Classification of Alzhermer Disease and Normal Congnitive Status with Recurrent Neural Networks in Resting State fMRI

Tao Sun

Ohio University

2016

Classification of Alzhermer Disease and Normal Congnitive Status with Recurrent Neural Networks in Resting State fMRI

Introduction

A human brain is a complex system composed of structural regions that are functionally specialized. Due to the conclusion that these locally segregated regions are actively interconnected even when a subject is at resting-state (Biswal, Yetkin, Haughton, & Hyde, 1995), the resting-state functional Magnetic Resonance Imaging (fMRI), which is a neuroimaging procedure that measures the changes of signals associated with blood flow, has become a prevailed tool for investigation of brain functional networks. Since functional connectivity in the brain is an significant measure that could indicates disease-induced changes in the network, it could provide assist to the diagnosis of brain diseases such as Alzheimer Disease (AD) or its early stage Mild Congnitive Impairment (MCI).

With the typical assumption that the functional networks in a brain is stationary, many diagnosis methods of MCI and AD with resting-state fMRI (rs-fMRI) model the network with correlation analysis such as Pearson's correlation, independent component analysis (Li et al., 2012). However, recent studies (Hutchison, 2013) suggest that significant temporal changes exist in functional connectivity. Thus, valuable information could be lost when connectivity estimation is based on analysis restricted to a single value obtained from the entire scanning time.

In this paper, we present a novel method to classify subjects with AD and Normal healthy Control (NC) by combining Deep Auto-Encoder (DAE) and Recurrent Neural Networks (CNN). Initially rs-fMRI images data is preprocessed and mean time series of Regions of Interest (ROIs) are extracted. Then high-dimensional time-series data is reduced to a lower dimensionality by the DAE and then splitted into multiple identical-sized sub-series. A RNN classifier is trained on the sub-series which can tag each sub-series as AD or NC. Finally, the diagnosis for a subject is made by ensemble of the outputs of the sub-series classifier. Tests shows that accuracy of the method approaches 70% on test data.

Problem Definition and Algorithm

Data set and Preprocessing

The data used for training and test of the proposed classifier are retrieved from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database. After filtering, images of 33 AD subjects and 50 NC subjects, are downloaded. With most of these subjects are scanned more than once, we have 89 AD examples and 139 NC examples in the data set.The data set are divided into training data, test data, and validation data with a ratio of 7:2:1.

The preprocess can be divided into 4 phases ((Suka, Weeb, Leea, & Shen, 2016)) :

1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. Removal of noise signa

The results of the preproces are a set of mean time series

$$F^{(n)} \in \{F|F = [\mathbf{f}_1, \cdots, \mathbf{f}_t, \cdots, \mathbf{f}_T], \mathbf{f}_t \in \mathbb{R}^R\}, n = 1, \cdots, N \ ,$$

where $N = 228$ is number of the scans, $R = 120$ is the number of ROIs, and $T = 135$ is the length of a time series.

Dimensionality Reduction with DAE

Suka et al. (2016) proposed that a stacked DAE can used as an intermediate building block for deeper models in neuroimaging analysis. DAE is an unsupervised multilayer feed-forward neural networks, the goal of which is setting a latent representation of feature vectors of its input by training a nonlinear approximation function $h(\mathbf{f}_t) \approx \mathbf{f}_t$ (Figure 1). Concretely,

$$G = [\mathbf{g}_1, \cdots, \mathbf{g}_t, \cdots, \mathbf{g}_T], \mathbf{g}_t \in \mathbb{R}^R$$

is expected to converted from its original form

$$F = [\mathbf{f}_1, \cdots, \mathbf{f}_t, \cdots, \mathbf{f}_T], \mathbf{f}_t \in \mathbb{R}^R.$$

A stacked autoencoder (Figure 2) is composed of multiple layers of autoencoders, in which the outputs of each layer is fed as the inputs of successive layer ('Stacked Autoencoders', n.d.). Usually greedy layer-wise training is applied to train a stacked autoencoder. In current implementation, training for the stacked model is similar to the non-stacked one. Lleast-square loss function with is selected to train the stacked model, in which the difference is directly computed between input layer and final target layer. Although that setting of the hidden layer configure is heuristic, the combination in Suka et al. (2016)'s paper is followed.

After training, only the first part of the DAE (i.e. from input layer to bottleneck hidden layer) is used to transform each $\mathbf{f}_t \in \mathbb{R}^R$ into $\mathbf{x}_t \in \mathbb{R}^r$, where $r < R$. As a result, the encoded representation of a scan becomes

$$X = [\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_T]$$

to fed into the classification model to come.

High-level RNN classifiers and their ensemble for subject diagnosis

Wee, Yang, Yap, and Shen (2015) proposed a framework for brain functional connectivity analysis, in which each time series of each scan are decomposed into multiple overlapping sub-series by a sliding window. Justified by their work, a encoded time-series is splitted into identical-sized sub-series

$$X = [\mathbf{x}_1, \cdots, \mathbf{x}_s, \cdots, \mathbf{x}_S], \text{ where } T = n * S, n \text{ is an integer.}$$

A RNN is a specialized class of neural network that is suitable for dynamic temporal sequences. Connections between units in a RNN form a directed cycle. For each unit, hidden nodes are created as internal memory to process sequences of inputs, which enables RNN to condition the model on all previous units in a sequence (Figure 3). Below are the formulas in the network.

$$S_k = \sigma(W^{(rec)} S_{k-1} + W^x x_k)$$

$$y = softmax(W^{(s)} S_k)$$

The detailed notation is explained below:

– $\mathbf{x}_k \in \mathbb{R}^r, k = 1, \cdots, n$: the input of a unit

– S_k: current hidden state

– $W^{(rec)}$: weights matrix used to condition the hidden state of the previous time-step

– W^x: weights matrix used to condition the input of a unit $\mathbf{x}_k$

– $\sigma()$: the non-linearty function ()

– y: predicted class for the sequence

– $W^{(s)}$: weight matrix that transform S_k to y

In terms of the sub-series setting, data in each time point of a series $\mathbf{x}_t$ become the input of a unit. The final output at the end of the sequence is the predicted class (AD or NC) for the whole sub-series. For the purpose of diagnosis of a subject, ensemble learning is used to predict the class for the time-series of a scan. Each high-level classifier votes with equal weight and the class with the most votes is selected as the class of a time series.

Study 1

Study 2

Implementation of this Project

References

Biswal, B., Yetkin, F. Z., Haughton, V. M., & Hyde, J. (1995). Functional connectivity in the motor cortex of resting human brain using echo-planar mri. *Magnetic resonance in medicine*, *34*, 537–541.

Hutchison, R. M. (2013). Dynamic functional connectivity: promise, issues, and interpretations. *Neuroimage*, *80*, 360–378.

Li, Y., Wang, Y., Wu, G., Shi, F., Zhou, L., Lin, W., & Shen, D. (2012). Reasoning with exceptions: an event-related brain potentials study. *Neurobiology of aging*, *33*(2), 427.e15–427.e30.

Stacked Autoencoders. (n.d.). Retrieved from http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

Suka, H.-I., Weeb, C.-Y., Leea, S.-W., & Shen, D. (2016). State-space model with deep learning for functional dynamics estimation in resting-state fmri. *Neuroimage*, *129*, 292–307.

Wee, C.-Y., Yang, S., Yap, P.-T., & Shen, D. (2015). Temporally dynamic resting-state functional connectivity networks for early mci identification. *Brain Imaging and Behavior*, 1–15.

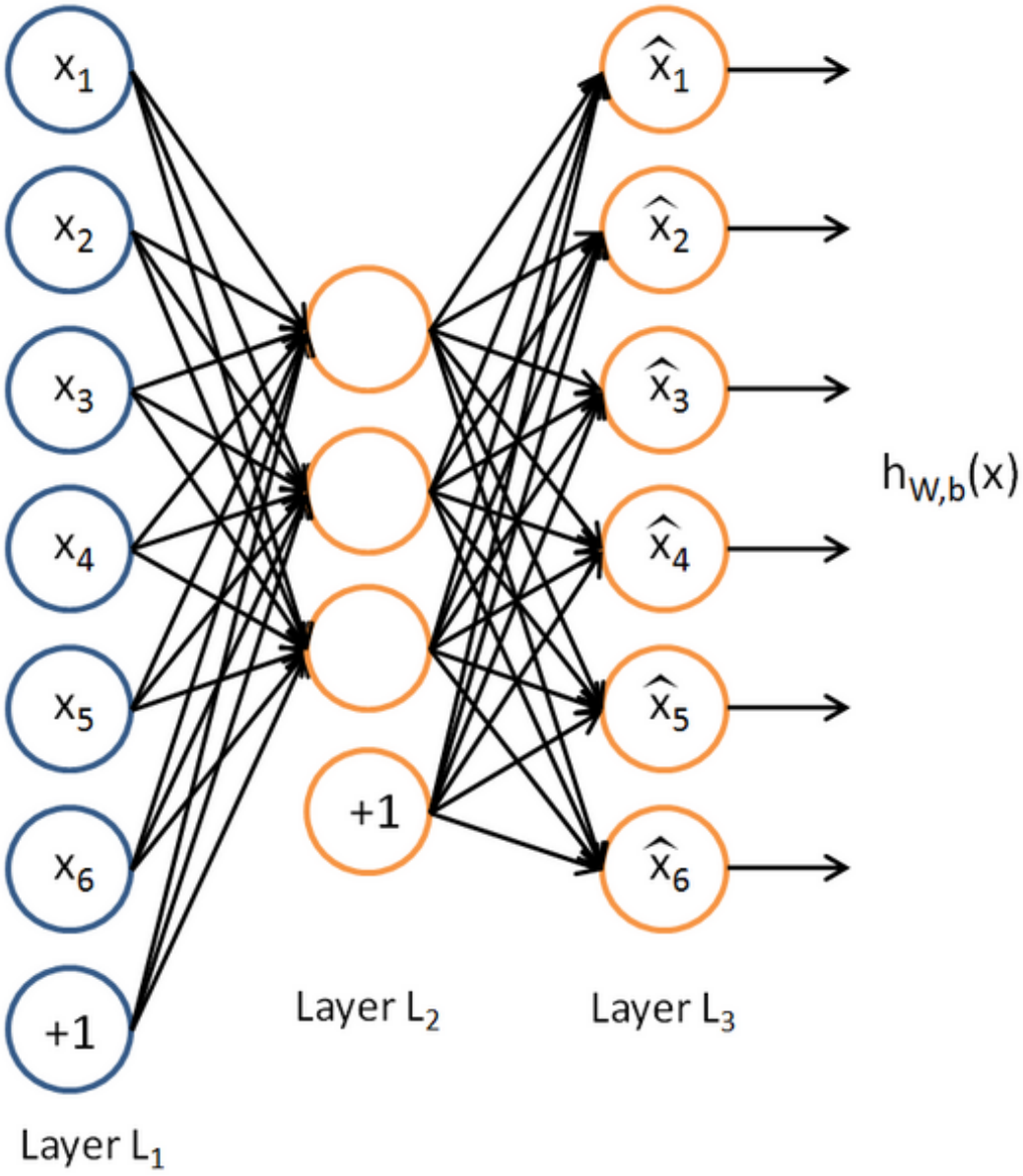

Figure 1. Autoencoder

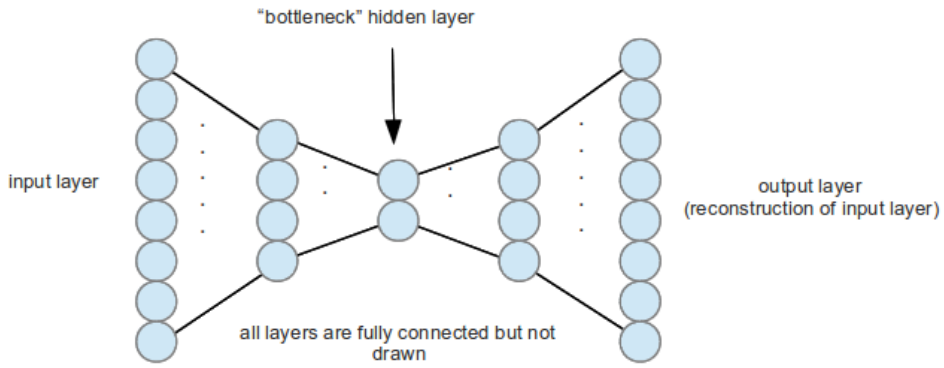

Figure 2. Stacked Autoencoder

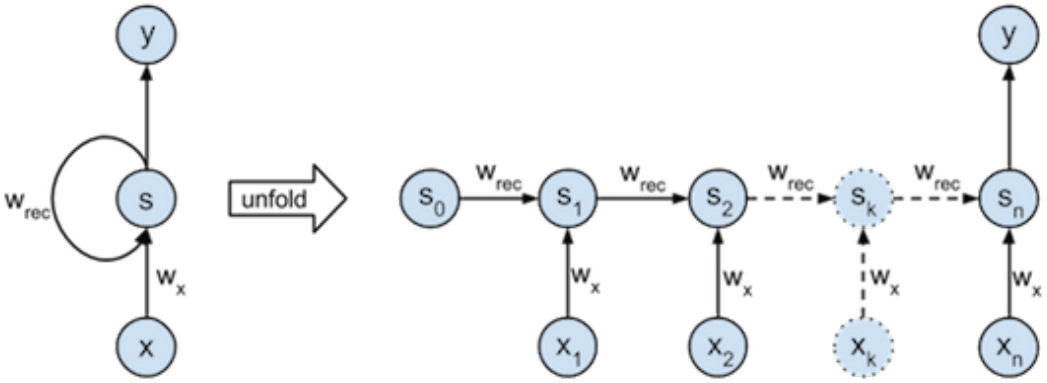

Figure 3. Recurrent Neural Networks

Appendix A

Visual vocabulary of an Elementary System Net

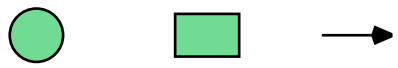

Figure A1. A circle depicts a place, a rectangle depicts a transition and an arrow depicts and arc. The arrow connects places with transitions and can be bent for that purpose. The green color is the standard color of the

Appendix B

Random Petri Net Generation Algorithm

```
Input
  n: number of graph elements
  f: number of interconnections
Output
  P: set of places
  T: set of transitions
  F: set of f el P x T U T x P


Algorithm
  tmp := gaussian(location=n)
  while tmp < 0:
    tmp := gaussian(location=n)
  sizeP := n - tmp
  sizeT := n - sizeP
  P := {p_1, p_2,..., p_sizeP}
  T := {t_1, t_2,..., t_sizeT}


  P_2 = {}
  T_2 = {}
  p_last := random element from P
  t_last := random element from T
  while f > 0:
    p_Pool := P \ P_2 if (P \ P_2) != {} else P
    t_Pool := T \ T_2 if (T \ T_2) != {} else T
    p := random(p_alt, random element from p_Pool)
    t := random(t_alt, random element from t_Pool)
    P_2 := P_2 U p
```

```
T_2 := T_2 U t
if random(TRUE, FALSE):
  F := F U (p, t)
  p_last := random element from P_2
  t_last := t
else:
  F := F U (t, p)
  p_last := p
  t_last := random element from T_2
f--
if places or transitions are not yet connected:
  discard Petri net
```

Appendix C

Layout Algorithm

```
Input
  P: set of places
  T: set of transitions
  F: set of f el P x T U T x P
Output
  Layout


Algorithm
  Draw grid
  Search for whether there is p el P with no incoming arc
  If yes: P_2 := all p el P with no incoming arc
  Else: P_2 := {random p el P}
  i := 0
  while F_2 != F:
    draw all p el P_2 on the ith grid line vertically arranged if not yet present
    draw all t for which (p, t) el F on the (i+1)th grid line if not yet present
    draw the arcs between p and t for all drawn t if not yet present
    P_2 := {p | (t, p) el F for all drawn t}
    F_2 := F_2 U (t, p) U (t, p) for t, p part of the drawing
    i++
```