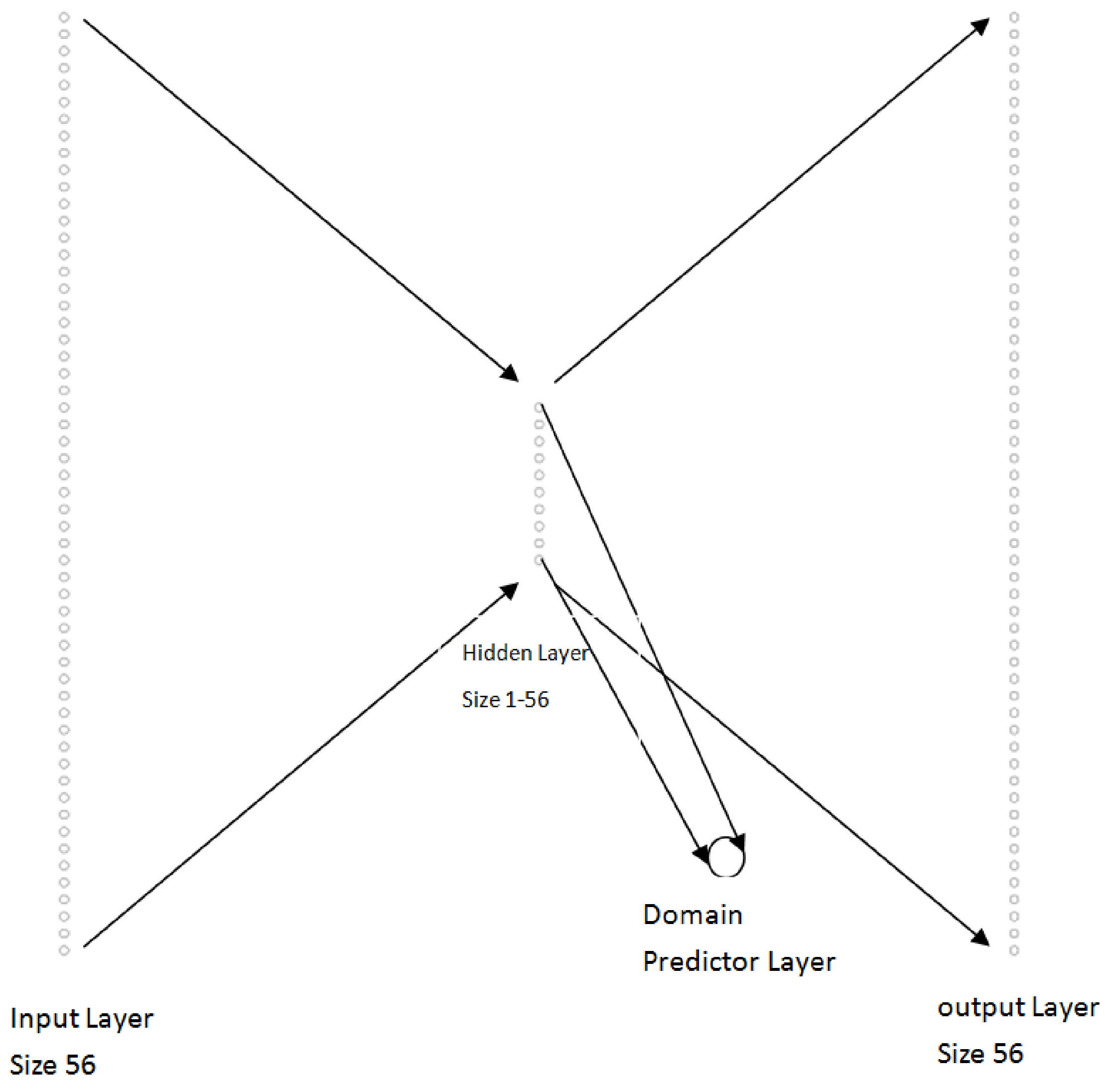

Figure 4.2: Structure of DANN for EEG data

At every neuron, the activation function will using **Sigmoid**.

4.3 Criteria

For our neural network, there are 2 purposes:

1. Minimize the reconstruction error of the EEG data.

2. Maximize the domain(session) prediction error, so that the sessions cannot be discriminated, even with the best classifier.

Considering the requirements above, two criterion functions will be defined in the program.

1. For the reconstruction loss of EEG, we want to be able to reconstruct the initial EEG data, so we choose the MSE cost function(mean square error) like:

$$loss(x, \tilde{x}) = \frac{1}{n} \times \sum_{i=1}^{56} |x_i - \tilde{x_i}|^2$$

2. The second criterion is session-invariant. We will assess this criterion by measuring the BCE(Binary Cross Entropy) criterion of the domain(session) predictor. So the loss function will be:

$$loss(o, t) = -\frac{1}{n} \sum (t[i] \times \log o[i] + (1 - t[i]) \times \log(1 - o[i])$$

where o is original prediction and t is target prediction. This criterion will give the entropy of the error, so when the result have a lot surprise, criterion will be large, otherwise the criterion will be 0.

4.4 Framework

To realize this neural network, I have using the **Torch** framework. Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.

In the torch, there's already a **gradient reversal layer** prepared to use, so we can easy connect the auto-encoder with the gradient reversal layer to build our neural network.

In the program, I have defined three separate neural network, which are:

1. **Encoder**, the structure is 56 - n ($n \in [1, 56]$), this layer is used like feature extractor in the DANN, it also serves as a conpression part.

2. **Decoder**, the structure is n - 56 ($n \in [1, 56]$), this layer is used like label predictor in the DANN, it also serves to decode the data to get the initial data.

3. **Gradient reversal Layer**, the structure is n - 1 ($n \in [1, 56]$), this layer is used to predict which domain the instance is belonged to. With the help of negative gradient for this layer, we can maximize the error to make the two session indistinguishable.

Then in the training phase, the EEG data will be used as input data for Encoder, the output of Encoder will then be used as the input data of Decoder and Gradient reversal layer.

Here below is the Lua code used to build our neural network.

```
//Definition of the encoder
featExtractor = nn.Sequential()
featExtractor:add(nn.Linear(nInputEEG,opt.hiddenLayerUnits))
featExtractor:add(nn.Sigmoid())

//Definition of the decoder
labelPredictor = nn.Sequential()
labelPredictor:add(nn.Linear(opt.hiddenLayerUnits,nInputEEG))
labelPredictor:add(nn.Sigmoid())

//Definition of the domain classifier
domainClassifier = nn.Sequential()
domainClassifier:add(nn.GradientReversal(opt.domainLambda))
domainClassifier:add(nn.Linear(opt.hiddenLayerUnits,1))
domainClassifier:add(nn.Sigmoid())
```

Chapter 5

Experiments and Results

There are two principle parts for my experiments. First is to test the EEG data with an autoencoder(didn't add the gradient reversal branch), to see the reconstruction error in order to decide the hidden layer units and other hyper-parameters for training. Second, using all the two sessions EEG data and add the gradient reversal branch to fully built the DANN, and see the performance of the neural network for our objectives.

5.1 Architecture of the Neural Network

For a neural network, first thing how to decide is the architecture of our neural network, as the input layer and output layer architecture have already been fixed(size 56). Two things left to decide:

1. Number of hidden layers
2. Number of units in a hidden layer

As we know, more hidden layers in a neural network will always get better performance like lower mean squared error for reconstruction, but beside, it will always consume more computation time. So choosing the architecture is a trade-off between performance and computation time. In fact, we found that for a architecture with more than one hidden layer, the computation time will be several hours for one training, but we need also to vary the number of hidden units in each layer. In order to do more experiments, we choose at last the architecture like 56-n-56 ($n \in [1, 56]$).

5.2 Learning Rate Strategy

In the training phase of an neural network, the hyper parameter affecting the computation time and the converge speed is the learning rate μ. The more larger learning rate is, the faster will the neural network trains; the smaller learning rate is, the more accurate the result will be. Instead of setting an constant learning rate, we can adjust the value of learning rate related to the training status. For example, when the error isn't stable, we can set a larger learning rate to accelerate the descent; when the error become stable, we can set a smaller learning rate to let the result more stable.

Tracing the result error with every epoch of training can also help us to determine the maximum iteration needed for converge, so we can find a optimal point between the computation time and the performance.

5.2.1 Learning Rate Strategy for Auto-encoder

For the auto-encoder without the gradient reversal branch, there are several strategy of learning rate strategy proposed by Dr. Sebag and Dr. Gaetan. Which are:

1. Exp1: constant learning rate, $\mu = 0.1$
2. Exp2: constant learning rate, $\mu = 1$
3. Exp3: reciprocal of square of epoch number, $\mu(i_{epoch}) = \frac{1}{\sqrt{i_{epoch}}}$
4. Exp4: reciprocal of epoch number, $\mu(i_{epoch}) = \frac{1}{i_{epoch}}$
5. Exp5: adaptive learning rate, in this strategy, we will adjust the learning rate by comparing the current loss with previous loss,
    - if the error decreases, multiply the previous learning rate by 1.2
    - if the error increases, divide the previous learning rate by 2

The result is shown at Figure 5.1. The structure of this experiments is 56-5-56, max epoch = 2000, mini-batch size is 50. The training time for each experiment is 40 mins.

The training set we use are example [1,6000] from subject 1 session 1 and valid set are examples [6001,12000]. X axis is the number of epoch and Y axis is the training MSE.

We can see that experiment 1 converges very fast but with a high MSE loss. Experiment 2 are about to converging after 2000 epochs and the MSE loss is small, this looks an good strategy but it takes too much time to converge.

Experiment 3 didn't converge after 2000 epoch and the MSE loss is large. Expriment 4 didn't converge and the MSE loss is average. Experiment 5 which using the adaptive schedule converges after 300 epoch with a very small MSE loss.

From the schema we can tell that the adaptive schedule is the best learning rate strategy because it converges very fast (<300 epochs) and get the smallest MSE loss, except the MSE loss oscillates at the beginning of the training which is normal because the learning rate oscillates a lot at the beginning.

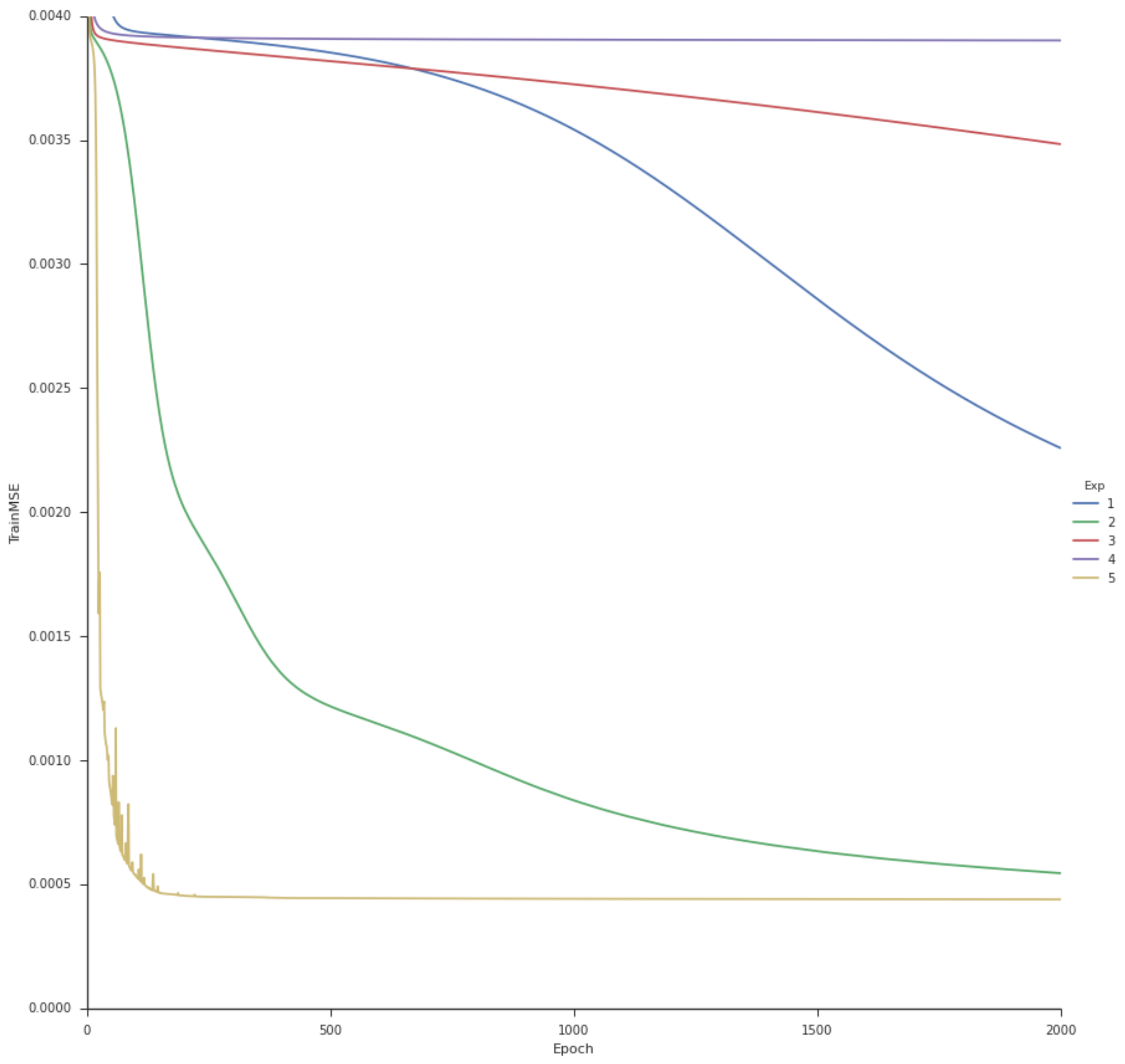

FIGURE 5.1: Train MSE in function with the Epoch number for different learning rate strategy in Auto-encoder

5.2.2 Learning Rate Strategy for DANN

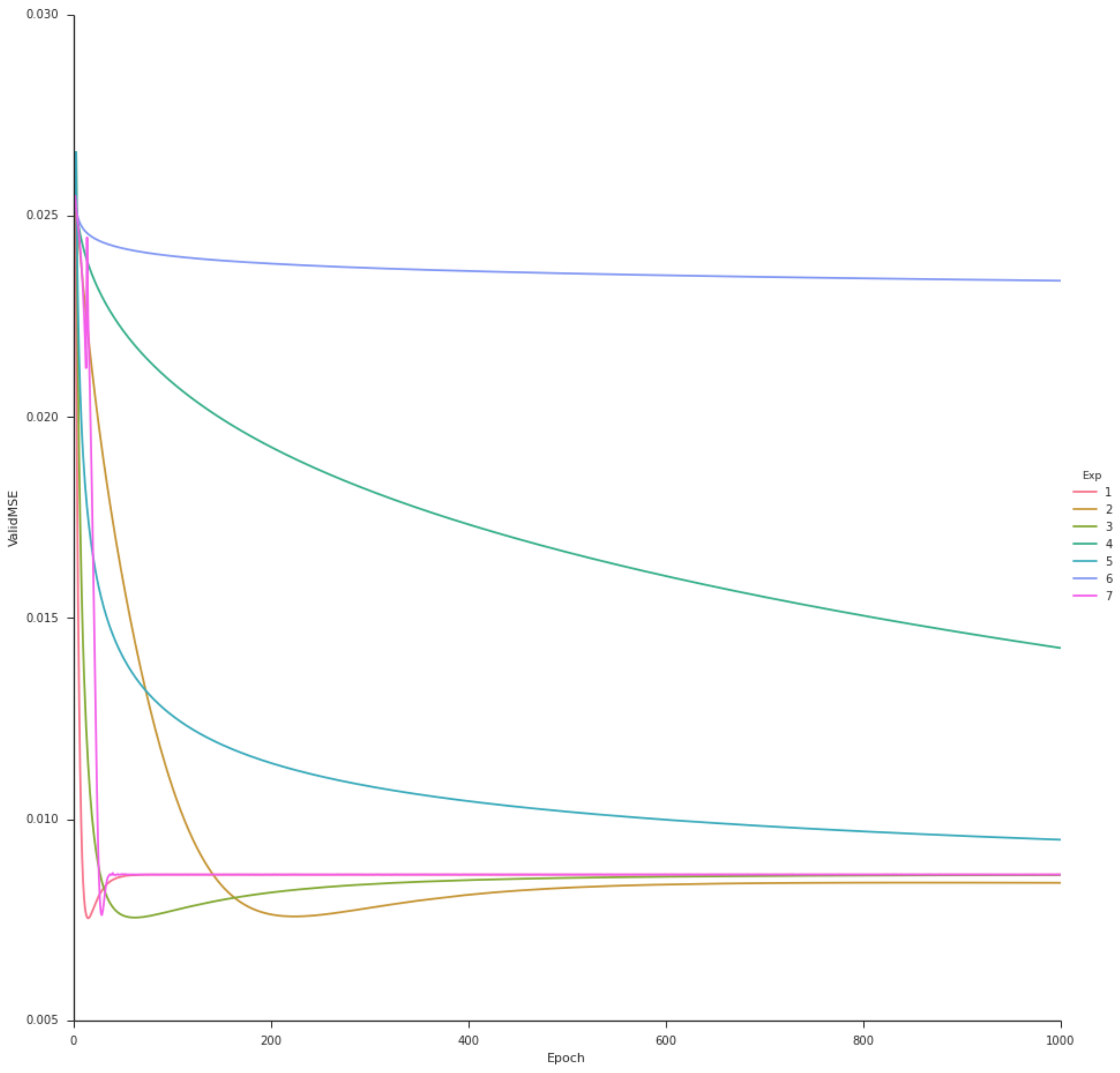

FIGURE 5.2: Train MSE in function with the Epoch number for different learning rate strategy in DANN

As the same above, we have also defined several strategy for **DANN**, those are:

1. Exp1: constant learning rate, $\mu = 1$
2. Exp2: constant learning rate, $\mu = 0.1$
3. Exp3: reciprocal of square of epoch number, $\mu(i_{epoch}) = \frac{1}{\sqrt{i_{epoch}}}$
4. Exp4: reciprocal of square of epoch number, $\mu(i_{epoch}) = \frac{0.1}{\sqrt{i_{epoch}}}$
5. Exp5: reciprocal of epoch number, $\mu(i_{epoch}) = \frac{1}{\sqrt{i_{epoch}}}$
6. Exp6: reciprocal of epoch number, $\mu(i_{epoch}) = \frac{0.1}{\sqrt{i_{epoch}}}$

7. Exp7: adaptive learning rate schudule

The results are shown in Figure 5.2.

The conclusion is pretty as like in the auto-encoder, adaptive strategy still works the best from all the strategies.

In the following experiments, I will use adaptive strategy in the training phase.

5.3 Auto-encoder experiments results

First experiments is to see the impact of the hidden layer units for an auto-encoder, results shown at Figure 5.3 and Figure 5.4.

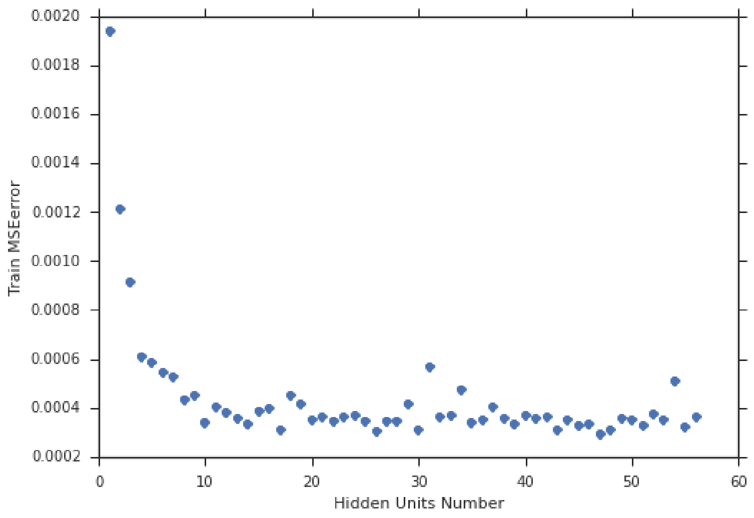

FIGURE 5.3: Train reconstruction error with different hidden layer units

This schema gives the MSE error of the auto-encoder with different hidden layer units. The network is trained with mini-batch size=50, learning rate is adaptive, max epoch=1000, structure is 56-n-56.The n I tested varies from 1 to 56. Result is average of 5 runs. The training time for one value of hidden layer units are 50 mins, so the time total used for training is 46 hours.

The training set we use are example[1,6000] from subject 1 session 1 and valid set are examples [6001, 12000] from subject 1 session 1. So in the schema, x axis is the hidden layer units from 1 - 56, on the y axis is the training and valid MSE error.

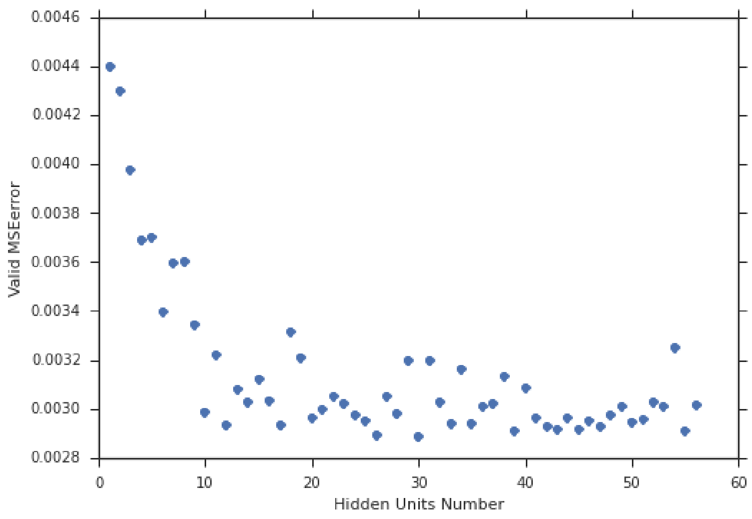

FIGURE 5.4: Valid reconstruction error with different hidden layer units

We can see that the MSE basically decrease when the hidden layer units increases, which is normal for an auto-encoder. Else, we can see before n=8, the error is more large than n¿8, which means EEG data is more than 4 dimensions, but the error is basically same if n bigger than 10. For when n=56, ideally, the error should be 0 if we initialize the weight parameters in a matrix like:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

From the results, we can see that it's better to choose the n larger than 10 to ensure that we can reconstruct the initial information from EEG data.

5.4 DANN experiments results

After adding the gradient reversal branch to the auto-encoder, at every back-propagation, the parameters will be updated both by the two branch (label predictor and domain predictor). Label predictor is for minimize the reconstruction loss of the EEG data, where domain predictor is aims at maximizing the domain classification error. So the result will be a trade-off between this two predictor.

Two parameters will be varied in the following experiments:

- The number of units in the hidden layer, this parameter helps us to find the optimal architecture of the DANN to balance the computation time and the performance
- The parameter lambda λ in the gradient reversal layer. λ means how much the domain predictor will affect the gradient descent in the back-propagation.

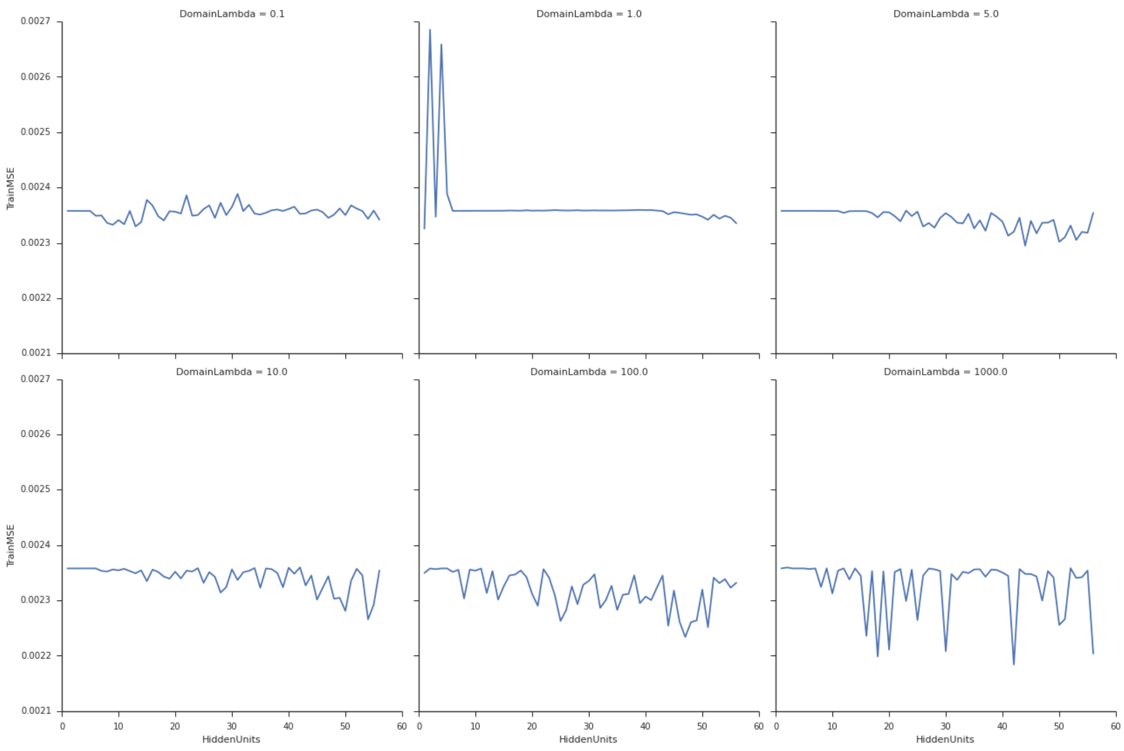

FIGURE 5.5: Train reconstruction error with different hidden layer units in DANN

These schema(Figure 5.5 and Figure 5.6) gives the train and valid reconstruction error with different hidden layer units in DANN. The network is trained with mini-batch size=50, learning rate is adaptive, max epoch=300, structure is 56-n-56.The n I tested varies from 1 to 56. Result is average of 5 runs. The training time for one value of hidden layer units are 70 mins.

The source domain training set we use are example[1,6000] from subject 1 session 1 and valid set are examples [6001, 12000] from subject 1 session 1. The target domain training set are examples [1,6000] from subject 1 session 2, valid set are examples [6001, 12000] from subject 1 session 2.In the schema, x axis is the hidden layer units from 1 - 56, on the y axis is the training and valid MSE error.

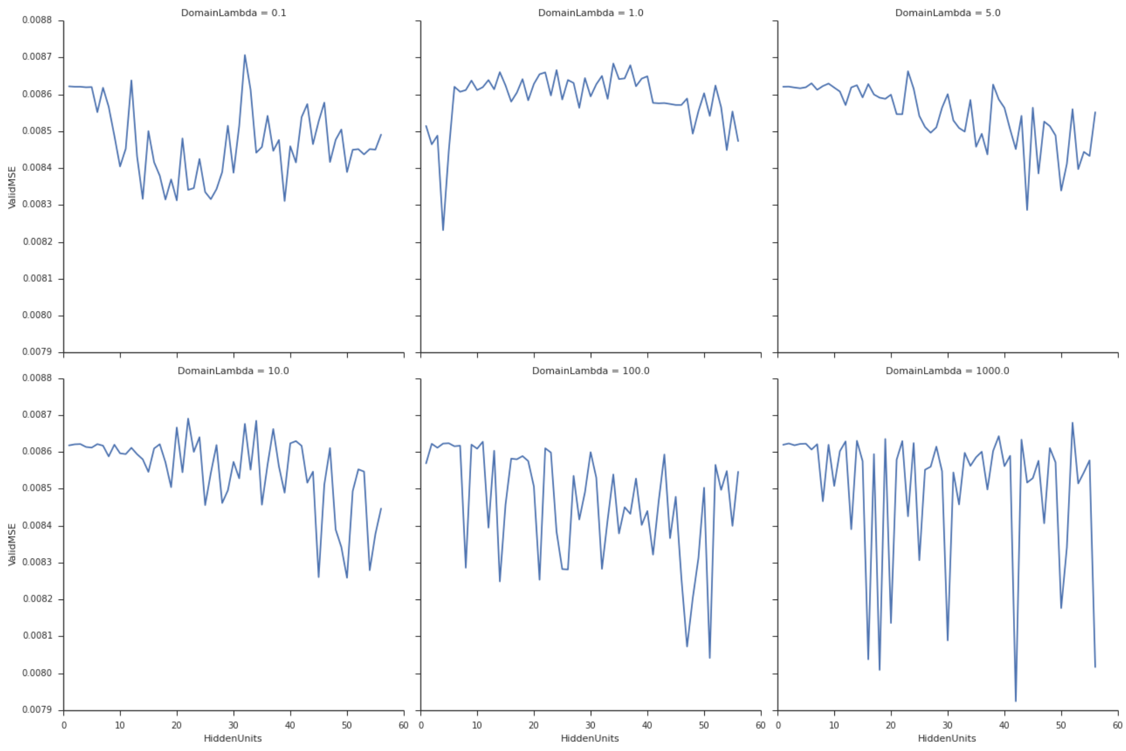

FIGURE 5.6: Valid reconstruction error with different hidden layer units in DANN

For $\lambda = 1$, the MSE loss is pretty stable except when $n \in [1, 8]$. With the value of λ increases, the MSE loss become more oscillated. But the base losses are basically the same, when the hidden layer units number are large, it can sometimes reach a smaller MSE loss. In our inference, the MSE loss should decreases with the hidden layer units increases, the result didn't totally accord with our inference but shows the same tendency.

These schema(Figure 5.7 and Figure 5.8) gives the train and valid domain BCE loss with different hidden layer units in DANN. The network is trained with mini-batch size=50, learning rate is adaptive, max epoch=300, structure is 56-n-56.The n I tested varies from 1 to 56. Result is average of 5 runs. The training time for one value of hidden layer units are 70 mins.

The source domain training set we use are example[1,6000] from subject 1 session 1 and valid set are examples [6001, 12000] from subject 1 session 1. The target domain training set are examples [1,6000] from subject 1 session 2, valid set are examples [6001, 12000] from subject 1 session 2.In the schema, x axis is the hidden layer units from 1 - 56, on the y axis is the training and valid MSE error.

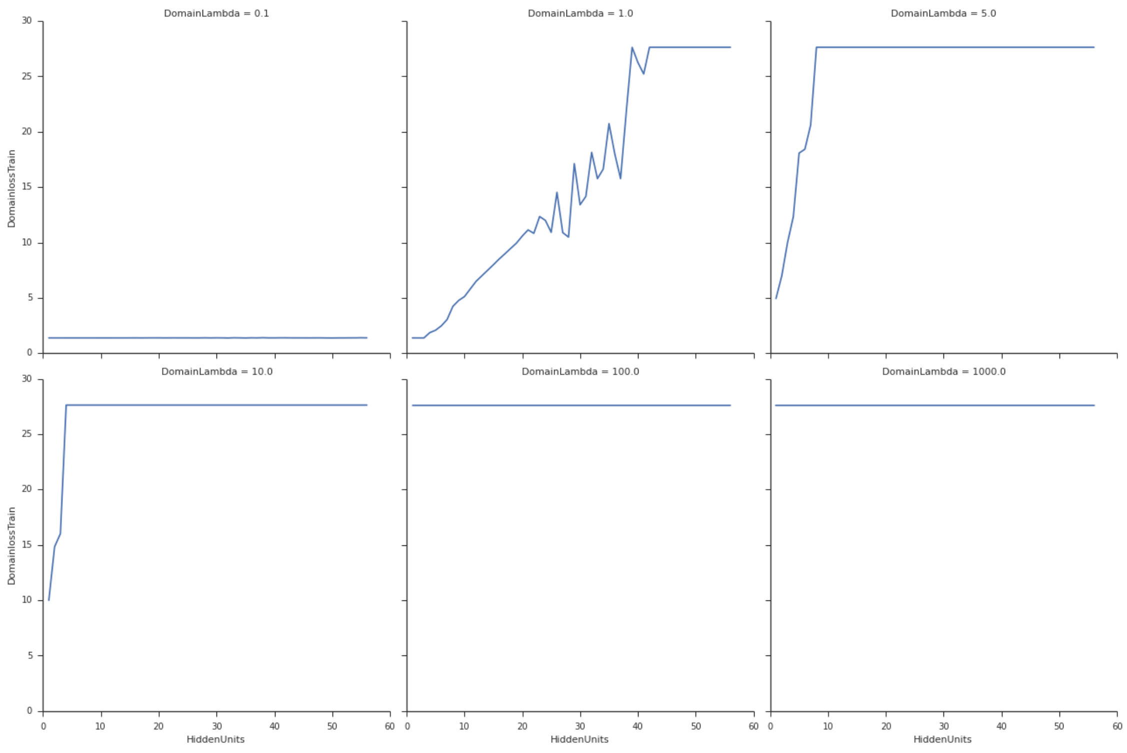

FIGURE 5.7: Domain BCE loss with different hidden layer units in DANN

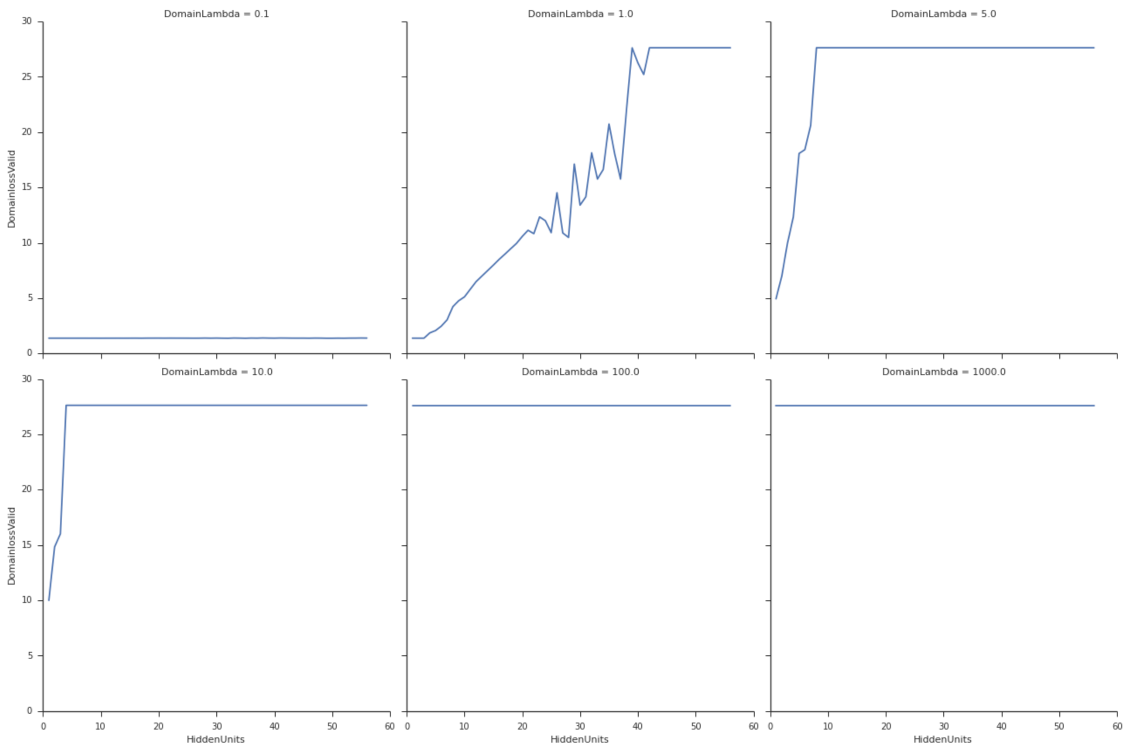

FIGURE 5.8: Domain BCE loss with different hidden layer units in DANN

For $\lambda = 1$, the domain BCE loss is positively correlated with the hidden layer units. When the λ increases from 1 to 1000, the domain BCE loss become more stable regarding the hidden units number. That is to say the influence of the architecture become smaller. When the domain BCE loss become 27, than means the domain classifier has a high error rate, so that EEG data from the two sessions become indiscriminate which is our purpose.

Chapter 6

Discussions

The experiments results are interesting and kind of beyond our assumption. Firstly, in our assumption, the the MSE loss should be negatively correlated with the number of hidden layer units, but in fact, the MSE loss may oscillated a little and have the tendency but not much. Especially when lambda λ become large, it's like hidden layer units number losses its influence on the MSE error.

Secondly, the result about domain BCE loss seems promising, it's the same for training and for valid. When the lambda λ is 0.1, we can see there's little influence about the neural network so the BCE loss is small, means we can still distinguish the session 1 and session 2. but when lambda λ become large, the domain BCE loss stay large so that means we have reach our goal (make the representation of the 2 different session become indiscriminate).

Considering the above two parts, for this special subject who provided the EEG data, we should choose a lambda λ larger than 10, and the number of hidden layer units should be [10,20] to get the balance between the performance and computation time.

Bibliography

[1] Bruno A Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[2] John S Bridle and Stephen Cox. Recnorm: Simultaneous normalisation and classification applied to speech recognition. In *NIPS*, pages 234–240, 1990.

[3] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[4] Koby Crammer, Michael Kearns, and Jennifer Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9(Aug):1757–1774, 2008.

[5] Jonathan R Wolpaw, Niels Birbaumer, William J Heetderks, Dennis J McFarland, P Hunter Peckham, Gerwin Schalk, Emanuel Donchin, Louis A Quatrano, Charles J Robinson, Theresa M Vaughan, et al. Brain-computer interface technology: a review of the first international meeting. *IEEE transactions on rehabilitation engineering*, 8(2):164–173, 2000.

[6] Ingrid Wickelgren. Brain researchers try to salvage estrogen treatments. *Science*, 302(5648):1138, 2003.

[7] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[8] Wei-Long Zheng, Jia-Yi Zhu, Yong Peng, and Bao-Liang Lu. Eeg-based emotion classification using deep belief networks. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[9] Suwicha Jirayucharoensak, Setha Pan-Ngum, and Pasin Israsena. Eeg-based emotion recognition using deep learning network with principal component based covariate shift adaptation. *The Scientific World Journal*, 2014, 2014.

[10] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22 (2):199–210, 2011.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520, 2011.

[12] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[13] Cheng-Yuan Liou, Jau-Chi Huang, and Wen-Chie Yang. Modeling word perception using the elman network. *Neurocomputing*, 71(16):3150–3157, 2008.

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.