EE595

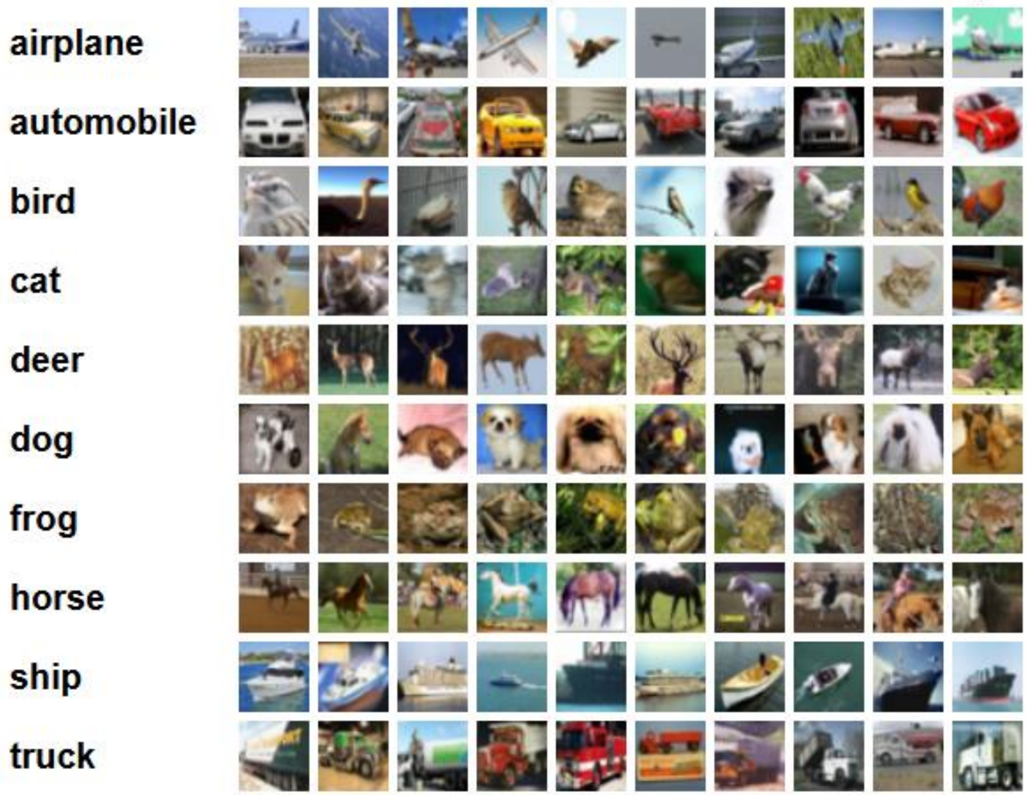
Lab 7 KNN, Naïve Bayes

Pengmiao Zhang

1. Image Classification with PCA and KNN

- ## 1. Dataset: CIFAR-10

Here are the classes in the dataset, as well as 10 random images from each:



Total 5 batches, total 50000 images for training and 10000 images for testing.

We use first 1000 images from the data batch 1

1. Image Classification with PCA and KNN

- 1. Dataset: CIFAR-10
- CIFAR-10 Website:

Dict Type

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

An image:

red 32*32
green 32*32
blue 32*32

•data -- a 10000x3072 numpy array of uint8s. Each row of the array stores a 32x32 color image:

The first 1024 red channel values, the next 1024 the green, the final 1024 the blue.

The image is stored in row-major order, so that the first 32 entries of the array are the red channel values of the first row of the image.

•labels -- numbers in the range 0-9.

1. Image Classification with PCA and KNN

- **2. Dimension reduction:**
- **2.1 Convert the RGB images to grayscale with the following formula manually:**

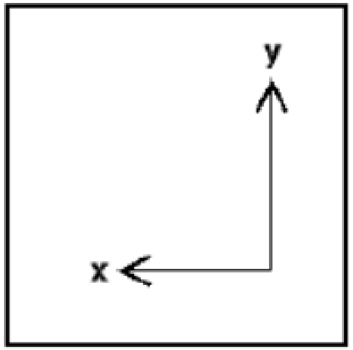
$$Y = 0.299R + 0.587G + 0.114B$$

- **2.2 Use PCA to reduce dimensions.**
  - **Compute the PCA transformation by using only the training set with the sklearn PCA package**
  - **Perform dimensionality reduction on both training and testing sets to reduce the dimension of data from 1024 to D.**
  - **full SVD solver**

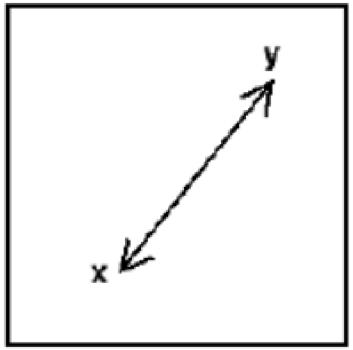
1. Image Classification with PCA and KNN

- **3. KNN Classifier from scratch:**
  - **3.1 use features obtained by PCA.**
  - **3.2 use Manhattan distance**
  - **3.3 K is specified as an input argument**
  - **3.4 you are NOT allowed to use any library providing KNN classifier (e.g. sklearn)**

$$d = \sum_{i=1}^{n} |x_i - y_i|$$



Manhattan



Euclidean

1. Image Classification with PCA and KNN

- **4. KNN Classifier with sklearn:**

- **Use sklearn package to do the KNN part again.**
- **Compare the results to verify your implementation.**

1. Image Classification with PCA and KNN

- **5. Program script and output**
- **5.1 script:**
- python knn_<student_id>.py K D N PATH_TO_DATA
- **5.2 output:**
  - predicted label and a ground truth label,separated by a single space
  - prediction accuracy
  - **two files**
    - knn_results_sklearn.txt
    - knn_results.txt

```
1 2
3 3
4 7
8 1
1 6
…
…
0.2
```

2. Naïve Bayes

- ## 1. Dataset

UCI Seeds dataset
7 features, 3 types

15.26	14.84	0.871	5.763	3.312	2.221	5.22	1
14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
14.29	14.09	0.905	5.291	3.337	2.699	4.825	1
13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1
14.38	14.21	0.8951	5.386	3.312	2.462	4.956	1
14.69	14.49	0.8799	5.563	3.259	3.586	5.219	1

Attribute Information:

To construct the data, seven geometric parameters of wheat kernels were measured:
1. area A,
2. perimeter P,
3. compactness C = 4*pi*A/P^2,
4. length of kernel,
5. width of kernel,
6. asymmetry coefficient
7. length of kernel groove.

All of these parameters were real-valued continuous.

txt, space seperation
pandas.read_table: dataframe

2. Naïve Bayes

- ## 2. Naïve Bayes from Scratch

$$h_{MAP} \equiv \underset{h \in H}{\operatorname{argmax}}\, P(h \mid D)$$

$$= \underset{h \in H}{\operatorname{argmax}} \frac{P(D \mid h) P(h)}{P(D)}$$

$$= \underset{h \in H}{\operatorname{argmax}}\, P(D \mid h) P(h)$$

2. Naïve Bayes

- **3. Algorithm**
- **1) log likelihood:** $\log p(\mathbf{X}|Y) = \sum_{i=1}^{n} \log p(X_i|Y)$

$$p(X_i = x_i | Y = y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(x_i - \mu_i)^2}{\sigma^2}$$

conditional mean μ_i and variance σ_i^2 for each feature

- **2) Estimate prior distribution** $p(Y)$
- **3) In test, log-likelihood**

$$\hat{y} = \underset{y}{\operatorname{argmax}} \log p(\mathbf{X} = \mathbf{x}, Y = y)$$

2. Naïve Bayes

- ## 4. Comparison with the Implementation in Sklearn

  sklearn.naive_bayes.GaussianNB

- ## 5. Playing with Other Classifiers

  SVM, Decision Trees, Random Forests, Multi-layer Perceptron (MLP) classifier

2. Naïve Bayes

- ## 6. Script

python naive_bayes_<STUDENT_ID>.py

- ## 5. Output

No output generation, print information to console in this format:

```
My Naive Bayes:
Training acc: XX.XX% Training time: XXXX s
Testing acc: XX.XX% Testing time: XXXX s
Sklearn Naive Bayes:
Training acc: XX.XX% Training time: XXXX s
Testing acc: XX.XX% Testing time: XXXX s
<Other Classifier>:
Training acc: XX.XX% Training time: XXXX s
Testing acc: XX.XX% Testing time: XXXX s
```

3. Extra Credit

- 1. linear SVM
  - Generate your own dataset randomly
  - LIBSVM
- 2. Implementing Multilayer Perceptron
  - MNIST
  - pytorch