QBUS64840 Predictive Analytics

Forecasting with Neural Networks and Deep Learning

University of Sydney Business School

Recommended reading

These lecture slides are comprehensive enough for your study. Optional readings include

- Online textbook Section 9.1 and 9.3: introduces (very briefly) some concepts in neural networks.
- A comprehensive book is *Deep Learning* by Goodfellow, Bengio and Courville, freely available at `https://www.deeplearningbook.org`

Table of contents

Learning objectives

- Understand the importance of data representation in data analysis, and that neural network modeling and deep learning are efficient data representation tools
- Understand some basic concepts of neural network (NN) and deep learning (DL)

Introduction

Introduction

- In regression modelling, sometimes it is advisable to add interaction terms $X_i \times X_j$ or quadratic terms X_i^2 to the model.
- These terms are examples of non-linear effects: when appropriate non-linear effect terms are added into the regression/classification model, the prediction accuracy is better
- How to select non-linear effect terms? When should they be added?
- Sometimes, this can be done manually, but requires domain-knowledge, trial and error: not efficient and not always possible!

Introduction

- In regression modelling, sometimes it is advisable to add interaction terms $X_i \times X_j$ or quadratic terms X_i^2 to the model.
- These terms are examples of non-linear effects: when appropriate non-linear effect terms are added into the regression/classification model, the prediction accuracy is better
- How to select non-linear effect terms? When should they be added?
- Sometimes, this can be done manually, but requires domain-knowledge, trial and error: not efficient and not always possible!

Introduction

A simple example

	A	B	C	D	E	F	G	H	I	J	K	L
1	Children	Catalogs	Salary	Gender_b	Married_b	Location_l	Ownhome	Age_y	Age_m	Hist_m	Hist_h	AmountSpent
2	0	6	47500	1	0	0	1	0	0	0	1	755
3	0	6	63600	0	0	1	0	0	1	0	1	1318
4	0	18	13500	1	0	1	0	1	0	0	0	296
5	1	18	85600	0	1	1	1	0	1	0	1	2436
6	0	12	68400	1	0	1	1	0	1	0	1	1304
7	0	6	30400	0	1	1	1	1	0	0	0	495
8	0	12	48100	1	0	1	0	0	1	1	0	782
9	0	18	68400	0	0	1	1	0	1	0	1	1155
10	3	6	51900	1	1	1	1	0	1	0	0	158
11	0	18	80700	0	1	0	1	0	0	0	0	3034
12	1	12	43700	0	1	1	0	1	0	0	0	927
13	3	18	111800	0	1	0	1	0	1	0	1	2065
14	1	24	44100	1	1	1	1	0	1	1	0	704
15	0	12	111400	0	1	1	1	0	1	0	1	2136
16	0	24	110000	1	1	0	1	0	0	0	1	5564
17	1	12	83100	1	1	0	1	0	1	0	0	2766

- Let's look at the Direct Marketing dataset (provided on Canvas)
- There are totally 11 covariates. The response is AmountSpent
- Let's use the first 900 observations as training data, the rest 100 as test data (in practice, data should be shuffled first)

Introduction

A simple example

	A	B	C	D	E	F	G	H	I	J	K	L
1	Children	Catalogs	Salary	Gender_b	Married_t	Location_l	Ownhome	Age_y	Age_m	Hist_m	Hist_h	AmountSpent
2	0	6	47500	1	0	0	1	0	0	0	1	755
3	0	6	63600	0	0	1	0	0	1	0	1	1318
4	0	18	13500	1	0	1	0	1	0	0	0	296
5	1	18	85600	0	1	1	1	0	1	0	1	2436
6	0	12	68400	1	0	1	1	0	1	0	1	1304
7	0	6	30400	0	1	1	1	1	0	0	0	495
8	0	12	48100	1	0	1	0	0	1	1	0	782
9	0	18	68400	0	0	1	1	0	1	0	1	1155
10	3	6	51900	1	1	1	1	0	1	0	0	158
11	0	18	80700	0	1	0	1	0	0	0	0	3034
12	1	12	43700	0	1	1	0	1	0	0	0	927
13	3	18	111800	0	1	0	1	0	1	0	1	2065
14	1	24	44100	1	1	1	1	0	1	1	0	704
15	0	12	111400	0	1	1	1	0	1	0	1	2136
16	0	24	110000	1	1	0	1	0	0	0	1	5564
17	1	12	83100	1	1	0	1	0	1	0	0	2766

The MSE of the prediction on the test data D_{test} is defined as

$$MSE = \frac{1}{n_{\text{test}}} \sum_{y_i \in D_{\text{test}}} (\widehat{y}_i - y_i)^2$$

To ease comparison, let's use the square root $RMSE = \sqrt{MSE}$, to get back to the original scale ($).

First, try the full linear regression model

```
import numpy as np
import pandas as pd
DM = pd.read_csv('DirectMarketing.csv')
import statsmodels.formula.api as smf
n = 900; n_test = 1000 - n;
lm = smf.ols('AmountSpent~Children + Catalogs + Salary + Gender_b + Married_b + Location_b + Ownhome_b \
            + Age_y + Age_m + Hist_m + Hist_h',DM.head(n)).fit()
predictions = lm.predict(DM.tail(n_test))
DM = pd.DataFrame.as_matrix(DM); DM = DM.astype(float); y_test = DM[n:1001,11]
MSE_lm = np.mean((predictions-y_test)**2)
print('Root of MSE on the test data for linear regression: ',np.sqrt(MSE_lm))

Root of MSE on the test data for linear regression:  604.499026646
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            AmountSpent   R-squared:                       0.740
Model:                            OLS   Adj. R-squared:                  0.736
Method:                 Least Squares   F-statistic:                     229.3
Date:                Mon, 18 Sep 2017   Prob (F-statistic):          1.49e-250
Time:                        12:56:29   Log-Likelihood:                -6840.2
No. Observations:                 900   AIC:                         1.370e+04
Df Residuals:                     888   BIC:                         1.376e+04
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     16.7689     79.692      0.210      0.833    -139.639     173.176
Children    -192.2424     18.272    -10.521      0.000    -228.103    -156.382
Catalogs      42.1146      2.597     16.217      0.000      37.018      47.212
Salary         0.0209      0.001     19.470      0.000       0.019       0.023
Gender_b      21.8884     34.741      0.630      0.529     -46.296      90.073
Married_b    -35.2676     47.141     -0.748      0.455    -127.789      57.254
Location_b  -470.7842     37.810    -12.451      0.000    -544.992    -396.576
Ownhome_b     28.8854     38.724      0.746      0.456     -47.115     104.886
Age_y        -31.8724     57.027     -0.559      0.576    -143.795      80.050
Age_m        -45.4188     50.817     -0.894      0.372    -145.154      54.317
Hist_m      -296.0326     44.673     -6.627      0.000    -383.709    -208.356
Hist_h        41.2316     53.742      0.767      0.443     -64.244     146.707
==============================================================================
```

A better linear regression model

```
DM = pd.read_csv('DirectMarketing.csv')
lm = smf.ols('AmountSpent~Children + Catalogs + Salary + Children*Salary+ Location_b + Hist_m ',DM.head(n
lm.summary()
predictions = lm.predict(DM.tail(n_test))
DM = pd.DataFrame.as_matrix(DM)
DM = DM.astype(float)
y_test = DM[n:1001,11]
MSE_lm = np.mean((predictions-y_test)**2)
print('Root of MSE on the test data for linear regression: ',np.sqrt(MSE_lm))
```

```
Root of MSE on the test data for linear regression:  584.887682063
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            AmountSpent   R-squared:                       0.753
Model:                            OLS   Adj. R-squared:                  0.751
Method:                 Least Squares   F-statistic:                     453.6
Date:                Mon, 18 Sep 2017   Prob (F-statistic):          4.39e-267
Time:                        13:15:14   Log-Likelihood:                -6816.5
No. Observations:                 900   AIC:                         1.365e+04
Df Residuals:                     893   BIC:                         1.368e+04
Df Model:                           6
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept        -189.5696     63.208     -2.999      0.003    -313.623     -65.517
Children            2.0501     32.178      0.064      0.949     -61.103      65.204
Catalogs           42.0576      2.475     16.993      0.000      37.200      46.915
Salary              0.0245      0.001     33.133      0.000       0.023       0.026
Children:Salary    -0.0036      0.000     -7.209      0.000      -0.005      -0.003
Location_b       -482.4083     35.589    -13.555      0.000    -552.255    -412.561
Hist_m           -262.2361     39.793     -6.590      0.000    -340.334    -184.138
==============================================================================
Omnibus:                      218.337   Durbin-Watson:                   1.979
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              750.945
Skew:                           1.142   Prob(JB):                    8.60e-164
Kurtosis:                       6.848   Cond. No.                     4.55e+05
==============================================================================
```

Now use a neural network model

```
np.random.seed(1000) # fix random seed
# import data
DM = pd.read_csv('DirectMarketing.csv')
DM = pd.DataFrame.as_matrix(DM); DM = DM.astype(float)
DM_test = DM[n:1001,:]; DM_train = DM[0:n,:]
X_train = DM_train[:,0:11]; y_train = DM_train[:,11]
X_test = DM_test[:,0:11]; y_test = DM_test[:,11]

# standardize the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
# apply same transformation to test data
X_test = scaler.transform(X_test)

# now buid the neural net model
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(11, input_dim=11, activation='relu')) # the first hidden layer has 11 units, input has 11
model.add(Dense(11, activation='relu')) # add another hidden layer with 11 units
model.add(Dense(1, activation='linear')) # the output layer has 1 unit with the linear activation

# Compile model
model.compile(loss='MSE', optimizer='adam')
# Fit the model
model.fit(X_train, y_train, epochs=100, batch_size=10)
# evaluate the model
MSE_nn = model.evaluate(X_test, y_test)
print('\n Root of MSE on the test data for neural net: ', np.sqrt(MSE_nn))

Root of MSE on the test data for neural net:  502.117223614
```

So for this dataset, which model is better in terms of prediction accuracy?

Introduction

- Neural networks and deep neural networks (called deep learning) have become an exciting research and application area in the last few years
- Deep learning is widely known for its high prediction accuracy
- It has been successfully applied to many large-scale industry problems, image recognition, language processing
- Its secret is Data Representation Learning

Introduction

- Neural networks and deep neural networks (called deep learning) have become an exciting research and application area in the last few years
- Deep learning is widely known for its high prediction accuracy
- It has been successfully applied to many large-scale industry problems, image recognition, language processing
- Its secret is Data Representation Learning

Introduction: Representation Learning

β or x?

Introduction: Representation Learning

- We want to predict a response Y, based on *raw/original* covariates $X = (X_1, ..., X_p)$, using linear regression modelling
- Usually, before doing regression modelling, some appropriate transformation of the covariates X_i is needed: $Z_1 = \phi_1(X)$, ..., $Z_d = \phi_d(X)$.
- The Z_i are called predictors or features.
- Then we model

$$\mathbb{E}(Y|X) = \beta_0 + \beta_1 Z_1 + ... + \beta_d Z_d$$

- Selection of the transformations $\phi_i(X)$ is an art!
- $Z = (Z_1, ..., Z_d)$ is a representation of $X = (X_1, ..., X_p)$. A better representation (in terms of predicting Y) leads to a better prediction accurary

Introduction: Representation Learning

- We want to predict a response Y, based on *raw/original* covariates $X = (X_1, ..., X_p)$, using linear regression modelling
- Usually, before doing regression modelling, some appropriate transformation of the covariates X_i is needed: $Z_1 = \phi_1(X)$, ..., $Z_d = \phi_d(X)$.
- The Z_i are called predictors or features.
- Then we model

$$\mathbb{E}(Y|X) = \beta_0 + \beta_1 Z_1 + ... + \beta_d Z_d$$

- Selection of the transformations $\phi_i(X)$ is an art!
- $Z = (Z_1, ..., Z_d)$ is a representation of $X = (X_1, ..., X_p)$. A better representation (in terms of predicting Y) leads to a better prediction accurary

Introduction: Representation Learning

- We want to predict a response Y, based on *raw/original* covariates $X = (X_1, ..., X_p)$, using linear regression modelling
- Usually, before doing regression modelling, some appropriate transformation of the covariates X_i is needed: $Z_1 = \phi_1(X)$, ..., $Z_d = \phi_d(X)$.
- The Z_i are called predictors or features.
- Then we model

$$\mathbb{E}(Y|X) = \beta_0 + \beta_1 Z_1 + ... + \beta_d Z_d$$

- Selection of the transformations $\phi_i(X)$ is an art!
- $Z = (Z_1, ..., Z_d)$ is a representation of $X = (X_1, ..., X_p)$. A better representation (in terms of predicting Y) leads to a better prediction accurary

Introduction: Representation Learning

- We want to predict a response Y, based on *raw/original* covariates $X = (X_1, ..., X_p)$, using linear regression modelling
- Usually, before doing regression modelling, some appropriate transformation of the covariates X_i is needed: $Z_1 = \phi_1(X)$, ..., $Z_d = \phi_d(X)$.
- The Z_i are called predictors or features.
- Then we model

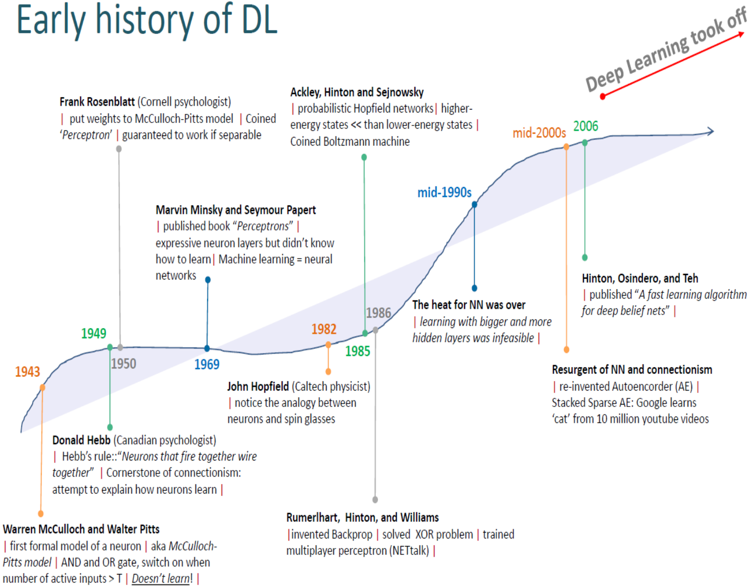
$$\mathbb{E}(Y|X) = \beta_0 + \beta_1 Z_1 + ... + \beta_d Z_d$$

- Selection of the transformations $\phi_i(X)$ is an art!
- $Z = (Z_1, ..., Z_d)$ is a representation of $X = (X_1, ..., X_p)$. A better representation (in terms of predicting Y) leads to a better prediction accurary

Introduction: Representation Learning

Neural network modeling is a representation learning method. It provides an efficient way to design a represenation $Z = \phi(X)$ that is effective for predicting the response Y.

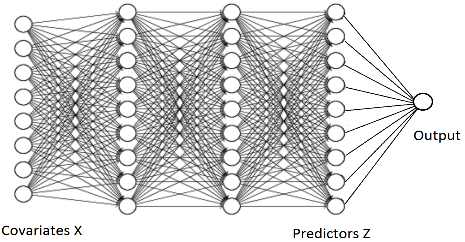
Early history of DL



[Image credit: Ding Phung]

What are neural networks?

They are a set of very flexible non-linear methods for regression/classification and other tasks.



A neural network, also called artificial neural network (ANN) is a computational model that is inspired by the network of neurons in the human brain

What are neural networks?

- A neural network is an interconnected assembly of simple processing units or neurons, which communicate by sending signals to each other over weighted connections
- A neural network is made of layers of similar neurons: an input layer, (one or many) hidden layers, and an output layer.
- The input layer receives data from outside the network. The output layer sends data out of the network. Hidden layers receive/process/send data within the network.
- A neural network is said to be deep, if it has many hidden layers. Deep neural network modelling is collectively refered to as deep learning.

What are neural networks?

- A neural network is an interconnected assembly of simple processing units or neurons, which communicate by sending signals to each other over weighted connections
- A neural network is made of layers of similar neurons: an input layer, (one or many) hidden layers, and an output layer.
- The input layer receives data from outside the network. The output layer sends data out of the network. Hidden layers receive/process/send data within the network.
- A neural network is said to be deep, if it has many hidden layers. Deep neural network modelling is collectively refered to as deep learning.

What are neural networks?

- A neural network is an interconnected assembly of simple processing units or neurons, which communicate by sending signals to each other over weighted connections
- A neural network is made of layers of similar neurons: an input layer, (one or many) hidden layers, and an output layer.
- The input layer receives data from outside the network. The output layer sends data out of the network. Hidden layers receive/process/send data within the network.
- A neural network is said to be deep, if it has many hidden layers. Deep neural network modelling is collectively refered to as deep learning.

What are neural networks?

- A neural network is an interconnected assembly of simple processing units or neurons, which communicate by sending signals to each other over weighted connections
- A neural network is made of layers of similar neurons: an input layer, (one or many) hidden layers, and an output layer.
- The input layer receives data from outside the network. The output layer sends data out of the network. Hidden layers receive/process/send data within the network.
- A neural network is said to be deep, if it has many hidden layers. Deep neural network modelling is collectively refered to as deep learning.

What are neural networks?

- In a nutshell, a neural net is a multivariate function: output η is a function of the inputs $X = (X_1, ..., X_p)^\top$

$$\eta = f(X_1, ..., X_p)$$

- More precisely, this function is a layered composite function

$$\begin{aligned} Z_1 &= f_1(X) \\ Z_2 &= f_2(Z_1) \\ &\ldots \\ Z_L &= f_L(Z_{L-1}) \\ \eta &= f_{L+1}(Z_L) \end{aligned}$$

What are neural networks?

A neural network provides a mechanism for functional approximation

- Suppose that $f_{\text{true}}(X)$ is a true, yet unknown, function that we want to estimate. E.g.,
  - $f_{\text{true}}(X) = \mathbb{E}(Y|X)$: the conditional mean of a response Y given X
- A neural net with the output $\eta = f(X)$ provides an approximation of $f_{\text{true}}(X)$, i.e. we use $f(X)$ to approximate $f_{\text{true}}(X)$.

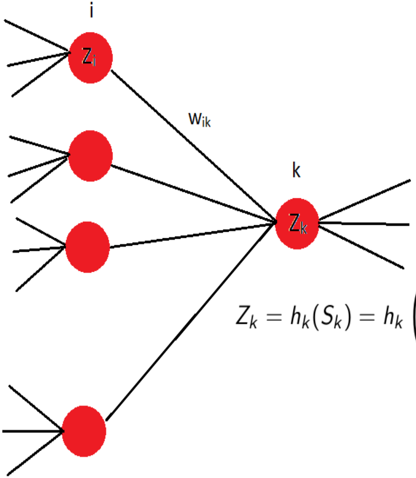
Note

There are several variants of neural networks:

- The network structure considered so far is often called feed-forward neural networks, which are most suitable for cross-sectional data. Can be used for time series data too.
- In the next lecture, you will study recurrent neural networks, which are most suitable for time series data.

Fundamental concepts

Elements of a neural network



$$Z_k = h_k(S_k) = h_k\left(\sum_i w_{ik} Z_i + w_{0k}\right)$$

Elements of a neural network

A (feedforward) neural net includes

- a set of processing units (also called neurons, nodes)
- weights w_{ik}, which are connection strengths from unit i to unit k
- a propagation rule that determines the total input S_k of unit k, from the units that send information to unit k
- the output Z_k for each unit k, which is a function of the input S_k
- an activation function h_k that determines the output Z_k based on the input S_k, $Z_k = h_k(S_k)$

Elements of a neural network

A (feedforward) neural net includes

- a set of processing units (also called neurons, nodes)
- weights w_{ik}, which are connection strengths from unit i to unit k
- a propagation rule that determines the total input S_k of unit k, from the units that send information to unit k
- the output Z_k for each unit k, which is a function of the input S_k
- an activation function h_k that determines the output Z_k based on the input S_k, $Z_k = h_k(S_k)$

Elements of a neural network

A (feedforward) neural net includes

- a set of processing units (also called neurons, nodes)
- weights w_{ik}, which are connection strengths from unit i to unit k
- a propagation rule that determines the total input S_k of unit k, from the units that send information to unit k
- the output Z_k for each unit k, which is a function of the input S_k
- an activation function h_k that determines the output Z_k based on the input S_k, $Z_k = h_k(S_k)$

Elements of a neural network

A (feedforward) neural net includes

- a set of processing units (also called neurons, nodes)
- weights w_{ik}, which are connection strengths from unit i to unit k
- a propagation rule that determines the total input S_k of unit k, from the units that send information to unit k
- the output Z_k for each unit k, which is a function of the input S_k
- an activation function h_k that determines the output Z_k based on the input S_k, $Z_k = h_k(S_k)$

Elements of a neural network

A (feedforward) neural net includes

- a set of processing units (also called neurons, nodes)
- weights w_{ik}, which are connection strengths from unit i to unit k
- a propagation rule that determines the total input S_k of unit k, from the units that send information to unit k
- the output Z_k for each unit k, which is a function of the input S_k
- an activation function h_k that determines the output Z_k based on the input S_k, $Z_k = h_k(S_k)$

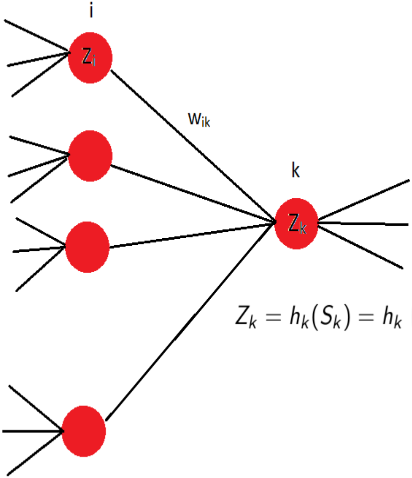
Elements of a neural network

It's useful to distinguish three types of units:

- input units (often denoted by X): receive data from outside the network
- hidden units (often denoted by Z): receive data from and send data to units within the network.
- output units: send data out of the network. The type of the output depends on the task (regression, binary classification or multinomial regression). In many cases, there is only one scalar output unit.

Given the signal from a set of inputs X, an NN produces an output.

Elements of a neural network



$$Z_k = h_k(S_k) = h_k\left(\sum_i w_{ik} Z_i + w_{0k}\right)$$

Elements of a neural network

The total input sent to unit k is

$$S_k = \sum_i w_{ik} Z_i + w_{0k}$$

which is a weighted sum of the outputs from all units i that are connected to unit k, plus a bias/intercept term w_{0k}.

Then, the output of unit k is

$$Z_k = h_k(S_k) = h_k\left(\sum_i w_{ik} Z_i + w_{0k}\right)$$

Usually, we use the same activation function $h_k = h$ for all units.

Elements of a neural network

The total input sent to unit k is

$$S_k = \sum_i w_{ik} Z_i + w_{0k}$$

which is a weighted sum of the outputs from all units i that are connected to unit k, plus a bias/intercept term w_{0k}.

Then, the output of unit k is

$$Z_k = h_k(S_k) = h_k\left(\sum_i w_{ik} Z_i + w_{0k}\right)$$

Usually, we use the same activation function $h_k = h$ for all units.

Elements of a neural network

The total input sent to unit k is

$$S_k = \sum_i w_{ik} Z_i + w_{0k}$$

which is a weighted sum of the outputs from all units i that are connected to unit k, plus a bias/intercept term w_{0k}.

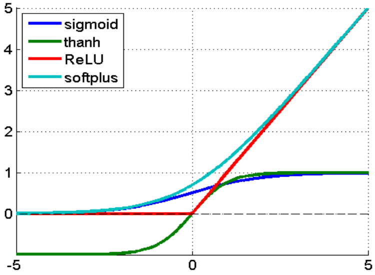
Then, the output of unit k is

$$Z_k = h_k(S_k) = h_k\left(\sum_i w_{ik} Z_i + w_{0k}\right)$$

Usually, we use the same activation function $h_k = h$ for all units.

Elements of a neural network

Popular activation functions:



Sigmoid activation function: $h(S) = \frac{1}{1+e^{-S}}$

Tang activation function: $h(S) = \frac{e^{S}-e^{-S}}{e^{S}+e^{-S}}$

Rectified activation function : $h(S) = \max(0, S) = \begin{cases} S, & S > 0 \\ 0, & S \leq 0 \end{cases}$

Neural Net as a Data Representation Learning tool

- We want to predict a response Y, based on p *raw* covariates $X = (X_1, ..., X_p)'$
- We want to represent X by d predictors/features $Z = (Z_1, ..., Z_d)' = \phi(X)$, before predicting Y based on Z.
- Neural network modelling is a data representation learning method, that transforms X into

$$Z = \phi(X) = \phi(X, w)$$

with the hope that predicting Y using the linear regression/classification techniques based on Z is more accurate than based on X directly.
The idea is that we tune/train w to achieve this goal.

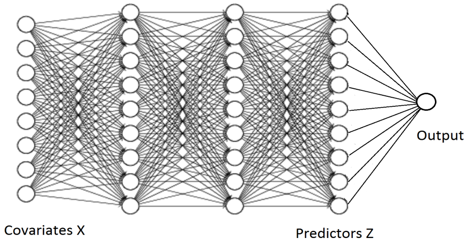
Neural Net as a Data Representation Learning tool

- We want to predict a response Y, based on p *raw* covariates $X = (X_1, ..., X_p)'$
- We want to represent X by d predictors/features $Z = (Z_1, ..., Z_d)' = \phi(X)$, before predicting Y based on Z.
- Neural network modelling is a data representation learning method, that transforms X into

$$Z = \phi(X) = \phi(X, w)$$

  with the hope that predicting Y using the linear regression/classification techniques based on Z is more accurate than based on X directly.
  The idea is that we tune/train w to achieve this goal.

Neural Net as a Data Representation Learning tool



Graphical representation of a neural net with $L = 3$ hidden layers. The input layer represents the raw covariates X. The last hidden layer (hidden layer 3) represents the predictors Z.

Neural Net as a Representation Learning tool

Denote the final output of the neural net as

$$\eta = \beta_0 + \beta_1 Z_1 + ... + \beta_d Z_d$$

with $\beta = (\beta_0, ..., \beta_d)'$.

Note that η is a function of X and depends on w and β

$$\eta = \eta(X, w, \beta)$$

w is the set of weights that connect covariates X to predictors Z, and β is the set of weights that connect Z to η.

We will use $\eta(X, w, \beta)$ to approximate $f_{\text{true}}(X)$.

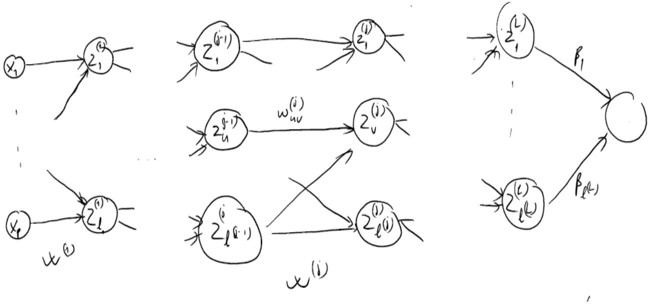
Forward propagation algorithm*

Slides with * are highly technical. You're encouraged to go through them, but these are not tested in the exams.

Forward propagation algorithm for computing the output

- Consider a neural net with the structure $(p, \ell^{(1)}, ..., \ell^{(L)}, 1)$
  - The input layer has p covariates $X_1, ..., X_p$.
  - L hidden layers: the first hidden layer has $\ell^{(1)}$ units, the second hidden layer has $\ell^{(2)}$, etc.
  - The last layer is a single output η

Forward propagation algorithm*



- Let $w_{uv}^{(j)}$ be the weight from unit u in the previous layer $j-1$ to unit v in layer j. Layer $j=0$ is the input layer, $\ell^{(0)} := p$.
- The total input to unit v of layer j is

$$S_v^{(j)} = w_{0v}^{(j)} + \sum_{u=1}^{\ell^{(j-1)}} w_{uv}^{(j)} Z_u^{(j-1)} = w_v^{(j)\prime} Z^{(j-1)}$$

Its output is $Z_v^{(j)} = h(S_v^{(j)})$.

Forward propagation algorithm*

$$w_v^{(j)} = \begin{pmatrix} w_{0,v}^{(j)} \\ w_{1,v}^{(j)} \\ \dots \\ w_{\ell(j-1),v}^{(j)} \end{pmatrix} : \text{set of weights sends signal to unit } v \text{ of layer } j$$

$$S^{(j)} = \begin{pmatrix} S_1^{(j)} \\ \dots \\ S_{\ell(j)}^{(j)} \end{pmatrix} : \text{vector of total inputs to layer } j,\ j = 1, \dots, L.$$

$$Z^{(j)} = \begin{pmatrix} 1 \\ Z_1^{(j)} \\ \dots \\ Z_{\ell(j)}^{(j)} \end{pmatrix} : \text{vector of outputs from layer } j,\ Z^{(0)} := \begin{pmatrix} 1 \\ X_1 \\ \dots \\ X_p \end{pmatrix}$$

Forward propagation algorithm*

- $$W^{(j)} = \begin{pmatrix} w_{01}^{(j)} & w_{11}^{(j)} & \dots & w_{\ell(j-1),1}^{(j)} \\ w_{02}^{(j)} & w_{12}^{(j)} & \dots & w_{\ell(j-1),2}^{(j)} \\ \dots & \dots & \dots & \\ w_{0,\ell(j)}^{(j)} & w_{1,\ell(j)}^{(j)} & \dots & w_{\ell(j-1),\ell(j)}^{(j)} \end{pmatrix} = \begin{pmatrix} w_1^{(j)\prime} \\ w_2^{(j)\prime} \\ \dots \\ w_{\ell(j)}^{(j)\prime} \end{pmatrix}$$

  be the matrix of all weights from layer $j-1$ to layer j.
- Then
  $$S^{(j)} = W^{(j)} Z^{(j-1)}$$
- The final output of the network is
  $$\eta = \beta_0 + \beta_1 Z_1^{(L)} + \dots + \beta_L Z_{\ell(L)}^{(L)} = \beta' Z^{(L)}.$$

Forward propagation algorithm*

Pseudo-code algorithm for computing the output.

Input: covariates $X_1, ..., X_p$ and weights $w = (W^{(1)}, ..., W^{(L)}), \beta$

Output: η

- $Z^{(0)} := (1, X_1, ..., X_p)'$
- For $i = 1, ..., L$:
  - $S^{(j)} = W^{(j)} Z^{(j-1)}$
  - $Z^{(j)} = \begin{pmatrix} 1 \\ h(S^{(j)}) \end{pmatrix}$
- $\eta = \beta' Z^{(L)}$.

Neural net for regression

Given a neural network, we now know how to compute its output η from an input vector X.

How is this output used for forecasting?

Neural net for forecasting

Suppose that the response Y is numerical.

The model is

$$\begin{aligned} Y &= \eta(X, w, \beta) + \epsilon \\ &= \beta_0 + \beta_1 Z_1 + \ldots + \beta_d Z_d + \epsilon \end{aligned}$$

where ϵ is an error term with mean 0 and variance σ^2. Often, we assume $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

The least squares method can now be used to estimate the model parameters $\theta = (w, \beta, \sigma^2)$.

Note on Python: In Python, the activation function of the output unit for regression is defined as the identity function, named linear.

Neural net for forecasting

Suppose that the response Y is numerical.

The model is

$$\begin{aligned} Y &= \eta(X, w, \beta) + \epsilon \\ &= \beta_0 + \beta_1 Z_1 + \ldots + \beta_d Z_d + \epsilon \end{aligned}$$

where ϵ is an error term with mean 0 and variance σ^2. Often, we assume $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

The least squares method can now be used to estimate the model parameters $\theta = (w, \beta, \sigma^2)$.

Note on Python: In Python, the activation function of the output unit for regression is defined as the identity function, named linear.

Training a neural net

- Given that a neural net model has been developed, given a dataset $\{y_i, x_i = (x_{i1}, ..., x_{ip})^\top\}$, $i = 1, ..., n$, the most difficult task is to estimate the model parameters θ
- Other problems in neural network modelling
  - How to select the number of hidden layers?
  - How to select the number of units in each hidden layer?
  - How to perform variable selection?
  - etc.

Next...

- We look at neural net for regression in detail
- How to train a neural net model
- How to use a neural net for prediction with cross-sectional data and time series data.