Advanced Learning Algorithms - Andrew Ng

Contents

1. Introduction **1**

2. Neural Network with binary classification **2**
Định nghĩa Neural Network with tensorflow . . . 2
Định nghĩa Neural Network with Numpy (let's take the course :D) . . . 3
Why is relu good ? . . . 3
Training with tensorflow . . . 3

3. Neural Network with multi-class classification **4**
Định nghĩa với Tensorflow (3 classes) . . . 4
Lưu ý . . . 4
Training with tensorflow . . . 4

4. How to evaluate and fine tune model **5**
How to evaluate? . . . 5
Bias and variance . . . 5
When is high bias or high variance ? . . . 5
How to judge whether model is overfit? . . . 5
Solution . . . 5
Error analysis . . . 6
Transfer learning . . . 6

5. Decision Tree **6**
Tree . . . 6
Tree Ensamble . . . 6
Random Forest . . . 6
XGBoost . . . 7

6. When should we choose decision tree or neural network ? **7**

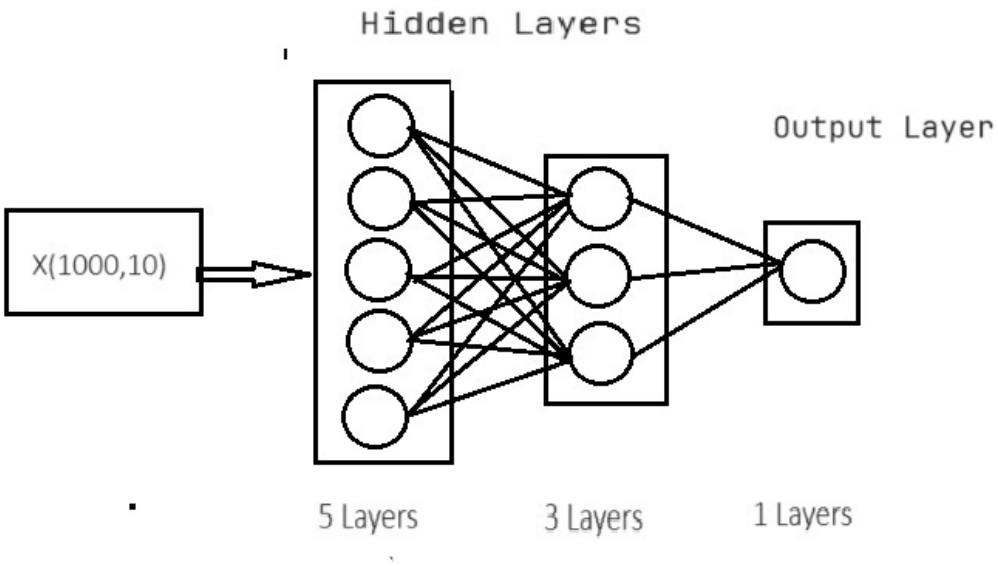
7. Titanic Example **7**
With Tensorflow . . . 7
With XGBoost . . . 8

8. Conclusion **8**

1. Introduction

- Tóm tắt khóa học Advanced Learning Algorithms - Machine learning - Andrew Ng

2. Neural Network with binary classification



- Giả sử ta có bài toán với đầu vào là một dataset 1000 samples và 10 features.

- Đầu ra là probability (1 or 0 ~ True or False). Ví dụ. Is it cat ? 1 -> Cat, 0 -> not Cat.
- Với mạng neuron này ta có 3 layers: 2 hidden layers và 1 output layers.
- Trong mỗi layers có các neurons. Đầu vào của mỗi layers và output của layer trước. Đầu ra của mỗi layers được tổng hợp từ đầu ra của các neurons của chính nó.
- Công thức cơ bản: Trong một layer nếu đầu vào là X thì tại mỗi neurons sẽ tính wX +b. Hàm kích hoạt (sigmoid, relu, Tanh, softmax,...) dùng để biến đổi output của mỗi layers cho phù hợp.
- Tóm lại trong một mạng neural network ta cần định nghĩa: Layers và hàm kích hoạt. Tại mỗi layers ta cần định nghĩa neuron(unit).

Định nghĩa Neural Network with tensorflow

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Senquential(
  [
    tf.keras.Input(shape= (100,)),
    Dense(units = 5 , activation = 'relu'),
    Dense(units = 3, activation = 'relu') ,
    Dense(units = 1 , activation = 'sigmoid')
  ] ,name = 'my_model'
)
```

- Với Dense ~ Layer, units ~ neuron, activation ~ hàm kích hoạt, Senquential dùng để kết hợp các Dense. Trong bài toán binary classification đầu ra luôn là là sigmoid.

Định nghĩa Neural Network with Numpy (let's take the course :D)

Why is relu good ?

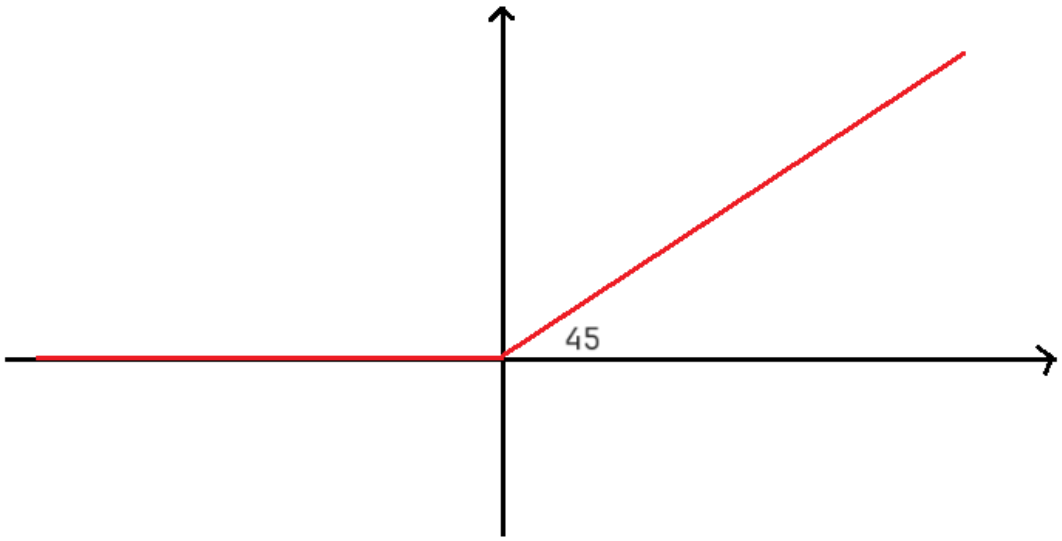


Figure 1: ReLU

- Ít tốn chi phí tính toán: ReLU(x) = max(0,x)
- Hội tụ nhanh vì có độ dốc lớn (lớn hơn sigmoid) mà vẫn đảm bảo non-linear

-> relu cho hidden layers là a good choice "_"

Training with tensorflow

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Senquential(
  [
    tf.keras.Input(shape= (100,)),
    Dense(units = 5 , activation = 'relu'),
    Dense(units = 3, activation = 'relu') ,
    Dense(units = 1 , activation = 'sigmoid')
  ] ,name = 'my_model'
)
model.compile(
    loss= tk.keras.losses.BinaryCrossentropy(),
    optimizer = tf.keras.optimizers.Adam(0.001) ,
)
model.fit(
    X_train,y_train,
    epochs = 100,
)
```

3. Neural Network with multi-class classification

- Khi ta mong muốn phân loại nhiều class. Ví dụ: handwritten digit recognition
- Khi đó hàm kích hoạt ở output layer thay vì là sigmoid bây giờ là softmax
- Sigmoid function:

$$a = \frac{1}{1 + e^{-z}}$$

- Softmax function:

$$a_m = \frac{e^{z_m}}{e^{z_1} + ... + e^{z_m} + ... + e^{z_n}}$$

- Lưu ý với softmax đầu ra là vector do đó cần dùng np.argmax để tìm ra class với probability cao nhất
- Thực chất sigmoid là một trường hợp đặc biệt của softmax khi n = 2

Định nghĩa với Tensorflow (3 classes)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Senquential(
  [
    tf.keras.Input(shape= (100,)),
    Dense(units = 5 , activation = 'relu'),
    Dense(units = 3, activation = 'relu') ,
    Dense(units = 3 , activation = 'softmax')
  ] ,name = 'my_model'
)
```

Lưu ý

- Ta có thể kết hợp softmax vào trong quá trình tính toán loss (nghĩa là tách ra khỏi forward propagation) để tăng độ chính xác (do tính chất làm tròn trong python)
- Do đó trong lúc định nghĩa Dense cho output layer. Let's set 'linear' for activation
- Use from_logits = True (is a parameter of SparseCategoricalCrossentropy)
- If so, khi dùng model.predict ->z which is not probanility -> let's deliver it through Softmax (tf.nn.softmax(z).numpy())
- Remember argmax

Training with tensorflow

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Senquential(
  [
    tf.keras.Input(shape= (100,)),
    Dense(units = 5 , activation = 'relu'),
```

```
    Dense(units = 3, activation = 'relu') ,
    Dense(units = 3 , activation = 'linear')
  ] ,name = 'my_model'
)
 model.compile(
    loss= tk.keras.losses.SparseCategoticalCrossentropy(from_logits = True),
    optimizer = tf.keras.optimizers.Adam(0.001) ,
)
 model.fit(
    X_train,y_train,
    epochs = 100,
)
```

4. How to evaluate and fine tune model

How to evaluate?

- Đầu tiên chúng ta nên cần một tập test riêng để đánh giá xem model có tốt trên dữ liệu mới (chưa train) hay không. Dùng metrics phù hợp để đo trên tập đó xem có đạt chỉ tiêu hay chưa. Nếu chưa thực hiện cải tiến các hyparameters. Ví dụ trong linear regression thì hyparameters là learning_rate, bậc của linear, lambda của regularization,....
- Nhưng khi thực hiện fine-tune hyparameters mà dựa vào test set như vậy thì không đảm bảo tính mới(unseen) khi đánh giá. Do đó phần nào chúng ta đã cung cấp thông tin cho model về dữ liệu test. Vậy nên trong test set chung ta cần chia thành dev/valid set và test set. Trong đó dev/valid set dùng để fine-tune hyparameters. Sau khi hoàn thành tất cả từ việc training và fine-tune chúng ta mới thực hiện đo trên test set và lấy số liệu đó để đánh giá model(report).

Bias and variance

When is high bias or high variance ?

$loss_t$	$loss_v$	conclusion
high	high	high bias or underfit
low	high	high variance or overfit
low	low	right model
low	high	no conclusion

Table 1: model according losstrain and lossvalid

- Khi model quá đơn giản thì sẽ dẫn tới high bias. Khi model quá phức tạp, không generalize data dẫn tới quá fit với dữ liệu train.

How to judge whether model is overfit?

- Keep track by baseline level of performance
- Keep track by learning curves

Solution

- Với high bias:
- Adding polynomial features (x1^{2}, x2^{2},x1x2).
- Giảm lambda (regularization)
- Thêm features

- Với high variance
- More data
- Tăng lambda
- Bớt features

*Note: Việc thêm regularization kết hợp mô hình phức tạp thường tốt hơn mô hình ít phức tạp mà không có regularization

"'python from Keras import regularizers Dense(units = 10 , activation = 'relu' , kernel_regularizer = regularizers.L2(0.01))

Error analysis

- Phân tích lỗi của những sample bị predict sai. Từ đó đưa ra phương pháp chỉnh sửa nó. Ví dụ trong phân loại chó mèo. Nếu mèo thường bị sai hơn thì augment data thuộc lớp mèo

Transfer learning

- Dùng neural network được train trên dữ liệu lớn để fine-tune.
- Với computer vision có các mô hình như ResNet, VGG, Inception, EfficientNet được huấn luyện trên ImageNet
- Trong NLP: BERT, GPT-3
- Xử lý âm thanh

5. Decision Tree

Tree

- Dựa vào feature để phân chia
- Chọn feature sao cho khi chọn feature đó ta đạt được information gain nhiều nhất.
- Cách tính entropy cho mỗi node:

$$H(p1) = -p1 \log_2 p1 - (1-p1) \log_2 (1-p1)$$

- Cách tính imformation gain cho mỗi feature được chọn:

$$InformationGain = H(pre_node) - wl * H(left_selectednode) - wr * H(right_selectednode)$$

- Với wl,wr = len(pre_node) / len(cur_node)

Tree Ensamble

- Use sampling with replacement technique to create various training sets
- Combine results to give the final result
- To help the model becomes stable

Random Forest

- Is tree ensamble. Nhưng chọn k < n feature để huấn luyện cây.

XGBoost

- Is tree ensamble. Nhưng cây sau được xây dựng giúp sửa lỗi của cây trước bằng cách, khi sampling with replacement ta tăng tỉ lệ chọn của các samples bị sai.

6. When should we choose decision tree or neural network ?

- Decision Tree:
  - With tabular (structured) data
  - Giúp dễ dàng diễn giải kết quả
  - may be faster
- Neural Network
  - Có thể tốt trên cả structured data or unstructured data.
  - Have transfer learning

7. Titanic Example

With Tensorflow

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import regularizers
for dirname, _, filenames in os.walk('/kaggle/input'):
   for filename in filenames:
       print(os.path.join(dirname, filename))
train_data = pd.read_csv('/kaggle/input/titanic/train.csv')
test_data = pd.read_csv('/kaggle/input/titanic/test.csv')
train_data.head()
y = train_data['Survived'].astype('float32')
features = ['Fare','Sex','Age']
X = pd.get_dummies(train_data[features])
X_test = pd.get_dummies(test_data[features])
X = X.astype('float')
X_test = X_test.astype('float')
X = X.fillna(X.mean())
X_test = X_test.fillna(X_test.mean())
model = Sequential([
   Dense(units = 3 , activation = 'relu' , kernel_regularizer = regularizers.L2(1.0)) ,
   Dense(units = 1 , activation = 'sigmoid' )
])
model.compile(loss = tf.keras.losses.BinaryCrossentropy(),
            optimizer = tf.keras.optimizers.Adam(0.005))
model.fit(X,y,epochs = 10)
predictions = model.predict(X_test)
predictions = np.ravel(predictions)
predictions = (predictions >=0.5).astype('int')
```

```
output = pd.DataFrame({'PassengerId' : test_data.PassengerId , 'Survived':predictions})
output.to_csv('submission.csv',index = False)
```

With XGBoost

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from xgboost import XGBClassifier
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
train_data = pd.read_csv('/kaggle/input/titanic/train.csv')
test_data = pd.read_csv('/kaggle/input/titanic/test.csv')
train_data.head()
y = train_data['Survived']
features = ['Fare','Sex','Age']
X_train = train_data[ features]
X_test = test_data[features]
X_train['Sex']= X_train['Sex'].astype('category').cat.codes
X_test['Sex']= X_test['Sex'].astype('category').cat.codes
bst = XGBClassifier(n_estimators=100, max_depth=4, learning_rate=28, objective='binary:logistic')
bst.fit(X_train,y)
predictions = bst.predict(X_test)
output = pd.DataFrame({'PassengerId' : test_data.PassengerId , 'Survived' : predictions})
output.to_csv('submission.csv' , index = False)
```

8. Conclusion

- 5 stars :)