PhamNgocHai_21002139_Week7_ANN_ViDu3

Pham Ngoc Hai

April 1, 2024

1 Ví dụ 1.

Phân loại dữ liệu tự tạo

1.1 Tạo dữ liệu

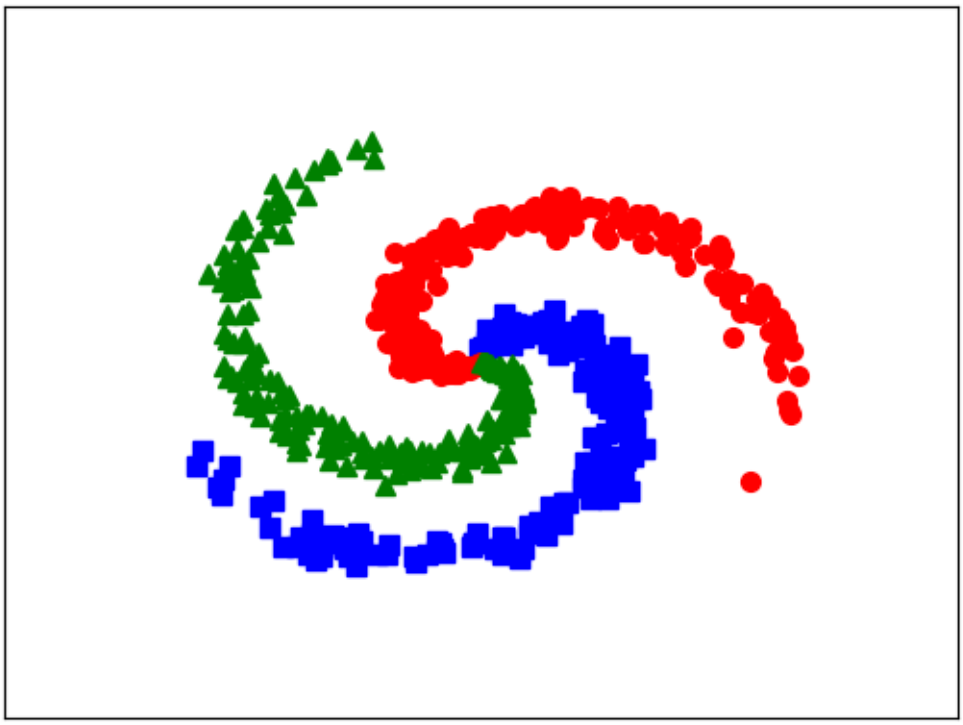
3 class, 2D sao cho không thể dùng phân loại tuyến tính

[ ]:
```
# import the necessary libraries
from __future__ import division, print_function, unicode_literals
import math
import numpy as np
import matplotlib.pyplot as plt

N = 200  # number of points per class
d = 2  # dimensionality
C = 3  # number of classes
X = np.zeros((d, N * C))  # data matrix (each row = single example)
y = np.zeros(N * C, dtype="uint8")  # class labels

for j in range(C):
    ix = range(N * j, N * (j + 1))
    r = np.linspace(0.0, 1, N)  # radius
    t = np.linspace(j * 4, (j + 1) * 4, N) + np.random.randn(N) * 0.2  # theta: + 1 đại lượng random để đảm vào phân bố ngẫu nhiên của điểm dữ liệu
    X[:, ix] = np.c_[r * np.sin(t), r * np.cos(t)].T
    y[ix] = j
```

1.2 Trực quan hóa dữ liệu

[ ]:
```
# lets visualize the data:
plt.plot(X[0, :N], X[1, :N], "bs", markersize=7)
plt.plot(X[0, N : 2 * N], X[1, N : 2 * N], "ro", markersize=7)
plt.plot(X[0, 2 * N :], X[1, 2 * N :], "g^", markersize=7)

plt.xlim([-1.5, 1.5])
plt.ylim([-1.5, 1.5])
cur_axes = plt.gca()
```

```
cur_axes.axes.get_xaxis().set_ticks([])
cur_axes.axes.get_yaxis().set_ticks([])

# plt.savefig("example1_points.png", bbox_inches="tight", dpi=600)
plt.show()
```



1.3 Code mô hình ANN không sử dụng thư viện

1.3.1 Xây dựng softmax, cost, one-hot coding

[ ]:
```
# output layer to value in [0, 1]
def softmax(V):
    e_V = np.exp(V - np.max(V, axis=0, keepdims=True))
    Z = e_V / e_V.sum(axis=0)
    return Z


# cost or loss function
def cost(Y, Yhat):
    return -np.sum(Y * np.log(Yhat)) / Y.shape[1]


# make one-hot coding
```

```
from scipy import sparse
def convert_labels(y, C=3):
    Y = sparse.coo_matrix(
        (np.ones_like(y), (y, np.arange(len(y)))), shape=(C, len(y))
    ).toarray()
    return Y
```

1.3.2 Xây dựng kiến trúc mạng

Bổ sung 01 tầng hidden với số units của tầng này là 100 (có thể thay đổi cho phù hợp với đặc điểm của dữ liệu).
Layer output là 1 vector 3 thành phần cho 3 nhãn đầu ra.
Như vậy các tầng trong ANN này là:
- Tầng input: d0 = 2 - Tầng hidden: d1 = 100 - Tầng output: d2 = 3

[ ]:
```
def build_ANN(X, y):
    d0 = 2
    d1 = h = 100 # size of hidden layer
    d2 = C = 3
    # initialize parameters randomly
    W1 = 0.01*np.random.randn(d0, d1)
    b1 = np.zeros((d1, 1))
    W2 = 0.01*np.random.randn(d1, d2)
    b2 = np.zeros((d2, 1))
    # set data
    Y = convert_labels(y, C)
    N = X.shape[1]
    eta = 1 # learning rate
    # bước lặp tìm cực trị theo Gradient Descent
    for i in range(10000):
        ## Feedforward
        Z1 = np.dot(W1.T, X) + b1
        A1 = np.maximum(Z1, 0)
        Z2 = np.dot(W2.T, A1) + b2
        Yhat = softmax(Z2)

        # print loss after each 1000 iterations
        if i %1000 == 0:
            # compute the loss: average cross-entropy loss
            loss = cost(Y, Yhat)
            print("iter %d, loss: %f" %(i, loss))

        # backpropagation
        E2 = (Yhat - Y )/N
        dW2 = np.dot(A1, E2.T)
        db2 = np.sum(E2, axis = 1, keepdims = True)
        E1 = np.dot(W2, E2)
```

```
        E1[Z1 <= 0] = 0 # gradient of ReLU
        dW1 = np.dot(X, E1.T)
        db1 = np.sum(E1, axis = 1, keepdims = True)

        # Gradient Descent update
        W1 += -eta*dW1
        b1 += -eta*db1
        W2 += -eta*dW2
        b2 += -eta*db2

    # return ANN
    return (W1, W2, b1, b2, d0, d1, d2)

def run_ANN(model, X):
    # W1, W2, b1, b2 is ANN
    # X is input & y is output
    # this function return predicted class
    W1, W2, b1, b2 = model[0], model[1], model[2], model[3]
    Z1 = np.dot(W1.T, X) + b1
    A1 = np.maximum(Z1, 0)
    Z2 = np.dot(W2.T, A1) + b2
    predicted_class = np.argmax(Z2, axis=0)
    return predicted_class
```

1.3.3 Thực hiện chạy hàm xây dựng ANN và xác định Accuracy

[ ]:
```
model = build_ANN(X, y)
predicted_class = run_ANN(model=model, X=X)
acc = 100*np.mean(predicted_class == y)
print('training accuracy: %.2f %%' % (acc))
```

```
iter 0, loss: 1.098630
iter 1000, loss: 0.139456
iter 2000, loss: 0.045205
iter 3000, loss: 0.030869
iter 4000, loss: 0.025280
iter 5000, loss: 0.022114
iter 6000, loss: 0.020025
iter 7000, loss: 0.018516
iter 8000, loss: 0.017357
iter 9000, loss: 0.016437
training accuracy: 99.67 %
```

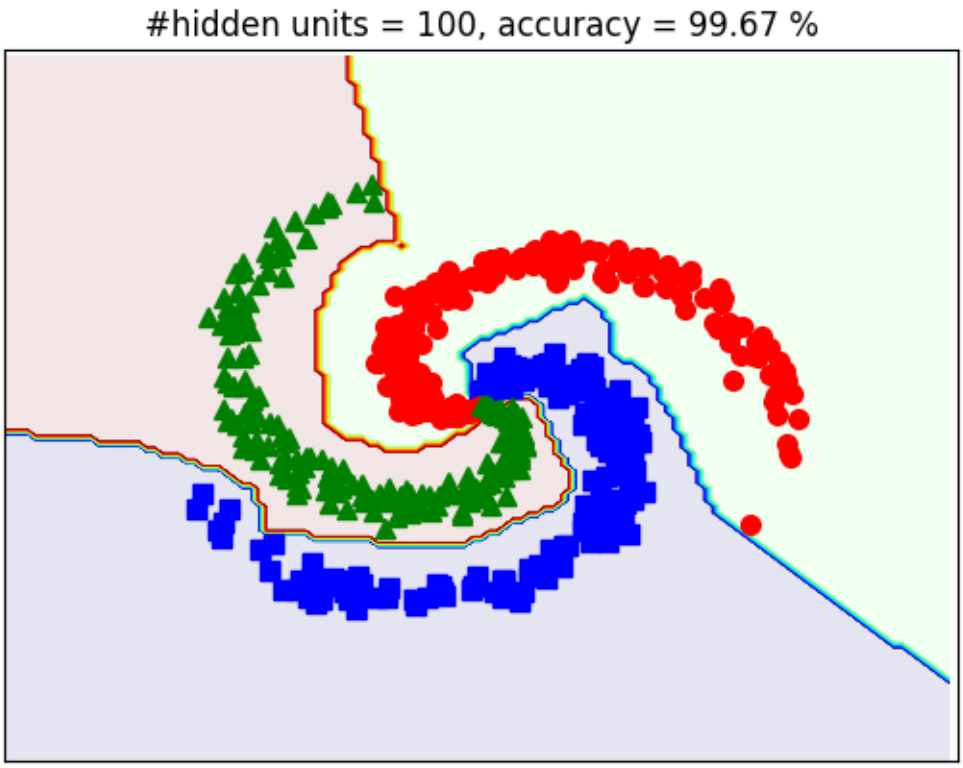
1.3.4 Trực quan hóa mô hình train được

[ ]:
```
def visualize_ANN(model=model):
    W1, W2, b1, b2, d1 = model[0], model[1], model[2], model[3], model[5]
    # Visualize results
    xm = np.arange(-1.5, 1.5, 0.025)
    xlen = len(xm)
    ym = np.arange(-1.5, 1.5, 0.025)
    ylen = len(ym)
    xx, yy = np.meshgrid(xm, ym)

    # xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,␣
 ↪h))
    # xx.ravel(), yy.ravel()

    print(np.ones((1, xx.size)).shape)
    xx1 = xx.ravel().reshape(1, xx.size)
    yy1 = yy.ravel().reshape(1, yy.size)

    # print(xx.shape, yy.shape)
    # XX = np.concatenate((np.ones((1, xx.size)), xx1, yy1), axis = 0)

    X0 = np.vstack((xx1, yy1))

    # print(X.shape)

    Z1 = np.dot(W1.T, X0) + b1
    A1 = np.maximum(Z1, 0)
    Z2 = np.dot(W2.T, A1) + b2
    # predicted class
    Z = np.argmax(Z2, axis=0)

    Z = Z.reshape(xx.shape)
    CS = plt.contourf(xx, yy, Z, 200, cmap="jet", alpha=0.1)

    # Plot also the training points
    # plt.scatter(X[:, 1], X[:, 2], c=Y, edgecolors='k', cmap=plt.cm.Paired)
    # plt.xlabel('Sepal length')
    # plt.ylabel('Sepal width')

    # X = X.T
    N = 200
    print(N)

    plt.plot(X[0, :N], X[1, :N], "bs", markersize=7)
    plt.plot(X[0, N : 2 * N], X[1, N : 2 * N], "ro", markersize=7)
    plt.plot(X[0, 2 * N :], X[1, 2 * N :], "g^", markersize=7)
```

```
    # plt.axis('off')
    plt.xlim([-1.5, 1.5])
    plt.ylim([-1.5, 1.5])
    cur_axes = plt.gca()
    cur_axes.axes.get_xaxis().set_ticks([])
    cur_axes.axes.get_yaxis().set_ticks([])

    plt.xlim(-1.5, 1.5)
    plt.ylim(-1.5, 1.5)
    plt.xticks(())
    plt.yticks(())
    plt.title("#hidden units = %d, accuracy = %.2f %%" % (d1, acc))
    # plt.axis('equal')
    # display(X[1:, :], original_label)
    fn = "ex_res" + str(d1) + ".png"
    # plt.savefig(fn, bbox_inches='tight', dpi = 600)
    plt.show()
```

[ ]:
```
visualize_ANN(model=model)
```

```
(1, 14400)
200
```



2 Bài tập ứng dụng 1.

Sử dụng ANN cho bài toán phân loại dữ liệu hoa Iris

3 Ví dụ 2.

Sử dụng Multinomial Logistic Regression và ANN để phân loại dữ liệu chữ số viết tay

4 Ví dụ 3.

(Bài tập nộp ngay trên lớp)
Sử dụng Multinomial Logistic Regression, Naive Bayes, ANN để phân loại dữ liệu khuôn mặt

4.1 Thông tin về bộ dữ liệu

Bộ dữ liệu gồm 165 ảnh của 15 người. Mỗi người chụp ở 11 trạng thái khác nhau lưu dưới dạng tệp .png.

Tên ảnh được ghép bởi:
- prefix 'subject', chỉ số của người tương ứng ghi theo kiểu 01, 02 ... đến 15 và dấu chấm "." .
- Tiếp theo là các trạng thái, gồm 11 trạng thái ['centerlight', 'glasses', 'happy', 'leftlight', 'noglasses', 'normal', 'rightlight', 'sad', 'sleepy', 'surprised', 'wink']
- Cuối cùng là phần mở rộng trong tên tệp, .png.

4.2 Đọc dữ liệu và tiền xử lý

Đọc ảnh -> Lấy ma trận điểm ảnh -> Duỗi thẳng thành vector với số chiều D = height * width

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import cv2


def read_data(path, prefix="subject", surfix=".png"):
    ids = range(1, 16)  # 15 people
    states = [
        "centerlight",
        "glasses",
        "happy",
        "leftlight",
        "noglasses",
        "normal",
        "rightlight",
        "sad",
        "sleepy",
        "surprised",
        "wink",
    ]
```

```
    # open one picture to get the image's size
    fn = prefix + "01." + states[0] + surfix
    im = cv2.imread(path + fn, 0)

    h = im.shape[0]  # hight
    w = im.shape[1]  # width

    D = h * w
    N = len(states) * 15

    X = np.zeros((D, N))

    # collect all data
    count = 0

    # there are 15 people
    for person_id in range(1, 16):
        for state in states:
            # get name of each image file
            fn = path + prefix + str(person_id).zfill(2) + "." + state + surfix

            # open the file and read as grey image
            tmp = cv2.imread(fn, cv2.IMREAD_GRAYSCALE)

            # then add image to dataset X
            X[:, count] = tmp.reshape(D)
            count += 1

    return X, (D, h, w)


# Sử dụng hàm read_data() để đọc dữ liệu
path = "data/face_data/"
result = read_data(path)
X, info_data = result[0], result[1]
print(X.shape)
```

```
(77760, 165)
```

4.3 Thực hiện giảm số chiều PCA

Giảm số chiều dữ liệu từ 77760 xuống còn 125. Nên giảm xuống còn dưới 165 do chỉ có 165 bản ghi.

[ ]:
```
def do_pca(info_data=info_data, prefix="subject", surfix=".png"):
    D, h, w = info_data[0], info_data[1], info_data[2]
```

```
# Doing PCA, note that each row is a datapoint
from sklearn.decomposition import PCA

# remain dim. k = 125 - change it!
pca = PCA(n_components=125)

# then apply to data X
pca.fit(X.T)

# then build projection matrix
U = pca.components_.T


# then reshape new dataset (reduced dim.) to be new images and save
# path to save reduced dim. images
path_save = (
    "data/visualize/"
)
for i in range(U.shape[1]):
    plt.axis("off")
    f1 = plt.imshow(U[:, i].reshape(h, w), interpolation="nearest")
    f1.axes.get_xaxis().set_visible(False)
    f1.axes.get_yaxis().set_visible(False)

    plt.gray()
    fn = path_save + "eigenface" + str(i).zfill(2) + ".png"
    plt.savefig(fn, bbox_inches="tight", pad_inches=0)

# test results by showing some images
# See reconstruction of first 6 persons
for person_id in range(1, 7):
    for state in ["centerlight"]:
        fn = path + prefix + str(person_id).zfill(2) + "." + state + surfix
        im = cv2.imread(fn, 0)
        plt.axis("off")

        f1 = plt.imshow(im, interpolation="nearest")
        f1.axes.get_xaxis().set_visible(False)
        f1.axes.get_yaxis().set_visible(False)
        plt.gray()
        fn = "ori" + str(person_id).zfill(2) + ".png"
        plt.savefig(fn, bbox_inches="tight", pad_inches=0)
        plt.show()

        # reshape and subtract mean, don't forget
        x = im.reshape(D, 1) - pca.mean_.reshape(D, 1)
```

```
            # encode
            z = U.T.dot(x)

            # decode
            x_tilde = U.dot(z) + pca.mean_.reshape(D, 1)

            # reshape to orginal dim
            im_tilde = x_tilde.reshape(h, w)
            plt.axis("off")

            f1 = plt.imshow(im_tilde, interpolation="nearest")
            f1.axes.get_xaxis().set_visible(False)
            f1.axes.get_yaxis().set_visible(False)
            plt.gray()
            # fn = 'res' + str(person_id).zfill(2) + '.png'
            # plt.savefig(fn, bbox_inches='tight', pad_inches=0)
            plt.show()
```

[ ]:
```
do_pca(info_data)
```

Viết lại hàm do_pca do quên không return giá trị X đã giảm chiều. Code ở trên vẫn giữ như là 1 cách lưu lại ảnh X đã giảm chiều để nhìn.

[ ]:
```
def do_pca(data=X):
    # Doing PCA, note that each row is a datapoint
    from sklearn.decomposition import PCA

    # Khởi tạo mô hình PCA với số thành phần chính mong muốn
    pca = PCA(n_components=125) # 125 or bất kỳ số nào nhỏ hơn 165 - số lượng␣
 ↪bản ghi

    # Huấn luyện mô hình PCA trên dữ liệu
    pca.fit(data.T)

    # Áp dụng phép biến đổi PCA để giảm số chiều của dữ liệu
    data_reduced = pca.transform(data.T)

    return data_reduced
```

[ ]:
```
X_train = do_pca(X)
```

[ ]:
```
X_train.shape
```

[ ]:
```
(165, 125)
```

4.4 Sử dụng mô hình phân loại

Sử dụng các phương pháp phân loại nhiều lớp:
- Multinomial Logistic Regression
- Naive Bayes phù hợp
- ANN (đã có code)
để phân loại, tỷ lệ train:test là 0.7:0.3.

4.4.1 Phân chia train:test theo 7:3

[ ]:
```
from sklearn.model_selection import train_test_split

states = [
    "centerlight",
    "glasses",
    "happy",
    "leftlight",
    "noglasses",
    "normal",
    "rightlight",
    "sad",
    "sleepy",
    "surprised",
    "wink",
]

X_train, X_test, y_train, y_test = train_test_split(
    X_train,
    np.repeat(np.arange(1, 16), len(states)),
    test_size=0.3,
    random_state=42,
)
```

[ ]:
```
print(X_train.shape, y_train.shape)
```

```
(115, 125) (115,)
```

[ ]:
```
print(X_test.shape, y_test.shape)
```

```
(50, 125) (50,)
```

4.4.2 Thực hiện sử dụng Multinomial Logistic Regression

[ ]:
```
from sklearn.linear_model import LogisticRegression

# Multinomial Logistic Regression
logistic_regression_model = LogisticRegression(
    max_iter=10000, multi_class="multinomial"
)
```

```
logistic_regression_model.fit(X_train, y_train)
logistic_regression_predictions = logistic_regression_model.predict(X_test)
```

[ ]:
```
from sklearn.metrics import accuracy_score

logistic_regression_accuracy = accuracy_score(
    y_test, logistic_regression_predictions
)
print("Multinomial Logistic Regression Accuracy:", logistic_regression_accuracy)
```

```
Multinomial Logistic Regression Accuracy: 1.0
```

[ ]:
```
from sklearn.metrics import confusion_matrix

# Tính ma trận nhầm lẫn
conf_matrix = confusion_matrix(y_test, logistic_regression_predictions)

# In ra ma trận nhầm lẫn
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 5 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 6 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 2 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 3 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 4 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 4 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 4 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 2 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 6 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 2 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 4]]
```

4.4.3 Thực hiện sử dụng Naive Bayes

[ ]:
```
from sklearn.naive_bayes import GaussianNB

# Naive Bayes Classifier
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)
naive_bayes_predictions = naive_bayes_model.predict(X_test)
```

[ ]:
```
from sklearn.metrics import accuracy_score
naive_bayes_accuracy = accuracy_score(y_test, naive_bayes_predictions)
print("Naive Bayes Accuracy:", naive_bayes_accuracy)
```

```
Naive Bayes Accuracy: 0.72
```

[ ]:
```
from sklearn.metrics import confusion_matrix

# Tính ma trận nhầm lẫn
conf_matrix = confusion_matrix(y_test, naive_bayes_predictions)

# In ra ma trận nhầm lẫn
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 5 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 3 0 0 0 0 1 1 0 0 0 0 0 1]
 [0 0 0 1 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 2 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 1 0 0 1]
 [0 0 0 0 0 0 0 2 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 0 4 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 2 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 1 4 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 2 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 4]]
```

4.4.4 Thực hiện sử dụng ANN

Các hàm xây dựng

[ ]:
```
# output layer to value in [0, 1]
def softmax(V):
    e_V = np.exp(V - np.max(V, axis=0, keepdims=True))
    Z = e_V / e_V.sum(axis=0)
    return Z


# cost or loss function
def cost(Y, Yhat):
    return -np.sum(Y * np.log(Yhat)) / Y.shape[1]


# make one-hot coding
```

```
from scipy import sparse
def convert_labels(y, C):
    Y = sparse.coo_matrix(
        (np.ones_like(y), (y, np.arange(len(y)))), shape=(C, len(y))
    ).toarray()
    return Y
```

[ ]:
```
def build_ANN(X, y):
    d0 = 125
    d1 = h = 100 # size of hidden layer
    d2 = C = 15
    # initialize parameters randomly
    W1 = 0.01*np.random.randn(d0, d1)
    b1 = np.zeros((d1, 1))
    W2 = 0.01*np.random.randn(d1, d2)
    b2 = np.zeros((d2, 1))
    # set data
    Y = convert_labels(y, C)
    N = X.shape[1]
    eta = 1 # learning rate
    # bước lặp tìm cực trị theo Gradient Descent
    for i in range(10000):
        ## Feedforward
        Z1 = np.dot(W1.T, X) + b1
        A1 = np.maximum(Z1, 0)
        Z2 = np.dot(W2.T, A1) + b2
        Yhat = softmax(Z2)

        # print loss after each 1000 iterations
        if i %1000 == 0:
            # compute the loss: average cross-entropy loss
            loss = cost(Y, Yhat)
            print("iter %d, loss: %f" %(i, loss))

        # backpropagation
        E2 = (Yhat - Y )/N
        dW2 = np.dot(A1, E2.T)
        db2 = np.sum(E2, axis = 1, keepdims = True)
        E1 = np.dot(W2, E2)
        E1[Z1 <= 0] = 0 # gradient of ReLU
        dW1 = np.dot(X, E1.T)
        db1 = np.sum(E1, axis = 1, keepdims = True)

        # Gradient Descent update
        W1 += -eta*dW1
        b1 += -eta*db1
        W2 += -eta*dW2
```

```
        b2 += -eta*db2

    # return ANN
    return (W1, W2, b1, b2, d0, d1, d2)

def run_ANN(model, X):
    # W1, W2, b1, b2 is ANN
    # X is input & y is output
    # this function return predicted class
    W1, W2, b1, b2 = model[0], model[1], model[2], model[3]
    Z1 = np.dot(W1.T, X) + b1
    A1 = np.maximum(Z1, 0)
    Z2 = np.dot(W2.T, A1) + b2
    predicted_class = np.argmax(Z2, axis=0)
    return predicted_class
```

[ ]:
```
print(set(y_test - 1))
print(set(y_train - 1))
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}
```

Xây dựng model

[ ]:
```
ANN_model = build_ANN(X_train.T, y_train - 1)
predicted_class = run_ANN(ANN_model, X_test.T)
acc = 100*np.mean(predicted_class == y_test - 1)
print('training accuracy: %.2f %%' % (acc))
```

```
iter 0, loss: 26.867507

/tmp/ipykernel_13087/2521560992.py:3: RuntimeWarning: invalid value encountered
in subtract
  e_V = np.exp(V - np.max(V, axis=0, keepdims=True))

iter 1000, loss: nan
iter 2000, loss: nan
iter 3000, loss: nan
iter 4000, loss: nan
iter 5000, loss: nan
iter 6000, loss: nan
iter 7000, loss: nan
iter 8000, loss: nan
iter 9000, loss: nan
training accuracy: 4.00 %
```

4.5 Test với 5 ảnh bất kỳ

[ ]:
```
import cv2
import numpy as np

five_image_test = []

# Đọc 5 ảnh và chuyển về kích thước (320, 243) và định dạng ảnh xám
for i in range(1, 6):
    image = cv2.imread(f'data/test{i}.jpg', cv2.IMREAD_GRAYSCALE)
    if image is not None:
        # Resize ảnh về kích thước (320, 243)
        resized_image = cv2.resize(image, (243, 320))

        # Chuyển đổi sang dạng màu thuộc [0, 1]
        normalized_image = resized_image.astype(np.float32) / 255.0

        # Thêm ảnh đã xử lý vào danh sách
        five_image_test.append(normalized_image)
    else:
        print(f"Không thể đọc ảnh data/test{i}.jpg")
```

[ ]:
```
len(five_image_test)
```

[ ]: 5

[ ]:
```
logistic_regression_predictions = logistic_regression_model.
↪predict(five_image_test)
```

5 Ví dụ 4.

Sử dụng Multinomial Logistic Regression, Naive Bayes, ANN để phân loại dữ liệu 7 loại đậu