TRƯỜNG ĐẠI HỌC ĐIỆN LỰC

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO CHUYÊN ĐỀ**

**HỌC MÁY NÂNG CAO**

**ĐỀ TÀI**

**TÌM HIỂU COGNITIVE TOOLKIT FRAMEWORK VÀ ỨNG DỤNG GIẢI CÁC BÀI TOÁN HỒI QUY TUYẾN TÍNH VÀ HỒI QUY LOGISTIC**

**Sinh viên thực hiện : NGUYỄN HỒNG KỲ**

**: NGUYỄN QUỐC TUẤN**

**: TRỊNH THÀNH ĐẠT**

**Giảng viên hướng dẫn : PHẠM THỊ KIM DUNG**

**Lớp : D12 – CNPM2**

**Chuyên ngành : CÔNG NGHỆ PHẦN MỀM**

**Khoa : CÔNG NGHỆ THÔNG TIN**

**Niên Khóa : 2017 - 2022**

**Hà Nội, tháng 1 năm 2021**

**PHIẾU CHẤM ĐIỂM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và Tên sinh viên** | **Nội dung thực hiện** | **Điểm** | **Chữ ký** |
| 1 | Nguyễn Hồng Kỳ |  |  |  |
| 2 | Nguyễn Quốc Tuấn |  |  |  |
| 3 | Trịnh Thành Đạt |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Giảng viên chấm thi** | **Chữ ký** | **Ghi chú** |
| Giảng viên chấm thi 1: |  |  |
| Giảng viên chấm thi 2: |  |  |

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH 4](#_Toc61440864)

[DANH MỤC BẢNG 6](#_Toc61440865)

[LỜI CẢM ƠN 7](#_Toc61440866)

[LỜI MỞ ĐẦU 8](#_Toc61440867)

[CHƯƠNG 1: FRAMEWORK COGNITIVE TOOLKIT 9](#_Toc61440868)

[1.1. Giới thiệu framework cognitive toolkit 9](#_Toc61440869)

[1.2. Các tính năng của CNTK 10](#_Toc61440870)

[1.3. Cài đặt và ví dụ minh họa 11](#_Toc61440871)

[CHƯƠNG 2: GIẢI CÁC BÀI TOÁN HỌC MÁY 20](#_Toc61440872)

[2.1. Ứng dụng neural network của cntk để đánh giá điểm rèn luyện của học sinh dựa trên thuộc tính về lối sống. (Nguyễn Hồng Kỳ) 20](#_Toc61440873)

[2.2. Ứng dụng neural network để dự đoán mắc bệnh tim hay không dự trên các thông số liện quan đến bệnh tim. (Nguyễn Quốc Tuấn) 34](#_Toc61440874)

[2.3. Ứng dụng hồi quy tuyến tính để dự đoán hiệu năng máy tính dựa trên các thông số phần cứng. (Trịnh Thành Đạt) 45](#_Toc61440875)

[KẾT LUẬN 53](#_Toc61440876)

[DANH MỤC TÀI LIỆU THAM KHẢO 54](#_Toc61440877)

# DANH MỤC HÌNH ẢNH

Hình 1: Dữ liệu trước khi xử lý 15

Hình 2: Dữ liệu sau khi xử lý 16

Hình 3: Biểu thị dữ liệu sau khi giảm chiều 17

Hình 4: Quá trình huấn luyện mô hình 18

Hình 5: Đưa ra mô hình và đánh giá mô hình 19

Hình 6: Mô hình kiến trúc mạng ANN 22

Hình 7: Quá trình xử lý của ANN 23

Hình 8: Mô tả hàm kích hoạt trong neural 24

Hình 9: Biểu diễn hồi quy tuyến tính trrong mạng neural 26

Hình 10: Dữ liệu tiền xử lý (HK) 29

Hình 11: Dữ liệu sau khi xử lý (HK) 30

Hình 12: Biểu đồ tương quan giữa X và Y (HK) 31

Hình 13: Quá trình huấn luyện mô hình 32

Hình 14: Hệ số hồi quy và bias (HK) 33

Hình 15: So sánh kết quả Y test với Y dự đoán và MSE (HK) 33

Hình 16: Đồ thị các activation function khác nhau 36

Hình 17: Dữ liệu trước khi xử lý (QT) 41

Hình 18: Dữ liệu sau khi xử lý (QT) 42

Hình 19: Biểu đồ tương quan giữa X và Y (QT) 43

Hình 20: Quá trình huấn luyện mô hình 43

Hình 21: Mô hình đưa ra các hệ số và bias của boundary 44

Hình 22: Kết quả kiểm tra với dữ liệu test 44

Hình 23: Dữ liệu của bài toán (D) 50

Hình 23: Biểu đồ tương quan giữa X và Y (D) 51

Hình 24:Hệ số hồi quy và bias 51

Hình 25: Kiểm tra lại mô hình với dữ liệu test (D) 52

# DANH MỤC BẢNG

Bảng 1: So sánh hai phiên bản CNTK 09

Bảng 2: Bảng thuộc tính dữ liệu loài bào ngư 11

Bảng 3: Bảng thuộc tính dữ liệu đánh giá rèn luyện học sinh (HK) 20

Bảng 4: Bảng thuộc tính dữ liệu bệnh tim (QT) 33

Bảng 5: Bảng thuộc tính dữ liệu đánh giá hiệu năng máy tính (D) 44

# LỜI CẢM ƠN

Trong lời đầu tiên của báo cáo môn học “ Học máy nâng cao” , chúng em muốn gửi những lời cám ơn và biết ơn chân thành nhất của mình tới tất cả những người đã hỗ trợ, giúp đỡ chúng em về kiến thức và tinh thần trong quá trình thực hiện báo cáo.

Em xin chân thành gửi lời cảm ơn tới các thầy cô giáo trong Trường Đại học Điện Lực nói chung và các thầy cô giáo trong Khoa Công nghệ thông tin nói riêng đã tận tình giảng dạy, truyền đạt cho em những kiến thức cũng như kinh nghiệm quý báu trong suốt quá trình học tập.

Đặc biệt, em xin gửi lời cảm ơn đến Giáo viên hướng dẫn Lê Thị Trang Linh, giảng viên Khoa Công nghệ thông tin - Trường Đại học Điện Lực. Thầy đã tận tình theo sát giúp đỡ, trực tiếp chỉ bảo, hướng dẫn trong suốt quá trình nghiên cứu và học tập của chúng em. Trong thời gian học tập với thầy, nhóm em không những tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần làm việc, thái độ nghiên cứu khoa học nghiêm túc, hiệu quả. Đây là những điều rất cần thiết cho chúng em trong quá trình học tập và công tác sau này.

Do thời gian thực hiện có hạn kiến thức còn nhiều hạn chế nên bài làm của em chắc chắn không tránh khỏi những thiếu sót nhất định. Em rất mong nhận được ý kiến đóng góp của thầy cô giáo và các bạn để em có thêm kinh nghiệm và tiếp tục hoàn thiện đồ án của mình.

Em xin chân thành cảm ơn!

# LỜI MỞ ĐẦU

Công nghệ thông tin (CNTT) ngày càng có vai trò quan trọng trong cuộc sống hằng ngày của chúng ta. Việc ứng dụng CNTT vào các lĩnh vực trong đời sống giúp công việc được tiến hành nhanh chóng và hiệu quả hơn. Có rất nhiều công việc mới phát triển song song với sự phát triển của CNTT, một trong những số đó là phân tích khai phá dữ liệu, hướng đi mang lại hiệu quả rất lớn.

Nhóm chúng em chọn đề tài “Tìm hiểu cognitive toolkit framework và ứng dụng giải các bài toán hồi quy tuyến tính, logistic” nhằm tìm hiểu sâu hơn về framework cách khai thác và sử dụng hiệu quả, từ đó viết một ứng dụng cụ thể thử nghiệm làm cơ sở củng cố kiến thức và định hướng, kế hoạch xây dựng các ứng dụng trong tương lai.

# CHƯƠNG 1: FRAMEWORK COGNITIVE TOOLKIT

## 1.1. Giới thiệu framework cognitive toolkit

Cognitive toolkit viết tắt là (CNTK) là một framework học sâu (Deep learning) mã nguồn mở do Microsoft Search phát triển. CNTK được viết bằng C++ để tối ưu hiệu suất, framework này có hỗ trợ cho 3 ngôn ngữ là C++, C# và python và hỗ trợ hai nền tảng là Windows và Linux. CNTK sử dụng các mạng thần kinh nhân tạo (neural network) để xây dựng các thuật toán học máy, hơn nữa framework này có hỗ trợ tính toán bằng GPU của NVIDIA đối với những máy sử dụng card đồ họa NVIDIA hoặc hỗ trợ tính toán bằng CPU với những máy không có card đồ họa NVIDIA.

Đối với phiên bản CNTK dành cho CPU sử dụng Intel MKLML được tối ưu hóa, trong đó MKLML là tập con của MKL (Math kernel library) và được phát hành cùng với Intel MKL-DNN dưới dạng phiên bản cuối cùng của Intel MKL cho MKL-DNN.

Đối với phiên bản CNTK dành cho GPU – NVIDIA thì sử dụng các thư viện của NVIDIA được tối ưu hóa cao như CUB và cuDNN. Nó hỗ trợ huấn luyện phân tán trên nhiều GPU và nhiều máy để huấn luyện các mô hình học máy nhanh hơn. Phiên bản GPU-build bao gồm: MSR-developed 1bit-quantized SGD và thuật toán huấn luyện song song block-momentum SGD

Các phiên bản của CNTK (bao gồm cả CPU và GPU): CNTK 1.0, CNTK 2.0, CNTK 2.1, CNTK 2.2, CNTK 2.3, CNTK 2.4, CNTK 2.5, CNTK 2.6 và CNTK 2.0. Tuy nhiên ta chú ý tới 2 phiên bản CNTK chính là: CNTK 1.0 và CNTK 2.0. Dưới đây là bảng so sánh giữa hai phiên bản CNTK 1.0 và CNTK 2.0

|  |  |
| --- | --- |
| **Phiên bản 1.0** | **Phiên bản 2.0** |
| Được phát hành năm 2016 | Được phát hành tháng 6 năm 2017 |
| Được sử dụng ngôn ngữ kịch bản độc quyền BrainScript | Các hàm của framework có thể được gọi bằng C++, Python. Có thể dễ dàng tải các modul bằng C# hoặc Java. BrainScript cũng được hỗ trợ trong phiên bản 2.0 |
| Có thể chạy được trên hệ thống windows và linux, không hỗ trợ chạy trên macOS | Có thể chạy được trên hệ thống windows và linux, không hỗ trợ chạy trên macOS |

Bảng 1: Bảng so sánh hai phiên bản CNTK

## 1.2. Các tính năng của CNTK

1. **Các thành phần tích hợp**

* CNTK có các thành phần tích hợp được tối ưu hóa cao để có thể xử lý dữ liệu dày đặc hoặc thưa thớt đa chiều từ Python, C ++ hoặc BrainScript.
* Ta có thể triển khai các mô hình học máy như: CNN, FNN, RNN, Batch Normalization và Sequence-to-Sequence
* Nó cung cấp cho ta chức năng để thêm các thành phần chính mới do người dùng xác định trên GPU từ python
* Cung cấp khả năng điều chỉnh thông số tự động
* Ta có thể triển khai các mô hình học tăng cường, GAN, học có giám sát, không giám sát
* Tích hợp các trình đọc tối ưu hóa cho những bộ dữ liệu lớn

1. **Sử dụng tài nguyên hiệu quả**

* Đối với phiên bản CNTK cho GPU thì nó cung cấp cho chúng ta khả năng tính toán song song trên nhiều GPU hay những máy thông qua GSD 1bit
* Với các mô hình lớn sử dụng tính toán bẳng GPU, nó cung cấp khả năng chia sẻ bộ nhớ và các phương pháp tích hợp khác

1. **Dễ dàng triển khai mô hình**

* CNTK có đầy đủ các API để xác định network, learner, reader, trainer và đánh giá từ Python, C ++ và BrainScript.
* CNTK giúp ta có thể dễ dàng đánh giá các mô hình với Python, C ++, C # hoặc BrainScript.
* Cung cấp cả hight-level API và low-level API
* Cung cấp các thành phần khác nhau để kiểm định mô hình
* Tạo dữ liệu nhật ký từ mô hình và từ dữ liệu này giúp ta theo dõi quá trình huấn luyện mô hình

## 1.3. Cài đặt và ví dụ minh họa

CNTK hỗ trợ trên 3 ngôn ngữ chính là C++, C# và Python và 2 hệ thống là windows và linux. Ở trong báo cáo chuyên đề này ta sẽ triển khai CNTK trên windows và sử dụng ngôn ngữ python. CNTK hiện tại chỉ hỗ trợ python 3.6 trở xuống và anaconda 4.3.1 trở xuống

1. **Cài đặt**

Sau khi tải và cài đặt python, ta vào command line gõ: **pip install cntk** và chờ đợi hoàn tất. Đối với anaconda, từ anaconda prompt ta gõ lần lượt 3 dòng sau:

**conda create --name cntk-py35 python=3.5 numpy scipy h5py jupyter**

**activate cntk-py35**

**pip install https://cntk.ai/PythonWheel/CPU-Only/cntk-2.6-cp35-cp35m-win\_amd64.whl**

Sau đó ta cài đặt biến môi trường cho CNTK. Đối với phiên bản CNTK cho CPU ta cài đặt biến MKL, từ command line gõ: **setx PATH "c:\local\mklml-2018.0.3\lib;%PATH%"**. Với phiên bản CNTK cho GPU – NVIDIA ta cài biến cuDNN, từ command line gõ lần lượt 3 dòng:

**setx PATH "C:\Program Files\NVIDIA Corporation\NVSMI;%PATH%"**

**setx PATH "C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin;%PATH%"**

**setx PATH "C:\local\cudnn-9.0-v7.0\cuda\bin;%PATH%"**

Nếu quá 1024 ký tự trong biến môi trường ta có thể cài đặt thông qua registry theo đường dẫn sau:

**HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment**

Sau đó vào mục Path để cài biến môi trường

1. **Ví dụ minh họa**

Ở đây ta sẽ ví dụ với bài toán hồi quy tuyến tính đơn giản

* Dữ liệu đầu vào:

Dữ liệu đầu vào là một file csv chứa thông tin về loài bào ngư, qua thông tin đó ta dự đoán tuổi của chúng. Dữ liệu gồm có 9 cột và 4175 dòng. Trong dữ liệu có 1 cột dạng category nên ta phải xử lý dữ liệu trước khi dùng huấn luyện mô hình

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Ý nghĩa** | **Dữ liệu** |
| sex | Giới tính | M là con đực, F là con cái, I là con nhỏ |
| length | Chiều dài | Chiều dài vỏ ngoài (mm) |
| diameter | Đường kính | Đường kính vỏ (mm) |
| height | Chiều cao | Chiều cao vỏ (mm) |
| whole\_weight | Khối lương nguyên con | Khối lượng (gam) |
| shucked\_weight | Khối lượng thịt | Khối lượng (gam) |
| viscera\_weight | Khối lượng ruột | Khối lượng (gam) |
| shell\_weight | Khối lượng thịt sau khi làm khô | Khối lượng (gam) |
| old | Tuổi | Tuổi (năm) |

Bảng 2: Bảng thuộc tính dữ liệu của loài bào ngư

Ta sẽ lấy thuộc tính ‘old’ làm biến mục tiêu các thuộc tính còn lại làm biến giải thích

* Import các thư viện

**import** warnings  
warnings.filterwarnings(**'ignore'**)  
**import** cntk  
**import** pandas **as** pd  
**from** sklearn.model\_selection **import** train\_test\_split  
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
**from** cntk.ops **import** \*  
**from** cntk **import** input\_variable  
**from** cntk **import** learning\_rate\_schedule, UnitType  
**from** sklearn.decomposition **import** PCA  
**from** cntk.learners **import** sgd  
**from** cntk.train.trainer **import** Trainer  
**from** sklearn **import** preprocessing

* Đọc dữ liệu và xử lý dữ liệu

*# Đọc dữ liệu*data = pd.read\_csv(**"abalone.csv"**, sep=**","**)  
data.head()  
print(**"Dữ LIỆU TRƯỚC KHI XỬ LÝ"**)  
print(data)  
*# Xử lý dữ liệu*lbEncoder = preprocessing.LabelEncoder()  
data[**'sex\_replace'**] = lbEncoder.fit\_transform(data[**'sex'**])  
print(**"DỮ LIỆU SAU KHI CHUẨN HÓA"**)  
print(data)  
  
Y = data[**'old'**].values  
Y = Y[:, **None**]  
X = data.drop([**"sex"**, **"old"**], axis=1).values  
  
*# Chia dữ liệu train,test*X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2)

* Biểu diễn dữ liệu

*# Biểu diễn dữ liệu*pca = PCA(n\_components=1)  
X\_scat = pca.fit\_transform(X\_train)  
plt.scatter(X\_scat, Y\_train)  
plt.xlabel(**"X"**)  
plt.ylabel(**"Y"**)  
plt.title(**"Bieu do tuong quan giua X va Y"**)  
plt.show()

* Tạo mô hình

*# Số chiều input, số chiểu output*input\_dim = 8  
num\_outputs = 1  
  
input = input\_variable(input\_dim, np.float32)  
label = input\_variable((num\_outputs), np.float32)  
  
mydict = {**"w"**: **None**, **"b"**: **None**}  
*# Tạo mô hình***def** linear\_layer(input\_var, output\_dim):  
 input\_dim = input\_var.shape[0]  
 weight\_param = cntk.parameter(shape=(input\_dim, output\_dim))  
 bias\_param = cntk.parameter(shape=(output\_dim))  
 mydict[**'w'**], mydict[**'b'**] = weight\_param, bias\_param  
 **return** cntk.times(input\_var, weight\_param) + bias\_param  
  
  
z = linear\_layer(input, num\_outputs)  
*# Setup loss and evaluation functions*loss = cntk.squared\_error(z, label)  
eval\_error = cntk.squared\_error(z, label)  
  
learning\_rate = 0.002 *# Adjust according to the model*lr\_schedule = learning\_rate\_schedule(learning\_rate, UnitType.minibatch)  
learner = sgd(z.parameters, lr\_schedule)  
trainer = Trainer(z, (loss, eval\_error), [learner])

* Huấn luyện mô hình với dữ liệu train

*# Initialize the parameters for the trainer*iter = 3  
minibatch\_size = 10  
num\_samples\_to\_train = X\_train.shape[0]  
num\_minibatches\_to\_train = int(num\_samples\_to\_train / minibatch\_size)  
  
**for** no\_iter **in** range(0, iter): *# Adjust according to the model* **for** i **in** range(0, num\_minibatches\_to\_train):  
 train\_features = X\_train[(i \* minibatch\_size):(i \* minibatch\_size + minibatch\_size), :]  
 train\_labels = Y\_train[(i \* minibatch\_size):(i \* minibatch\_size + minibatch\_size), :]  
 trainer.train\_minibatch({input: train\_features, label: train\_labels})  
 training\_loss = trainer.previous\_minibatch\_loss\_average  
 eval\_error = trainer.previous\_minibatch\_evaluation\_average  
 print(**"Minibatch: {0}, Loss: {1:.4f}, Error: {2:.2f}"**.format(i, training\_loss, eval\_error))

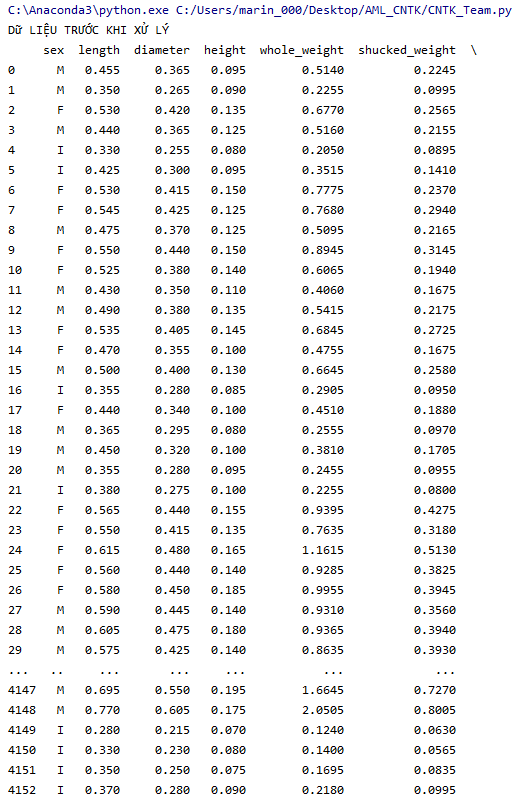
* Hiển thị kết quả

print(**"Hệ số: "**, mydict[**'w'**].value)  
print(**"BIAS: "**, mydict[**'b'**].value)

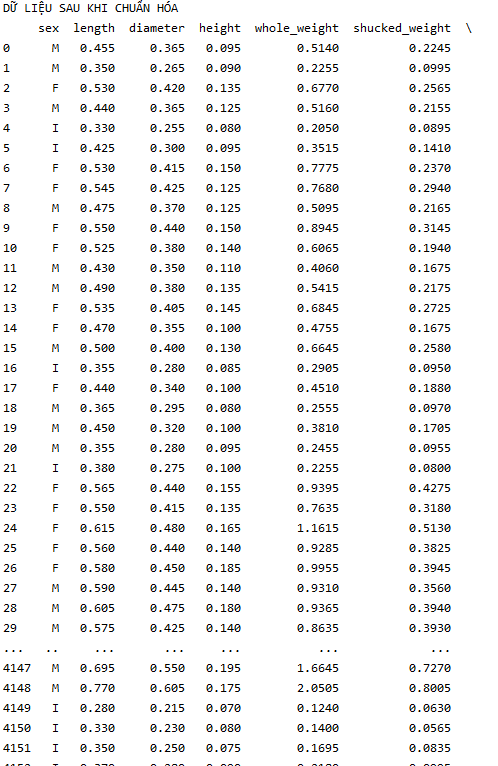
* Thử nghiệm mô hình với dữ liệu test và đánh giá

Y\_pre = np.dot(X\_test, mydict[**'w'**].value) + mydict[**'b'**].value  
*#Y\_pre = Y\_pre.astype(int)*print(**"Y test :"**, Y\_test.tolist())  
print(**"Y predict: "**, Y\_pre.tolist())  
test\_eval\_result = trainer.test\_minibatch({input: X\_test, label: Y\_test})  
print(**"Mean Square Error: {0:.2f}"**.format(test\_eval\_result))

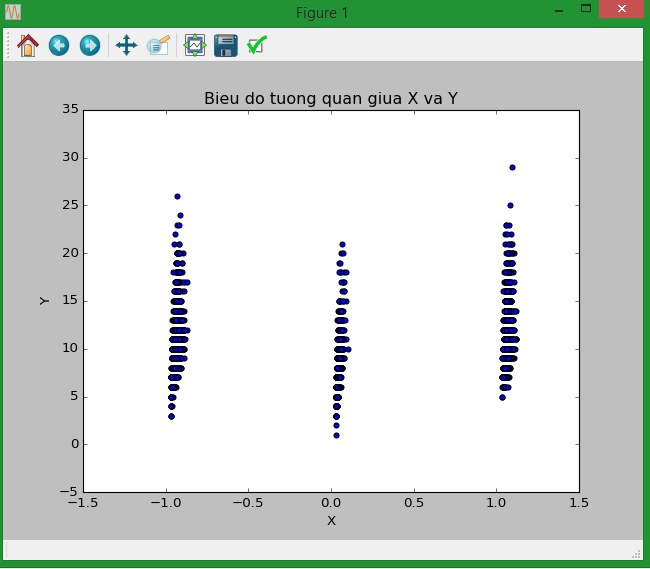
* Hình ảnh kết quả



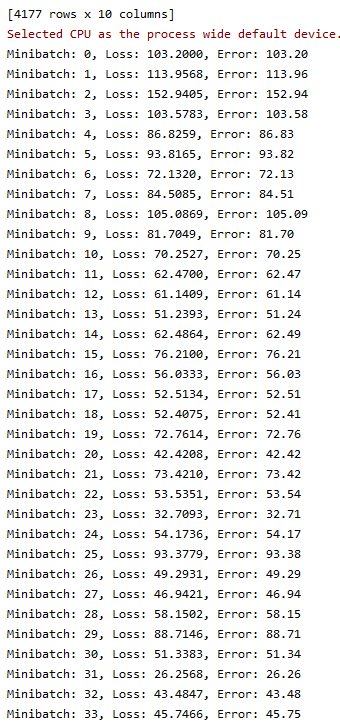
Hình 1: Dữ liệu trước khi xử lý



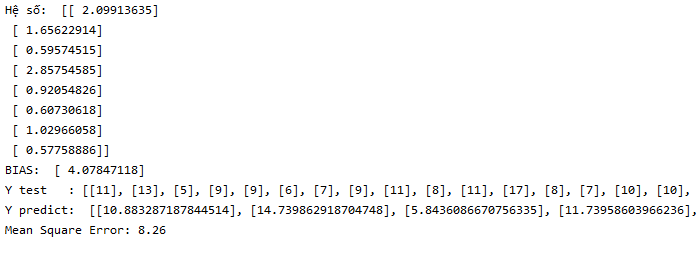
Hình 2: Dữ liệu sau khi xử lý



Hình 3: Biểu thị dữ liệu sau khi giảm chiều



Hình 4: Quá trình huấn luyện mô hình



Hình 5: Đưa ra mô hình và đánh giá mô hình

# CHƯƠNG 2: GIẢI CÁC BÀI TOÁN HỌC MÁY

## 2.1. Ứng dụng neural network của cntk để đánh giá điểm rèn luyện của học sinh dựa trên thuộc tính về lối sống. (Nguyễn Hồng Kỳ)

1. **Mô tả bài toán**

Tại hai trường học muốn đánh giá điểm rèn luyện của học sinh dựa vào các thuộc tính lối sống của học sinh. Từ đây ta áp dụng neural network cho hồi quy tuyến tính để dự đoán điểm rèn luyện.

1. **Mô tả dữ liệu**

Dữ liệu đầu vào gồm 33 cột và 650 dòng chứa thông tin về thuộc tính lối sống của học sinh qua đó đánh giá điểm rèn luyện cho học sinh.

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Ý nghĩa** | **Dữ liệu** |
| school | Tên trường học | GP tương ứng với Gabriel Pereira và MS tương ứng với Mousinho da Silveira |
| sex | Giới tính | M là nam và F là nữ |
| age | Tuổi | Tuổi học sinh từ 15 đến 22 |
| address | Vùng mà nơi học sinh sống | U là thành thị, R là nông thôn |
| famsize | Số lượng người trong gia đình | GT3 là lớn hơn 3 người và LE3 là ít hơn 3 người |
| Pstatus | Tình trạng sống chung của bố mẹ | T là bố mẹ sống gần nhau, A là cách xa nhau |
| Medu | Trình độ học vấn của mẹ | 0 là không có, 1 là tiểu học, 2 là trung học, 3 là trung học phổ thông, 4 là đại học |
| Fedu | Trình độ học vấn của bố | 0 là không có, 1 là tiểu học, 2 là trung học, 3 là trung học phổ thông, 4 là đại học |
| Mjob | Nghề nghiệp của mẹ | teacher là giáo viên, health là chăm sóc sức khỏe, service là dịch vụ, at\_home là công việc nhà, other là công việc khác |
| Fjob | Nghề nghiệp của bố | teacher là giáo viên, health là chăm sóc sức khỏe, service là dịch vụ, at\_home là công việc nhà, other là công việc khác |
| reason | Lý do chon trường | home là gần nhà, reputation là vì danh tiếng, course vì học tập, other lý do khác |
| guardian | Người giám hộ tại trường | mother là mẹ, father là bố, other là người khác |
| traveltime | Thời gian đi học từ nhà đến trường | 1 là nhỏ hơn 15 phút, 2 là từ 15 phút đến 30 phút, 3 là 30 phút đến 1 giờ, 4 là lớn hơn 1 giờ |
| studytime | Thời gian học hàng tuần | 1 là nhỏ hơn 2 giờ, 2 là từ 2 giờ đến 5 giờ, 3 là từ 5 giờ đến 10 giờ, 4 là lớn hơn 10 giờ |
| failures | Số bài thi trượt trước đó | Giá trị từ 0 đến 4 tướng ứng với số bài thi trượt |
| schoolsup | Hỗ trợ giáo dục từ nhà trường | Yes là có, no là không |
| famsup | Hỗ trợ giáo dục từ gia đình | Yes là có, no là không |
| paid | Học thêm các lớp trả phí | Yes là có, no là không |
| activities | Tham gia hoạt động ngoại khóa | Yes là có, no là không |
| nursery | Đã từng tham gia học nhà trẻ khi còn nhỏ | Yes là có, no là không |
| higher | Mong muốn học cao hơn | Yes là có, no là không |
| internet | Gia đình có sử dụng internet | Yes là có, no là không |
| romantic | Đã từng có mối quan hệ với bạn khác giới | Yes là có, no là không |
| famrel | Chất lượng mối quan hệ gia đình | Từ 1 đến 5 tương ứng từ rất tệ đến rất tốt |
| freetime | Thời gian rảnh sau ở trường | Từ 1 đến 5 tương ứng từ rất ít đến rất nhiều |
| goout | Thời gian đi chơi với bạn bè | Từ 1 đến 5 tương ứng từ rất ít đến rất nhiều |
| Dalc | Mức độ tiêu thụ đồ uống có cồn vào các ngày làm việc | Từ 1 đến 5 tương ứng từ rất ít đến rất nhiều |
| Walc | Mức độ tiêu thụ đồ uống có cồn vào các ngày cuối tuần | Từ 1 đến 5 tương ứng từ rất ít đến rất nhiều |
| health | Tình trạng sức khỏe hiện tại | Từ 1 đến 5 tương ứng từ rất kém đến rất tốt |
| absences | Số lần nghỉ học | Sô lần từ 0 đến 93 |
| G1 | Mức điểm đánh giá 2 năm trước đó | Số điểm đánh giá từ 0 đến 20 |
| G2 | Mức điểm đánh giá năm trước đó | Số điểm đánh giá từ 0 đến 20 |
| G3 | Mức điểm đánh giá hiện tại | Số điểm đánh giá từ 0 đến 20 |

Bảng 3: Bàng thuộc tính dữ liệu đánh giá rèn luyện học sinh (HK)

Giá trị cần dự đoán dự đoán: G3

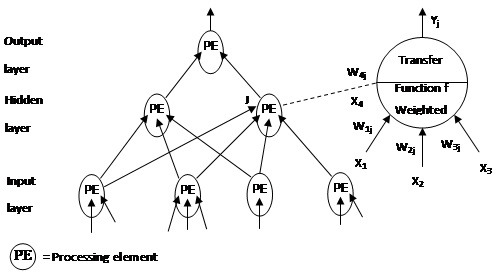
Giá trị làm biến giải thích: Các cột còn lại

Dự liệu trên bao gồm cả dữ liệu dạng số và dạng category, muốn huấn luyện được mô hình ta phải tiền xử lý dữ liệu dạng category về dạng số. Ở đây ta sẽ dùng label encoder của sklearn để xử lý dữ liệu dạng category này.

1. **Thuật toán sử dụng**

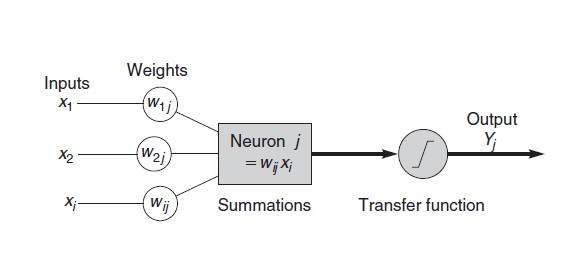
Mạng neural nhân tạo (Artificial Neural Network- ANN) là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thống thần kinh của sinh vật, bao gồm số lượng lớn các neural được gắn kết để xử lý thông tin. ANN giống như bộ não con người, được học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu giữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết.

Kiến trúc chung của một mạng nơron nhân tạo (ANN) gồm 3 thành phần đó là: input layer, hidden layer và output layer. Trong đó, lớp ẩn (hidden layer) gồm các neural nhận dữ liệu input từ các neural ở lớp (layer) trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo. Trong một ANN có thể có nhiều lớp ẩn. Trong ANN, trừ input layer thì tất cả các node thuộc các layer khác đều full-connected với các node thuộc layer trước nó. Mỗi node thuộc hidden layer nhận vào ma trận đầu vào từ layer trước và kết hợp với trọng số để ra được kết quả.



Hình 6: Mô hình kiến trúc mạng ANN

Trong đó các Processing Elements (PE) của ANN gọi là neural, mỗi neural nhận các dữ liệu vào (inputs) xử lý chúng và cho ra một kết quả (output) duy nhất. Kết quả xử lý của một neural có thể làm input cho các neural khác.



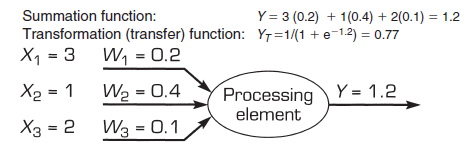
Hình 7: Quá trình xử lý của ANN

Quá trình xử lý thông tin của một ANN:

* Inputs (dữ liệu vào): Mỗi Input tương ứng với 1 thuộc tính (attribute) của dữ liệu (patterns).
* Output (kết quả): Kết quả của một ANN là một giải pháp cho một vấn đề.
* Connection Weights (Trọng số liên kết) : Đây là thành phần rất quan trọng của một ANN, nó thể hiện mức độ quan trọng (độ mạnh) của dữ liệu đầu vào đối với quá trình xử lý thông tin (quá trình chuyển đổi dữ liệu từ Layer này sang layer khác). Quá trình học (Learning Processing) của ANN thực ra là quá trình điều chỉnh các trọng số (Weight) của các input data để có được kết quả mong muốn.
* Summation Function (Hàm tổng): Tính tổng trọng số của tất cả các input được đưa vào mỗi Nơron (phần tử xử lý PE). Hàm tổng của một Nơron đối với n input được tính theo công thức sau:

iWi

* Transfer Function (Hàm chuyển đổi) hay còn gọi là (Activation Function): Hàm tổng (Summation Function) của một Nơron cho biết khả năng kích hoạt (Activation) của Nơron đó còn gọi là kích hoạt bên trong (internal activation). Các Nơron này có thể sinh ra một output hoặc không trong ANN (nói cách khác rằng có thể output của 1 Nơron có thể được chuyển đến layer tiếp trong mạng Nơron hoặc không). Mối quan hệ giữa Internal Activation và kết quả (output) được thể hiện bằng hàm chuyển đổi (Transfer Function).



Hình 8: Mô tả hàm kích hoạt trong neural

* Việc lựa chọn Transfer Function có tác động lớn đến kết quả của ANN.

*YT*

Trong đó :

*YT* : Hàm chuyển đổi

*Y* : Hàm tổng

Có nhiều loại hàm kích hoạt khác nhau như: Sigmoid, tanh, ReLU, Leaky ReLU, maxout, … Tuy nhiên nếu neural không có hàm kích hoạt thì mạng neural đó sẽ trở thành 1 neural duy nhất và các neural ở lớp sau sẽ không có ý nghĩa và khi đó mạng neural trở thành mô hình hồi quy tuyến tính. Dưới đây là chứng minh:

Ta xét đến mạng neural với 1 lớp input nhận giá trị đầu vào là X và 1 lớp output có giá trị đầu ra là Y:

Lớp thứ nhất có trọng số là W[1], hệ số bias là B[1] :

*Z[1] = W[1]\*X + B[1]*

Sau đó, output Z[1] được đẩy qua hàm kích hoạt tuyến tính g() thu được kết quả là a[1] là kết quả của lớp thứ nhất:

*a[1] = W[1]\*X + B[1]*

Tương tụ như vậy, output a[1] lại trở thành đầu vào của lớp thứ 2:

*Z[2] = W[2]\*a[1] + B[2]*

Vì ta đang giả sử hàm kích hoạt g() là tuyến tính nên ta có a[1] = c \* Z[1] (với c là một số thực). Giả sử c = 1 sử dụng các công thức bên trên ta có:

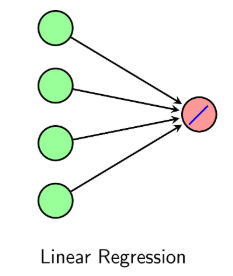
*Z[2] = W[2]\*Z[1] + B[2]*

*Z[2] = W[2]\* (W[1]\*X + B[1]) + B[2]*

Công thức này viết dạng:

*Z[1] =*

a[2] thực chất vẫn chỉ là một hàm tuyến tính của X ban đầu, vì thế việc xếp chồng các neural là vô nghĩa.



Hình 9: Biểu diễn hồi quy tuyến tính trrong mạng neural

Tuy nhiên vẫn có một số trường hợp sử dụng với hàm kích hoạt tuyến tính hay không có hàm kích hoạt vẫn được sử dụng trong các bài toán hồi quy tuyến tính như trong bài toán dưới đây.

1. **Code**

* Import các thư viện

**import** warnings  
warnings.filterwarnings(**'ignore'**)  
**import** cntk  
**import** pandas **as** pd  
**from** sklearn.model\_selection **import** train\_test\_split  
**import** matplotlib.pyplot **as** plt  
**from** cntk.ops **import** \*  
**from** cntk **import** input\_variable  
**from** cntk **import** learning\_rate\_schedule, UnitType  
**from** sklearn.decomposition **import** PCA  
**from** cntk.learners **import** sgd  
**from** cntk.train.trainer **import** Trainer  
**from** sklearn **import** preprocessing

* Đọc dữ liệu, xử lý dữ liệu và chia tỷ lệ dữ liệu train và test

*# Đọc dữ liệu*data = pd.read\_csv(**"school\_grades\_dataset.csv"**, sep=**","**)  
data.head  
  
*# In dữ liệu*print(**"DỮ LIỆU TRƯỚC KHI XỬ LÝ"**)  
print(data.shape)  
print(data)  
  
*# Xử lý dữ liệu*lbEncoder = preprocessing.LabelEncoder()  
data[**'school\_replace'**] = lbEncoder.fit\_transform(data[**'school'**])  
data[**'sex\_repalce'**] = lbEncoder.fit\_transform(data[**'sex'**])  
data[**'address\_replace'**] = lbEncoder.fit\_transform(data[**'address'**])  
data[**'famsize\_replace'**] = lbEncoder.fit\_transform(data[**'famsize'**])  
data[**'Pstatus\_replace'**] = lbEncoder.fit\_transform(data[**'Pstatus'**])  
data[**'Mjob\_replace'**] = lbEncoder.fit\_transform(data[**'Mjob'**])  
data[**'Fjob\_replace'**] = lbEncoder.fit\_transform(data[**'Fjob'**])  
data[**'reason\_replace'**] = lbEncoder.fit\_transform(data[**'reason'**])  
data[**'guardian\_replace'**] = lbEncoder.fit\_transform(data[**'guardian'**])  
data[**'schoolsup\_replace'**] = lbEncoder.fit\_transform(data[**'schoolsup'**])  
data[**'famsup\_replace'**] = lbEncoder.fit\_transform(data[**'famsup'**])  
data[**'paid\_replace'**] = lbEncoder.fit\_transform(data[**'paid'**])  
data[**'activities\_replace'**] = lbEncoder.fit\_transform(data[**'activities'**])  
data[**'nursery\_replace'**] = lbEncoder.fit\_transform(data[**'nursery'**])  
data[**'higher\_replace'**] = lbEncoder.fit\_transform(data[**'higher'**])  
data[**'internet\_replace'**] = lbEncoder.fit\_transform(data[**'internet'**])  
data[**'romantic\_replace'**] = lbEncoder.fit\_transform(data[**'romantic'**])  
  
data = data.drop([**'school'**, **'sex'**, **'address'**, **'famsize'**, **'Pstatus'**, **'Mjob'**, **'Fjob'**, **'reason'**, **'guardian'**, **'schoolsup'**,  
 **'famsup'**, **'paid'**, **'activities'**, **'nursery'**, **'higher'**, **'internet'**, **'romantic'**], axis=1)  
  
*# Dữ liệu sau khi xử lý*print(**"DỮ LIỆU SAU KHI XỬ LÝ"**)  
print(data)  
  
Y = data[**'G3'**].values  
Y = Y[:, **None**]  
X = data.drop([**'G3'**], axis=1).values  
*# Phân loại dữ liệu train, test*X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y,test\_size=0.2)  
*# Biểu diễn dữ liệu  
# Giảm chiều (PCA)*pca = PCA(n\_components=1)  
X\_scat = pca.fit\_transform(X\_train)  
*# Plot*plt.scatter(X\_scat, Y\_train)  
plt.xlabel(**"Gia tri X"**)  
plt.ylabel(**"Gia tri Y"**)  
plt.title(**"Bieu do tuong quan X va Y"**)  
plt.show()

* Tạo mô hình

*# Define the network using CNTK primitives to define a single node with no activation function  
# Số chiều input, số chiểu output*input\_dim = 32  
num\_outputs = 1  
  
input = input\_variable(input\_dim, np.float32)  
label = input\_variable((num\_outputs), np.float32)  
  
mydict = {**"w"**: **None**, **"b"**: **None**}  
*# Tạo mô hình***def** linear\_layer(input\_var, output\_dim):  
 input\_dim = input\_var.shape[0]  
 weight\_param = cntk.parameter(shape=(input\_dim, output\_dim))  
 bias\_param = cntk.parameter(shape=(output\_dim))  
 mydict[**'w'**], mydict[**'b'**] = weight\_param, bias\_param  
 **return** cntk.times(input\_var, weight\_param) + bias\_param  
  
z = linear\_layer(input, num\_outputs)  
*# Setup loss and evaluation functions*loss = cntk.squared\_error(z, label)  
eval\_error = cntk.squared\_error(z, label)  
  
learning\_rate = 0.0001 *# Adjust according to the model*lr\_schedule = learning\_rate\_schedule(learning\_rate, UnitType.minibatch)  
learner = sgd(z.parameters, lr\_schedule)  
trainer = Trainer(z, (loss, eval\_error), [learner])

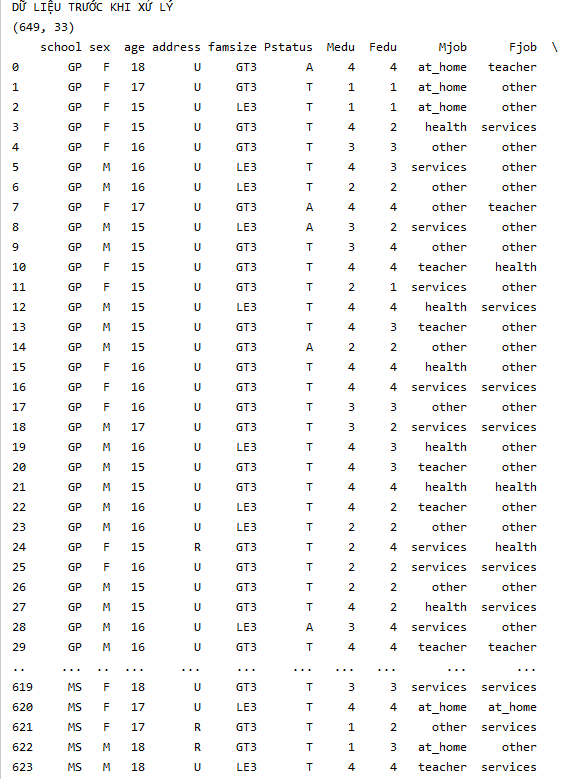
* Huấn luyện mô hình với dữ liệu train

*# Initialize the parameters for the trainer*iter = 3  
minibatch\_size = 25  
num\_samples\_to\_train = X\_train.shape[0]  
num\_minibatches\_to\_train = int(num\_samples\_to\_train / minibatch\_size)  
  
**for** no\_iter **in** range(0, iter): *# Adjust according to the model* **for** i **in** range(0, num\_minibatches\_to\_train):  
 train\_features = X\_train[(i \* minibatch\_size):(i \* minibatch\_size + minibatch\_size), :]  
 train\_labels = Y\_train[(i \* minibatch\_size):(i \* minibatch\_size + minibatch\_size), :]  
 trainer.train\_minibatch({input: train\_features, label: train\_labels})  
 training\_loss = trainer.previous\_minibatch\_loss\_average  
 eval\_error = trainer.previous\_minibatch\_evaluation\_average  
 print(**"Minibatch: {0}, Loss: {1:.4f}, Error: {2:.2f}"**.format(i, training\_loss, eval\_error))

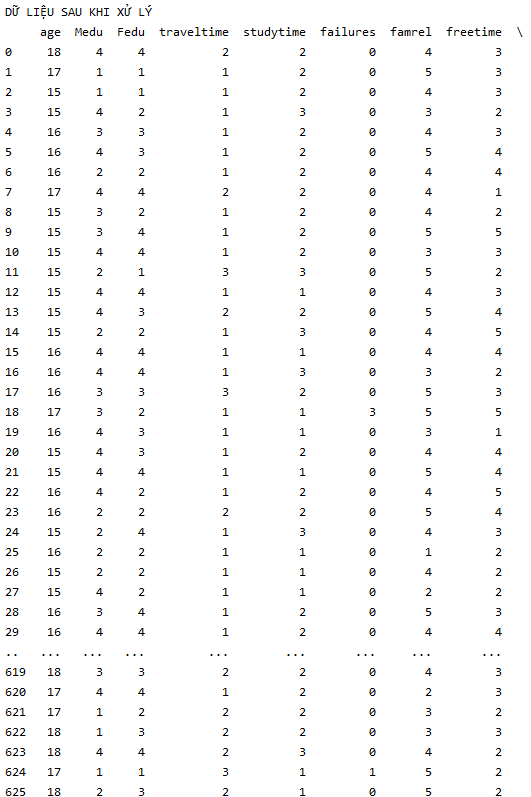
* Kiểm định mô hình với dữ liệu test

print(**"Hệ số: "**, mydict[**'w'**].value)  
print(**"BIAS: "**, mydict[**'b'**].value)  
  
*#test\_features = X\_test  
#test\_labels = Y\_test*Y\_pre = np.dot(X\_test, mydict[**'w'**].value) + mydict[**'b'**].value  
*#Y\_pre = Y\_pre.astype(int)*print(**"Y test :"**, Y\_test.tolist())  
print(**"Y predict: "**, Y\_pre.tolist())  
test\_eval\_result = trainer.test\_minibatch({input: X\_test, label: Y\_test})  
print(**"Mean Square Error: {0:.2f}"**.format(test\_eval\_result))

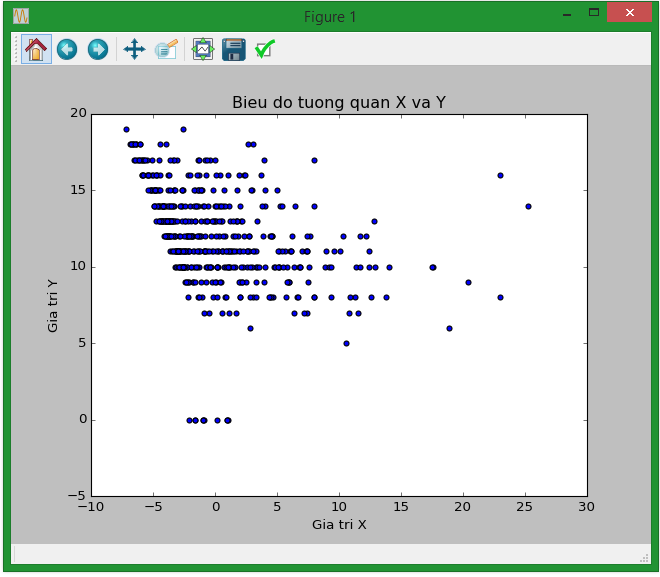
1. **Kết quả và đánh giá**



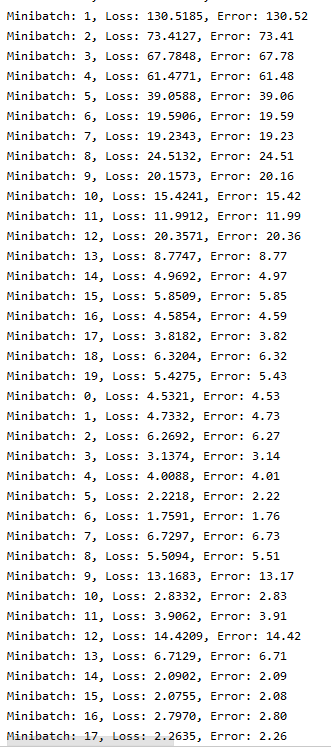
Hình 10: Dữ liệu tiền xử lý (HK)



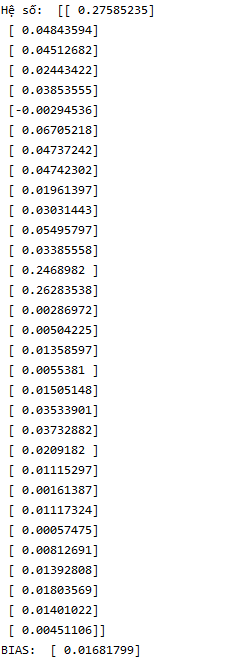
Hình 11: Dữ liệu sau khi xử lý (HK)



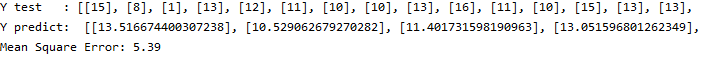
Hình 12: Biểu đồ tương quan giữa X và Y (HK)



Hình 13: Quá trình huấn luyện mô hình



Hình 14: Hệ số hồi quy và bias (HK)



Hình 15: So sánh kết quả Y test với Y dự đoán và MSE (HK)

**Đánh giá:** Mô hình đã đưa ra được nghiệm, thử nghiệm với các giá trị test thì cho kết quả khá tốt với MSE bằng 5.39 tương đương sai số giữa Y dự đoán và Y thực tế gần bằng 2.32

## 2.2. Ứng dụng neural network để dự đoán mắc bệnh tim hay không dự trên các thông số liện quan đến bệnh tim. (Nguyễn Quốc Tuấn)

1. **Mô tả bài toán**

Bệnh tim mạch, là nguy cơ hang đầu gây tử vong trên thế giới, việc phát hiện sớm và điều trị kịp thời là một vấn đề hết sức quan trọng. Vì vậy em đã sử dụng hồi quy logistic để dự đoán một người có bị bệnh tim không dựa vào dữ liệu kết quả khám sang lọc.

1. **Mô tả dữ liệu**

Dữ liệu đầu vào gồm 14 cột và 305 dòng chứa dữ liệu khám sàng lọc, qua đó chẩn đoán người đó mắ bệnh tim mạch hay không.

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Ý nghĩa** | **Dữ liệu** |
| age | Tuổi | Tuổi của bệnh nhân |
| sex | Giới tính | Male là nam và Female là nữ |
| cp | Kiểu đau thắt ngực | Có 4 loại kiểu đau thắt ngực: typical angina,atypical angine, non-anginal, asymptomatic |
| trestbps | Chỉ số huyết áp lúc nghỉ | Chỉ só huyết áp (mmHg) |
| chol | Nồng độ chlosterol máu | Chỉ số cholesterol (mg/dl) |
| fbs | Chỉ số đường huyết máu lúc đói | Chỉ số bằng 1 nếu nồng độ lớn hơn 120 mg/dl, 0 nếu nhỏ hơn 120 mg/dl |
| restecg | Kết quả điện tâm đồ | Giá trị 0 là bình thường, 1 có bất thường ở sóng ST-T, 2 là phì đại thất trái |
| thalach | Nhìm tim tối đa | Chỉ số nhịp tim tối đa: (nhịp/phút) |
| exang | Đau thắt ngực do tập thể dục | 1 là có bị đau, 0 là không đau |
| oldpeak | Độ lõm của đoạn sóng ST | 1 là có, 0 là không |
| slope | Độ dốc của đoạn sóng ST | 1 là dốc lên, 2 là phẳng, 3 là đi xuống |
| ca | Số mạch chính được định màu bởi flourosopy | Định màu flourosopy từ 0 đến 3 |
| thal | Độ nhập nháp cơ tim | 3 là bình thường, 6 là cố định khuyết tật, 7 là phục hồi khiếm khuyết |
| num | Đánh giá có bị mắc bệnh tim mạch hay không | 0 là có 1 là không |

Bảng 4: Bảng thuộc tính dữ liệu bệnh tim (QT)

Giá trị cần dự đoán dự đoán: num

Giá trị làm biến giải thích: Các cột còn lại

Dự liệu trên bao gồm cả dữ liệu dạng số và dạng category, muốn huấn luyện được mô hình ta phải tiền xử lý dữ liệu dạng category về dạng số. Ở đây ta sẽ dùng label encoder của sklearn để xử lý dữ liệu dạng category này.

1. **Thuật toán sử dụng**

Hai mô hình tuyến tính (linear models) Linear Regression và Perceptron Learning Algorithm (PLA) chúng ta đã biết đều có chung một dạng:

Trong đó f() được gọi là activation function và được hiểu là dữ liệu mở rông với = 1 được thêm vào để thuận tiện cho việc tính toán. Với linear regression thì f(s) = s, với PLA thì f(s) = sgn(s). Trong linear regression, tích vô hướng được trực tiếp sử dụng để dự đoán output loại này phù hợp nếu chúng ta cần dự đoán một giá trị thực của đầu ra không bị chặn trên và dưới. Trong PLA, đầu ra chỉ nhận một trong hai giá trị 1 hoặc -1, phù hợp với các bài toán binary classification.

Mô hình logistic regression:

* Đầu ra dự đoán của linear regression:
* Đầu ra dự đoán của PLA:

Đầu ra của logistic regression thường được viết dưới dạng:

Trong đó θ được gọi là logistic function. Một số activation cho mô hình tuyến tính được cho trong hình dưới đây:

Hình 16: Đồ thị các activation function khác nhau

* Hàm sigmoid

Hàm sigmoid thường được sử dụng nhiều vì nó chặn trong khoảng từ 0 đến 1 phù hợp với bài toán phân loại như logistic regression.

Lý do sigmoid được chọn nhiều nhất:

Hàm sigmoid có đạo hàm khá đặc biệt:

* Hàm tanh

NGoài hàm sigmoid ra thì hàm tanh cũng hay sử được sử dụng.

Hàm số này nhận giá trị trong khoảng (-1, 1) nhưng có thể dễ dàng đưa nó về khoảng (0, 1)

* Hàm mất mát:

Ta sẽ sử dụng thuật toán SGD (Stochastic gradient descent) để tối ưu hàm mất mát, ta sẽ được hàm mất mát tối ưu là hàm sigmoid.

* Công thức cập nhật cho logistic sigmoid regression

Công thức cập nhật theo SGD là:

1. **Code**

* Import các thư viện

**import** warnings  
warnings.filterwarnings(**'ignore'**)  
**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**from** sklearn.metrics **import** confusion\_matrix  
**from** sklearn.metrics **import** classification\_report  
**from** sklearn **import** preprocessing  
**import** cntk **as** C  
**from** cntk.ops **import** \*

* Đọc dữ liệu, xử lý dữ liệu

*# Đọc dữ liệu từ tệp CSV*data = pd.read\_csv(**"heart\_disease.csv"**, sep=**","**) *# Chu y dau ,*data.head()  
print(**"DỮ LIỆU TRƯỚC KHI XỬ LÝ"**)  
print(data.shape)  
print(data)  
  
*# Xử lý dữ liệu*lbEncoder = preprocessing.LabelEncoder()  
data[**'sex\_replace'**] = lbEncoder.fit\_transform(data[**'sex'**])  
data[**'cp\_replace'**] = lbEncoder.fit\_transform(data[**'cp'**])  
data = data.drop([**'sex'**, **'cp'**], axis=1)  
print(**"DỮ LIỆU SAU KHI XỬ LÝ"**)  
print(data)  
X\_data = data.drop(**"num"**, axis=1).values  
print(**"FEATURE"**)  
print(X\_data)  
Y = data[**"num"**].values  
Y\_data = Y[:, **None**]  
print(**"LABEL"**)  
print(Y\_data)

* Tạo mô hình

*# Lấy dữ liệu ngẫu nhiên từ tập dữ liệu***def** generate\_random\_data\_sample(sample\_size, feature\_dim, num\_classes):  
 *# Create synthetic data using NumPy.* Y = Y\_data  
 *# Make sure that the data is separable* X = X\_data  
 *# Specify the data type to match the input variable used later in the tutorial  
 # (default type is double)* X = X.astype(np.float32)  
  
 *# convert class 0 into the vector "1 0 0",  
 # class 1 into the vector "0 1 0", ...* class\_ind = [Y == class\_number **for** class\_number **in** range(num\_classes)]  
 Y = np.asarray(np.hstack(class\_ind), dtype=np.float32)  
 **return** X, Y  
  
  
*# Create the input variables denoting the features and the label data. Note: the input  
# does not need additional info on the number of observations (Samples) since CNTK creates only  
# the network topology first*train\_size = 250 *# Lấy 250 mẫu trong data để train*features, labels = generate\_random\_data\_sample(train\_size, input\_dim, num\_output\_classes)  
  
*# Plot the data*colors = [**'r' if** label == 0 **else 'b' for** label **in** labels[:, 0]]  
plt.scatter(features[:, 0], features[:, 1], c=colors)  
plt.xlabel(**"Gia tri X"**)  
plt.ylabel(**"Gia tri Y"**)  
plt.show()  
  
feature = C.input\_variable(input\_dim, np.float32)  
print(**"Feature"**)  
print(input\_dim)  
print(feature)  
  
*# Define a dictionary to store the model parameters*mydict = {}  
  
  
**def** linear\_layer(input\_var, output\_dim):  
 input\_dim = input\_var.shape[0]  
 weight\_param = C.parameter(shape=(input\_dim, output\_dim))  
 bias\_param = C.parameter(shape=(output\_dim))  
  
 mydict[**'w'**], mydict[**'b'**] = weight\_param, bias\_param  
  
 **return** C.times(input\_var, weight\_param) + bias\_param  
  
  
output\_dim = num\_output\_classes  
z = linear\_layer(feature, output\_dim)  
  
label = C.input\_variable(num\_output\_classes, np.float32)  
loss = C.cross\_entropy\_with\_softmax(z, label)  
  
eval\_error = C.classification\_error(z, label)  
  
*# Instantiate the trainer object to drive the model training*learning\_rate = 0.5  
lr\_schedule = C.learning\_rate\_schedule(learning\_rate, C.UnitType.minibatch)  
learner = C.sgd(z.parameters, lr\_schedule)  
trainer = C.Trainer(z, (loss, eval\_error), [learner])

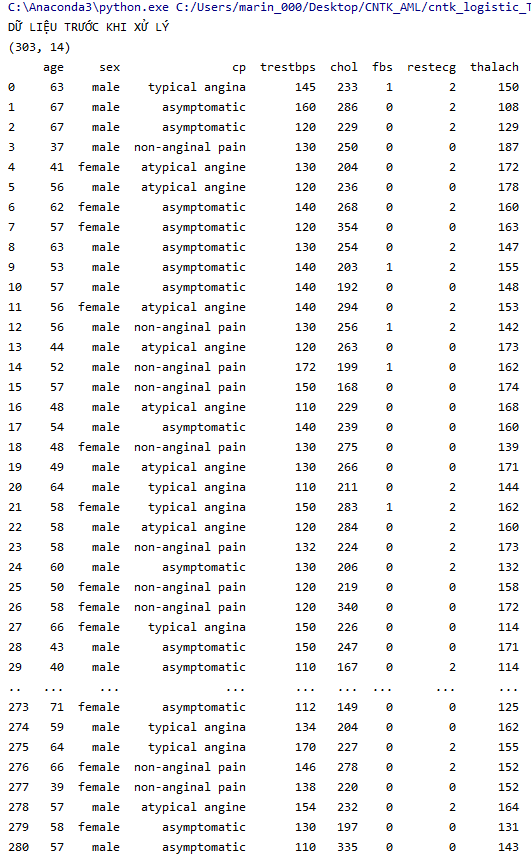
* Huấn luyện mô hình

**def** moving\_average(a, w=10):  
 **if** len(a) < w:  
 **return** a[:]  
 **return** [val **if** idx < w **else** sum(a[(idx - w):idx]) / w **for** idx, val **in** enumerate(a)]  
  
  
*# Define a utility that prints the training progress***def** print\_training\_progress(trainer, mb, frequency, verbose=1):  
 training\_loss, eval\_error = **"NA"**, **"NA"  
  
 if** mb % frequency == 0:  
 training\_loss = trainer.previous\_minibatch\_loss\_average  
 eval\_error = trainer.previous\_minibatch\_evaluation\_average  
 **if** verbose:  
 print(**"Minibatch: {0}, Loss: {1:.4f}, Error: {2:.2f}"**.format(mb, training\_loss, eval\_error))  
  
 **return** mb, training\_loss, eval\_error  
  
*# Initialize the parameters for the trainer*minibatch\_size = 25  
num\_samples\_to\_train = 20000  
num\_minibatches\_to\_train = int(num\_samples\_to\_train / minibatch\_size)  
  
**from** collections **import** defaultdict  
  
*# Run the trainer and perform model training*training\_progress\_output\_freq = 50  
plotdata = defaultdict(list)  
  
**for** i **in** range(0, num\_minibatches\_to\_train):  
 features, labels = generate\_random\_data\_sample(minibatch\_size, input\_dim, num\_output\_classes)  
 *# Assign the minibatch data to the input variables and train the model on the minibatch* trainer.train\_minibatch({feature: features, label: labels})  
 batchsize, loss, error = print\_training\_progress(trainer, i,  
 training\_progress\_output\_freq, verbose=1)  
  
  
*# Run the trained model on a newly generated dataset*test\_minibatch\_size = X\_data.shape[0] - train\_size *# Lấy dữ liệu còn lạ để test*features, labels = generate\_random\_data\_sample(test\_minibatch\_size, input\_dim, num\_output\_classes)  
  
trainer.test\_minibatch({feature: features, label: labels})  
  
out = C.softmax(z)  
result = out.eval({feature: features})  
  
*# Model parameters*print(**"Hệ số: "**,mydict[**'w'**].value)  
print(**"Bias: "**, mydict[**'b'**].value)

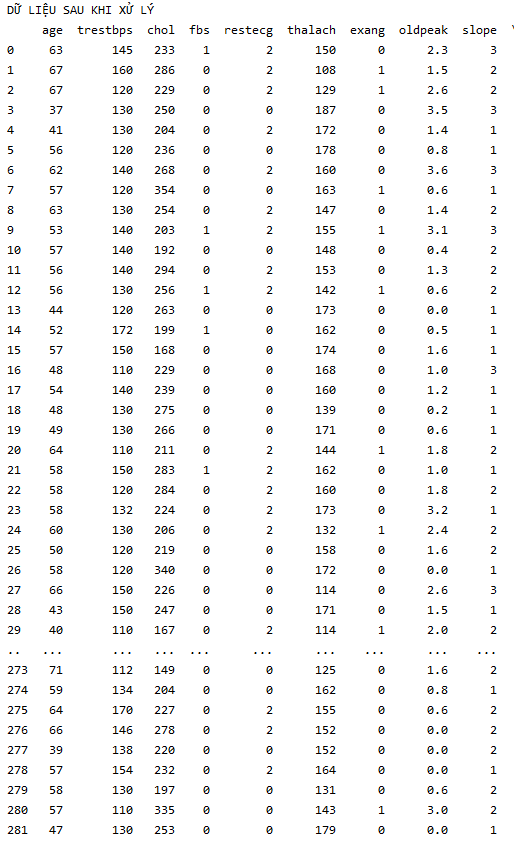
* Đánh giá mô hình với dữ liệu test

Y\_test = [np.argmax(label) **for** label **in** labels]  
Y\_pre = [np.argmax(x) **for** x **in** result]  
print(**"Giá trị Y test :"**, Y\_test)  
print(**"Giá trị Y dự đoán:"**, Y\_pre)  
  
Y\_test = np.array(Y\_test)  
Y\_pre = np.array(Y\_pre)  
num\_Y\_true = 0  
**for** i **in** range(0, Y\_pre.shape[0]):  
 **if**(Y\_pre[i] == Y\_test[i]):  
 num\_Y\_true += 1  
accuracy = (num\_Y\_true/Y\_test.shape[0]) \* 100  
print(**"ACCURACY: "**, accuracy, **"%"**)  
matrix = confusion\_matrix(y\_true=Y\_test, y\_pred=Y\_pre)  
print(**"CONFUSION MATRIX"**)  
print(matrix)  
print(**"CLASSFICATION REPORT"**)  
print(classification\_report(y\_true=Y\_test, y\_pred=Y\_pre))

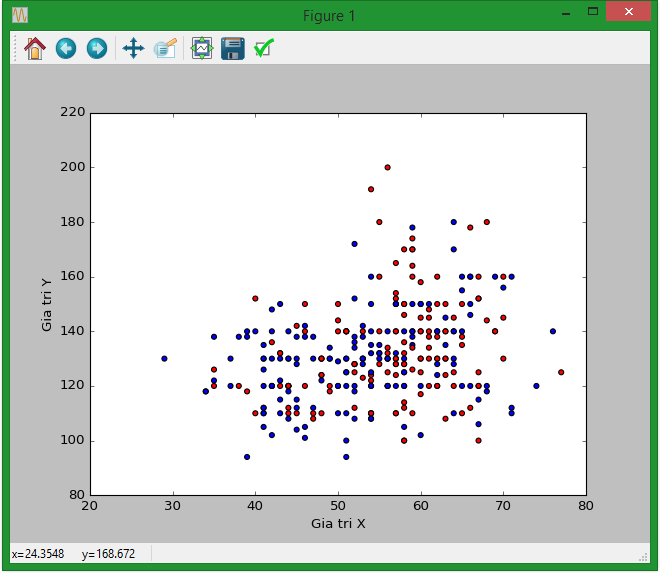
1. **Kết quả và đánh giá**



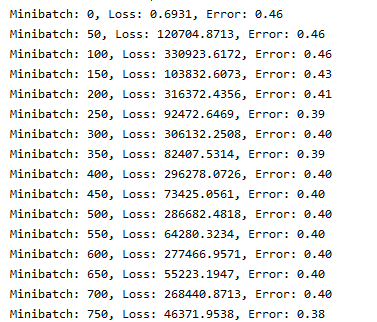
Hình 17: Dữ liệu trước khi xử lý (QT)



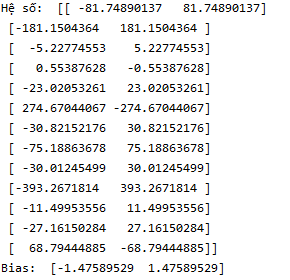
Hình 18: Dữ liệu sau khi xử lý (QT)



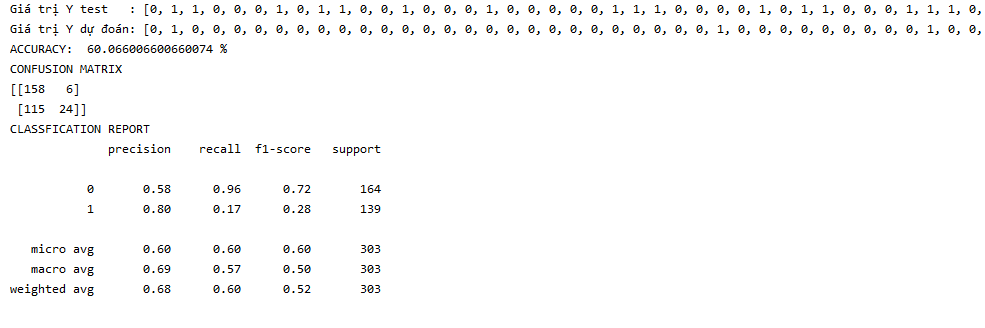
Hình 19: Biểu đồ tương quan giữa X và Y (QT)



Hình 20: Quá trình huấn luyện mô hình



Hình 21: Mô hình đưa ra các hệ số và bias của boundary



Hình 22: Kết quả kiểm tra với dữ liệu test

**Đánh giá:** Mô hình đã dự đoán số người bị bệnh tim và không bị bệnh với tỷ lệ là 60% chính xác.

## 2.3. Ứng dụng hồi quy tuyến tính để dự đoán hiệu năng máy tính dựa trên các thông số phần cứng. (Trịnh Thành Đạt)

1. **Mô tả bài toán**

Hiệu CPU khá quan trọng, nó gần như quyết định chiếc máy tính đó có thể chạy được các tác vụ nặng hay không. Và dự đoán hiệu suất CPU giúp cho người dùng có thể lựa chọn CPU tốt hơn dựa vào kết quả dự đoán.

1. **Mô tả dữ liệu**

Dữ liệu gồm 10 cột và 220 dòng. Dưới đây là bảng chi tiết dữ liệu đầu vào.

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Ý nghĩa** | **Dữ liệu** |
| vendor\_name | Hãng sản xuất | Bao gồm 30 hãng sản xuất khác nhau |
| model\_name | Tên CPU | Tên CPU của các hãng |
| MYCT | Chu kỳ CPU | Chu kỳ tính theo nano giây |
| MMIN | Bộ nhớ chính tối thiểu | Tính theo kb |
| MMAX | Bộ nhớ chính tối đa | Tính theo kb |
| CACH | Bộ nhớ đệm | Bộ nhớ đệm tính theo kb |
| CHMIN | Số kênh tối thiểu | Có giá trị từ 1 đến 52 |
| CHMAX | Số kênh tối đa | Có giá trị từ 1 đến 172 |
| PRP | Hiệu suất tương đối trong điều kiện lý tưởng | Hiệu suất của máy theo nhà sản xuất |
| ERP | Hiệu suất thực tế | Hiệu suất máy tính thực tế chạy |

Bảng 5: Bảng thuộc tính dữ liệu đánh giá hiệu năng máy tính (D)

Giá trị cần dự đoán dự đoán: ERP

Giá trị làm biến giải thích: Các cột còn lại

1. **Thuật toán sử dụng**

Thuật toán sử dụng trong bài toán là hổi quy tuyến tính (linear regression) bằng phương pháp đại số tuyến tính.

Hồi quy tuyến tính có phương trình đạng :

F(x) = α + w1x1 + w2x2 + w3x3 (1)

Trong đó, w1, w2, w3, w0, là các hằng số, α còn được gọi là bias hay sai số. Mối quan hệ giữa y f(x) bên trên là một mối quan hệ tuyến tính (linear). Bài toán chúng ta đang làm là một bài toán thuộc loại regression. Bài toán đi tìm các hệ số tối ưu { w1, w2, w3, w0 } chính vì vậy được gọi là bài toán Linear Regression.

Trong phương trình (1) nếu chúng ta đặt **w = [**w1, w2, w3, w0**]**T là một vecter (cột) hệ số cần phải tối ưu và = [1, x1, x2, x3] (đọc là x bar trong tiếng Anh) là vector (hàng) dữ liệu đầu vào mở rộng. Số 1 ở đầu được thêm vào để phép tính đơn giản hơn và thuận tiện cho việc tính toán. Khi đó, phương trình (1) có thể được viết lại dưới dạng:

y ( trong đó là một vecter hàng)

Chúng ta mong muốn rằng sự sai khác giữa giá trị thực và giá trị dự đoán ŷ (đọc là y hat trong tiếng Anh) là nhỏ nhất. Nói cách khác, chúng ta muốn giá trị sau đây càng nhỏ càng tốt:

Trong đó hệ số (lại) là thuận tiện cho việc tính toán khi tính đạo hàm thì số sẽ bị triệt tiêu. Chúng ta cần vì có thể là một số âm, việc nói nhỏ nhất sẽ không đúng vì khi là rất nhỏ nhưng sự sai lệch là rất lớn.

Máy học từ giá trị trung bình của một hàm mất mát. Đây là một phương pháp đánh giá độ hiệu quả của một thuật toán nào đó trên bộ dữ liệu cho trước. Nếu kết quả dự đoán chênh lệch quá nhiều so với kết quả thực tế, hàm mất mát sẽ là một số rất lớn. Điều tương tự xảy ra với tất cả các cặp (input, outcome) (xi, yi), i = 1, 2, 3, …, N với N là số lượng dữ liệu quan sát được. Để hàm mất mát nhỏ nhất khi đó tổng sai số là nhỏ nhất tương đương với việc tìm **w** để hàm số sau đạt giá trị nhỏ nhất:

F(w) = (2)

Hàm số F(w) được gọi là hàm mất mát (loss function) của bài toán Linear Regression. Chúng ta luôn mong muốn rằng sự mất mát (sai số) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số **w** sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị của **w** làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là điểm tối ưu (optimal point), ký hiệu:

****

Trước khi đi tìm lời giải, chúng ta đơn giản hóa phép toán trong phương trình hàm mất mát (2). Đặt là một vector cột chứa tất cả các output của training data; = là ma trận dữ liệu đầu vào (mở rộng) mà mỗi hàng của nó là một điểm dữ liệu. Khi đó hàm số mất mát F(w) được viết dưới dạng ma trận đơn giản hơn:

F(w) = = (3)

Với ||z||2 là Euclidean norm (chuẩn Euclid, hay khoảng cách Euclid), nói cách khác là tổng của bình phương mỗi phần tử của vector **z**. Tới đây, ta đã có một dạng đơn giản của hàm mất mát được viết như phương trình (3).

Để tìm nghiệm cho một bài toán tối ưu chúng ta thường giải phương trình đạo hàm (gradient) bằng 0. Tất nhiên đó là khi việc tính đạo hàm và việc giải phương trình đạo hàm bằng 0 không quá phức tạp. Thật may mắn, với các mô hình tuyến tính, hai việc này là khả thi.

Đạo hàm theo **w** của hàm mất mát là:



Phương trình đạo hàm bằng 0 tương đương với:



(Ký hiệu  nghĩa là  bằng **b**)

Nếu ma trận vuông  khả nghịch thì có nghiệm duy nhất là 

Nếu ma trận A có định thức bằng 0 thì phương trình có vô nghiệm hoặc vô số nghiệm khi đó ta sử dụng khái niệm giả nghịch đảo A† (đọc là A dagger). Giả nghịch đảo là trường hợp tổng quát của ma trận nghịch đảo khi ma trận đó không khả nghịch hoặc thậm chí không vuông.

Với khái niệm trên, điểm tối ưu của hồi quy tuyến tính có dạng:



Trong bài sẽ sử dụng thư viện sklearn để giải bài toán

1. **Code**

* Import các thư viện

**import** warnings  
warnings.filterwarnings(**'ignore'**)  
**import** numpy **as** np  
**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn **import** metrics **as** sq  
**from** sklearn **import** linear\_model  
**from** sklearn.decomposition **import** PCA  
regr = linear\_model.LinearRegression()

* Đọc dữ liệu, biểu diễn dữ liệu và phân chia dữ liệu train và test

*#Đọc tệp*data = pd.read\_csv(**"computer\_hardware.csv"**, sep=**","**)  
data.head  
*#In dữ liệu*print(data)  
*#Lấy dataframe ERP làm biên mục tiêu*Y = data[**'ERP'**]  
print(Y)  
*#Lấy datafarame không chứa ERP làm biến giải thích*X = data.drop(**"ERP"**, axis = 1)  
print(X)  
*#Biểu diễn dữ liệu  
#Giảm chiều*pca = PCA(n\_components=1)  
X\_scat = pca.fit\_transform(X)  
*#Bảng biểu so sánh giá trị y dự đoán và y thực tế*plt.scatter(X\_scat, Y)  
plt.xlabel(**"X"**)  
plt.ylabel(**"Y"**)  
plt.title(**"Tuong quan X va Y"**)  
plt.show()  
  
*#Phân loại dữ liệu train và test*X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2)

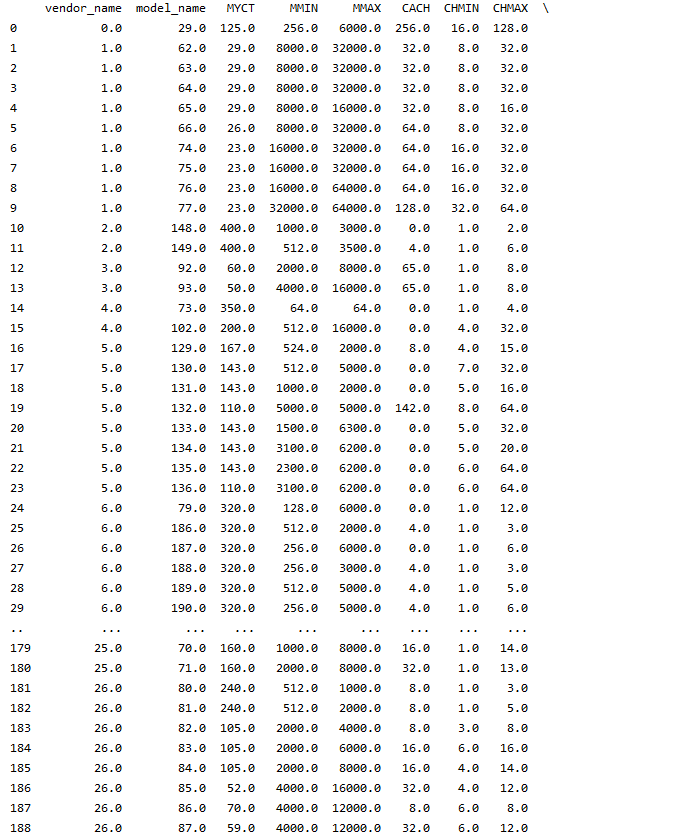
* Tạo mô hình và huấn luyện mô hình

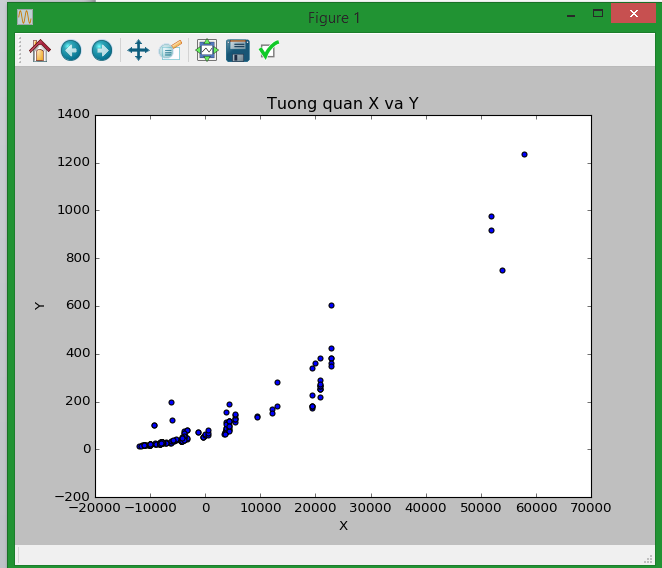
*#Tạo model suy đoán*regr.fit(X\_train, Y\_train)  
*#In hệ số hồi quy của các biến giải thích xếp theo thứ tự tăng dần*print(**"\nHỆ SỐ HỒI QUY"**)  
print(pd.DataFrame({**"Tên"**: X\_train.columns, **"Hệ số"**: np.abs(regr.coef\_)}).sort\_values(by=**'Hệ số'**))  
*# BIAS*print(**"\nBIAS"**)  
print(regr.intercept\_)

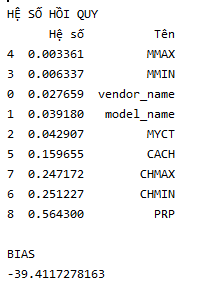
* Đánh giá mô hình với dữ liệu test

*#Tiến hành dự đoán với bộ dữ liệu test*Y\_pred = regr.predict(X\_test)  
print(**"\nGIÁ TRỊ Y DỰ ĐOÁN"**)  
print(Y\_pred)  
*#In giá trị y test thực tế*print(**"\nGIÁ TRỊ Y THỰC TẾ"**)  
print(Y\_test)  
*#Kiểm tra mức độ lỗi của model (Mean Squared Error)*mse = sq.mean\_squared\_error(Y\_test, Y\_pred)  
print(**"KIỂM TRA MỨC ĐỘ LỖI CỦA MÔ HÌNH"**)  
print(mse)  
print(**"SCORE: "**, regr.score(X\_train, Y\_train))

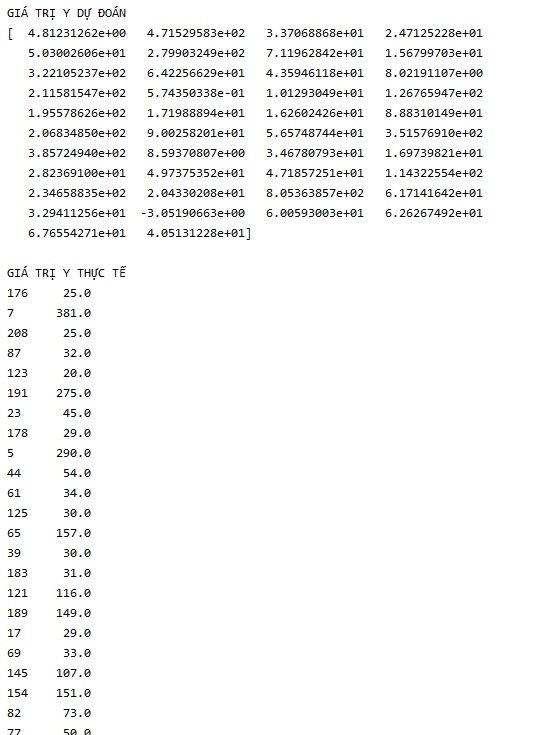
1. **Kết quả và đánh giá**

Hình 23: Dữ liệu của bài toán (D)

Hình 23: Biểu đồ tương quan giữa X và Y (D)



Hình 24:Hệ số hồi quy và bias



Hình 25: Kiểm tra lại mô hình với dữ liệu test (D)

**Đánh giá:** Mô hình đã đưa ra được nghiệm, thử nghiệm với các giá trị test thì cho MSE là 1117.

# KẾT LUẬN

Qua đây nhóm đã cài đặt được và sử dụng được framework cognitive toolkit của miircrosoft. Nhóm đã biết tiền xử lý dữ liệu trước khi đưa vào huấn luyện mô hình. Áp dụng framework vào để tìm hiểu và giải những bài toán cơ bản của học máy như hồi quy tuyến tính, hồi quy logistic, mạng neural. Tuy nhiên vẫn còn hạn chế như: chỉ làm với những bài toán nhỏ lẻ không áp dụng nhiều trong thực tế, trong tương lai nhóm sẽ cố gắng học hỏi và tìm hiểu thêm để áp dụng những thành tự của học máy vào trong cuộc sống hang ngày.

Do thời gian và kiến thức có hạn nên báo cáo chúng em vẫn còn nhiều sai sót, rất mong các thầy cô góp ý giúp chúng em hoàn thiện báo cáo hơn nữa.

# DANH MỤC TÀI LIỆU THAM KHẢO

https://stackoverflow.com

https://www.cntk.ai

https://www.tutorialspoint.com/microsoft\_cognitive\_toolkit/index.htm

https://docs.microsoft.com/en-us/cognitive-toolkit/