ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
THỰC HÀNH NHẬP MÔN CÔNG NGHỆ THÔNG TIN

NHẬN DẠNG CHỮ SỐ VIẾT TAY

SINH VIÊN THỰC HIỆN:

ĐỖ TRUNG HIẾU - 20120007
NGUYỄN VĂN HƯNG - 20120009
NGUYỄN HOÀNG HUY - 20120011
LƯU NGUYỄN TIẾN ANH - 20120245
PHẠM HỮU PHÚC - 20120351

Tháng 1, 2021

1 Phương pháp thực hiện

Chương trình nhận dạng chữ số viết tay dùng bộ dữ liệu MNIST Database. Sử dụng 3 phương pháp rút trích đặc trưng là: Véc-tơ hóa, sampling và histogram. Các phương pháp train là: KNN và mẫu trung bình.

2 Mã nguồn

2.1 Nạp dữ liệu

Hàm load_mnist nhận tham số path là đường dẫn, kind = 'train' nếu cần load tập dữ liệu train hoặc kind = 'test' nếu cần load tập dữ liệu test.

```
def load_mnist(path, kind='train'):

    """Load MNIST data from 'path'"""

    labels_path = os.path.join(path,
                  '%s-labels-idx1-ubyte.gz' %kind)

    images_path = os.path.join(path,
                  '%s-images-idx3-ubyte.gz' %kind)

    with gzip.open(labels_path, 'rb') as lbpath:
        lbpath.read(8)
        buffer = lbpath.read()
        labels = np.frombuffer(buffer, dtype=np.uint8)

    with gzip.open(images_path, 'rb') as imgpath:
        imgpath.read(16)
        buffer = imgpath.read()
        images = np.frombuffer(buffer, dtype=np.uint8).reshape(
                 len(labels), 28, 28).astype(np.float64)

    return images, labels
```

2.2 Các hàm thực hiện rút trích đặc trưng

- Véc-tơ hóa một mảng 2 chiều a[m][n] thành mảng một chiều av[m * n] trong đó av[i * n + j] = a[i][j] ($0 \leq i < m$, $0 \leq j < n$).

- Ý tưởng của phương pháp sampling như sau: Giả sử ta có một bức ảnh biểu diễn dưới dạng ma trận 28×28, ta chia đều chiều dài và chiều rộng thành từng phần ví dụ như 2×2. Với mỗi phần 2×2 đó, ta tính trung bình tổng

các giá trị, hoặc min, hoặc max. Như vậy, ta tạo được một ma trận mới là 14×14, sau đó ta tiến hành véc-tơ hóa ma trận vừa tạo.

- Histogram là một phương pháp rút trích đặc trưng của ảnh. Giả sử ảnh được chia thành n×n (ở đây là 28×28) điểm, và mỗi điểm có giá trị màu từ 0-255 thì với rút trích đặc trưng bằng histogram, chúng ta sẽ lưu dưới dạng vector gồm 256 phần tử từ 0 đến 255 là tần suất xuất hiện của các giá trị màu (0-255) có trong ảnh.

```
def vector_hoa(X):
    return X[:].reshape(-1, X[0].size)

def sampling(X):
    t = X.shape[0];
    n = range(t);
    m = range(0,28,2)
    Xt = np.zeros((t,14,14), dtype=int)
    for k in n:
        for i in m:
            for j in m:
                Xt[k][int(i/2)][int(j/2)] = (X[k][i][j] +
                                             X[k][i+1][j] +
                                             X[k][i][j+1] +
                                             X[k][i+1][j+1])/4
    return vector_hoa(Xt)

def histogram(X):
    t = X.shape[0];
    n = range(t);
    m = range(28)
    Xh = np.zeros((t,256), dtype=int)
    for k in n:
        for i in m:
            for j in m:
                Xh[k][int(X[k][i][j])]+=1
    return Xh
```

2.3 Các hàm xử lý tính toán và phương pháp train

- Lấy mẫu trung bình:
  - Bước 1: Từ kết quả rút trích đặc trưng, chuẩn bị tập mẫu trung bình bằng cách lấy trung bình các kết quả của từng tập. (Ở đây là tập ảnh có nhãn lần lượt là 0, 1, ..., 9).

– Bước 2: Với mỗi ảnh từ tập test, ta so sánh đến từng phần tử trong tập mẫu trung bình, và lấy nhãn của mẫu có "kết quả gần nhất".

- KNN: là phương pháp dự đoán nhãn của một loại dữ liệu dựa trên nhãn của một số dữ liệu có khoảng cách gần nó nhất. Từ những véc-tơ đã rút trích đặc trưng, ta dùng công thức Euclid để tính khoảng cách từ nó đến các véc-tơ còn lại trong tập train để tìm ra k véc-tơ có độ dài gần nhất với véc-tơ đang xét rồi gán nhãn cho nó.

```
def MPL(X, y):
    d = np.zeros(10)
    x = np.zeros((10, X[0].size), dtype=int)
    n = range(X.shape[0])
    for k in n:
        x[y[k]] = x[y[k]] + X[k]
        d[y[k]] += 1

    m = range(x[0].size)
    for k in range(10):
        for i in m:
            x[k][i] = round(x[k][i] / d[k])
    return x

def suydoan(X, x):
    y = np.zeros(X.shape[0])
    n = range(X.shape[0])
    from scipy.spatial import distance

    for k in n:
        t = 0
        nn = distance.euclidean(X[k], x[0])
        for i in range(10):
            tam = distance.euclidean(X[k], x[i])
            if tam < nn:
                t = i
                nn = tam
        y[k] = t
    return y

def tinh(X, x, Y, y):
    tam1 = 0
    for i in range(1,10,2):
        model = KNeighborsClassifier(n_neighbors=i)
        model.fit(X, Y)
        y_pred = model.predict(x)
        tam = 100 * accuracy_score(y, y_pred)
        if tam1 < tam:
            tam1 = tam
```

```
    cls = MPL(X, Y)
    y_pred = suydoan(x, cls)

    tam2 = 100*accuracy_score(y, y_pred)
    return tam1, tam2
```

3 Kết quả chạy chương trình

```
[INFO] Dang nap du lieu...
[INFO] Nap du lieu hoan tat! Rows: 60000, columns: 28
[INFO] Dang vecto hoa...
[INFO] Vecto hoa hoan tat
[INFO] Dang thuc hien Sampling...
[INFO] Sampling hoan tat
[INFO] Dang thuc hien Histogram...
[INFO] Histogram hoan tat
[INFO] Dang tinh toan...
[INFO] Tinh toan hoan tat
               vector       sampling      histogram
KNN            97.05%         97.42%         32.71%
Mau phan lop   82.05%         81.77%         26.14%
>>>
```

Hình 1: Kết quả chạy chương trình

Ta thấy rằng, với phương pháp train KNN kết quả nhận dạng cao nhất khi ta thực hiện cùng sampling với 97.42%. Với phương pháp mẫu phân lớp thì kết quả nhận dạng cao nhất khi ta thực hiện cùng véc-tơ hóa với 82.05%.