503 Final Tmp

15 April, 2018

Problem of Interest

Images have played a great part of our life after dat storage become cheaper and convenient in the 21st century. In 2001, Google enable users to search image with a 250 millions of images, and the data base has grown exponentially ever since its launch. Within a decade, Reverse image search[content-based image retrieval(CBIR)] was introduced. The importance of image recognition and automated image classification has become popular over time. Even though the concept of neural network was first introduced back in 1940s, computors was not available nor able to undertake massive calculation when it was first introduced, but nowadays we are able to compute complicated models and algorithmns such as Convolutional Neural Network(CNN), Deep Neural Network(DNN) ,... etc. Popular algorithmns were introduced to solve the image classification problems.

Data Exploration Challenge with Image data/ Difference with tabular data

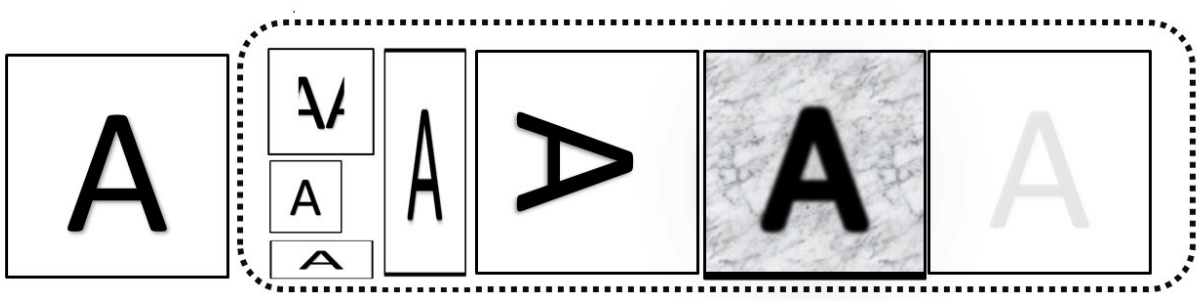
https://en.wikipedia.org/wiki/Channel_(digital_image).

Data Structure

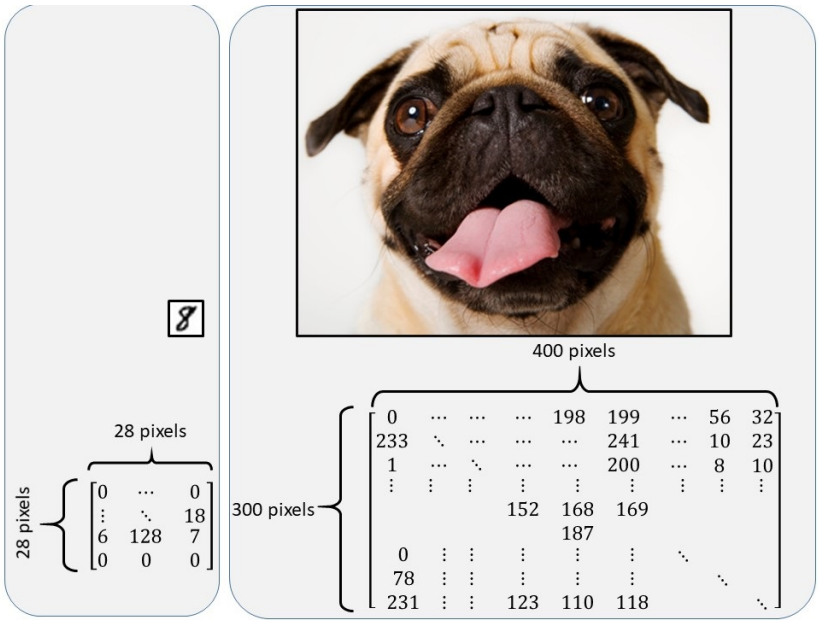
A color digital image can be decompose into an 3-dimensional array with each a n by m matrix *(or a nm length vector)*, which each matrix *(vector)* correspond to a specific channel **R,G,and B**.

Properties

The following figure displays variation of letter A, which in essence is still regonizable for human, be in the matrix representation is quiet different.



- Ordered/ Not exchangable: Even the data storage is tabular, but the columns are not exchangable, and pixels cells are highly correlated with its neighbors. Another problem arised when comverting vector back to original image, we need to be careful of the dimensions otherwise the image will be distorted.
- Rotation: Rotaion of an image may cause classifier to fail if not handle carefully, casue machine takes the matrix/vector representation of an image and and mention before, pixel are highly correlated and not exchangable.
- Noise: If the object is blend-in/ merge into the background capturing the main object image will be challenging.
- Scale: Size of the image grows in a non-linear rate, which results in the high dimensionality of the image. As an example, an image with 30x40 pixels ($\mathcal{R}^{1200}$) contains 1/100 entries compared with 300x400 pixels image ($\mathcal{R}^{120000}$).



Naïve Classifier - Nearest Neighbor Classifier

Introduction

First, we start with a naïve classifier which only consider the distance of the two matrices, the prediction of the test data is solely decided by the closet neighbor it can find among the training data space, we picked **L-1 norm** and **L-2 norm** for the disatnce calculation process.

- $$||M||_1 = max_{1 \leq j \leq m} \sum_{i=1}^{m} |m_{ij}|$$

Which is the maximum absolute column sum of the matrix.

- $$||M||_2 = \sigma_{max}(M) \leq \left(\sum_{i=1}^{n} \sum_{j=1}^{m} |m_{ij}|^2 \right)^{1/2}$$

  where $\sigma_{max}(M)$ is the maximun singular value of matrix M.

Algorithm

Suppose we have a test image, say M_{test} and set of training imgaes $M_{train_1}, M_{train_2}, \cdots, M_{train_n}$ with known class labels

- First calculate the matrix difference of the test image M_{test} with **ALL** training images $M_{train_1}, M_{train_2}, \cdots, M_{train_n}$, the difference of any two matrices is definded as the element-wise substraction, i.e.

$$A_{m\times n} - B_{m\times n} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} - \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix}$$
$$= \begin{bmatrix} a_{11} - b_{11} & \cdots & a_{1n} - b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} - b_{m1} & \cdots & a_{mn} - b_{mn} \end{bmatrix} = M_{m\times n}$$

- Second, we calculate the choose of *L-1 norm/L-2 norm* distance of the matrix from previous step.
- The prediction is made by comparing all *L-1/L-2* distances of the target test data to be classified, such that the predicted class is the one with minimun distance, i,e

$$\hat{Y_{Test}} = \left\{ N \in \{Class\ of\ ||M_i||_p\}; ||M_i||_p = min\left\{||M_1||_p, \cdots, ||M_m||_p\right\}\right\}, \text{where } p \text{ denotes } L_p \text{distance}$$

Algorithm 1: Nearest Neighbor Algorithm

Data: $X_{Train}, X_{Test}, Y_{Train}$
Result: Classification prediction forX_{Test}, $\hat{Y_{Test}}$
begin
  foreach X_{Test} **do**
    foreach X_{Train} **do**
      tmp ← Calculate difference of the two matrix(Element-wise)
      D[j] ← Calculate L_p-distance(tmp)
    Find idx_{Train} *s.t.* D[idx_{Train}] = $\underset{j}{\text{argmin}}$ D[i,j]
    $\hat{Y_{Test}}$[i] ⟵ $Y_{Train}[idx_{Train}]$
    Remove (D, idx_{Train}), release memory space
  return $\hat{Y_{Test}}$

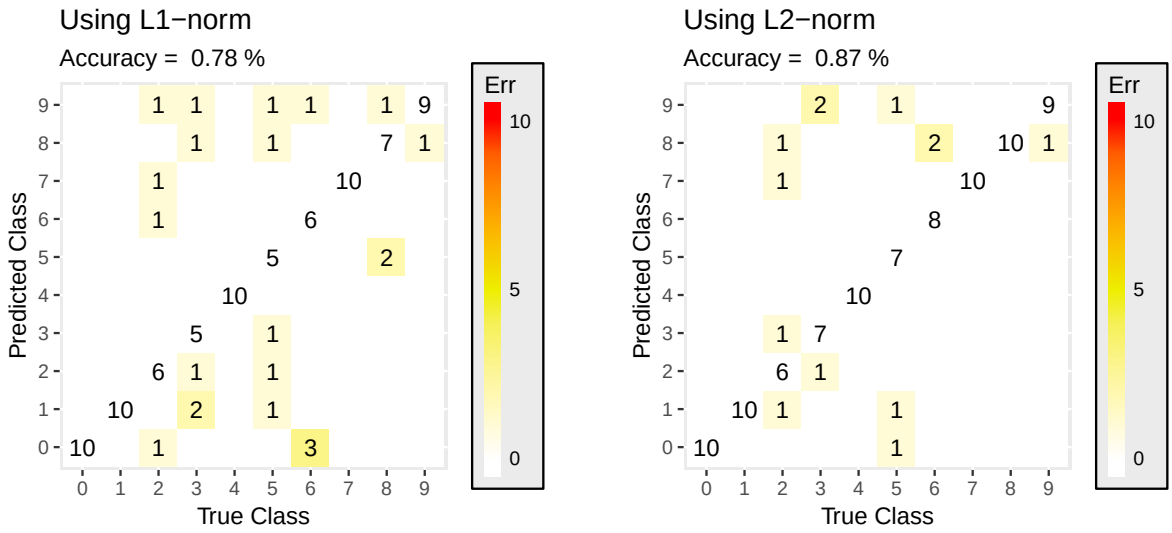
Data experiement

From the MNIST hand writting data set, we selected: + Training Set: First 100 observations per class (0-9), total of 1000 observations. + Test Set: First 10 observations per class (0-9), total of 100 observations.

QUESTION: where to deliver high- low -res problem

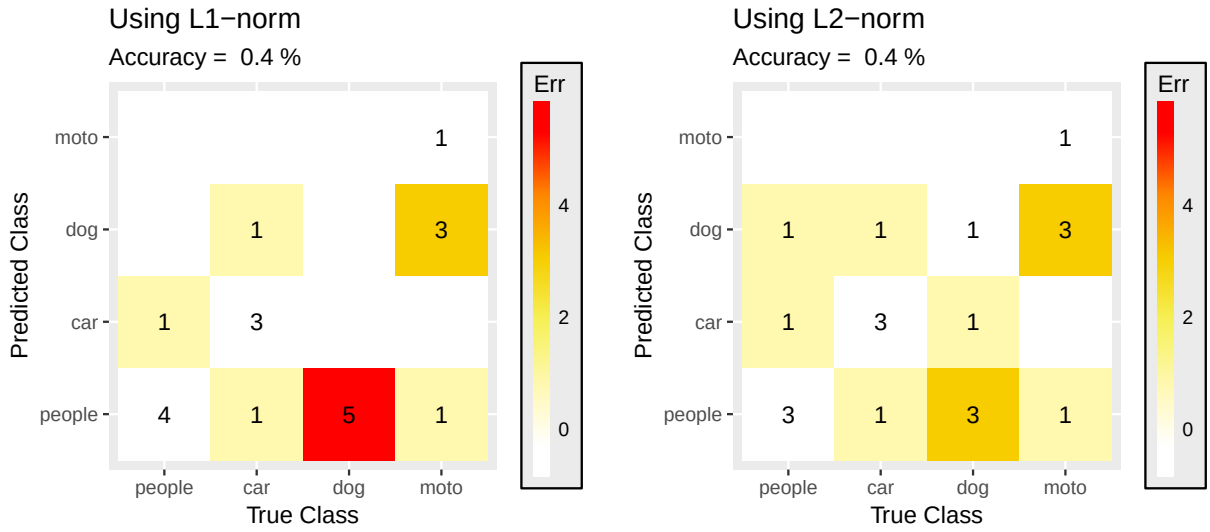
The result from the test data (consist of 100 observations per class):

Heatmap of classification matrix with *Low−resolution(28*28)* images



Notes: Err denotes the misclassified count frequency.

Heatmap of classification matrix with *High−resolution(400*300)* images



Notes: Err denotes the misclassified count frequency.

L1 vs. L2.

As we can seen that $L_2 - norm$ takes more information from the matrix as it penalizes among all the differences instead of the algorithm for $L_1 - norm$ which only takes one of all columns.

High- vc. Low- resolution images

The main difference between low- and high- resolution images are the information contained the the image vector. Results from applying naïve classifier to both high- and low- resolution images are what we expected to be quiet different. Even we only have fewer classes in the high-resolution set, the accuracy drop drastically.

Notes: Worth mentioning that this algorithm requires *ALL* images from train and test set to be consistant in width and height, ortherwise it will fail becasue the nature of the algorithm involves matrix element-wise

substraction and hence calculate matrix-norm.

Introducing Feature Descriptors

TODO: Describe what a feature detector is.

Histogram of Gradient (HOG)

TODO: Explain histogram of gradient TODO: Discuss HOG on low- and high- resolution data set.

PCA of Histogtam of gradient(HOG)

TODO: Explain model

PCA is a widely used to conquer high dimensional data issue, the algorithm tries to capture most variance of the original data set and projected into an orthogonal K dimension with selection of $K - dimensions$. The technique comes handy in image classification problems, of which the preprocessed data can be represented as a sparse matrix in high dimension space.

For each class j in training set

The algorithm first compute the HOG feature from the original image, as we try to capture the relationship among the neighbor pixels, the result from HOG feature for each image will be a vector with each element corresponds to the gradient of the sliding window.

Secondly, PCA is implemented for dimension reduction purposes, since HOG features tends to be sparse in high dimension, we performed PC on the **centered HOG feature** HOG feature and retrieve the first K eigenvectors denote $T_K^{(j)}$ (the first K PC loadings for class j).

For test classfication

First apply HOG on the test image, say h, and for each possible class:

+ Project the difference of the test HOG and average HOG feature,$\bar{h_j}$ onto PC space per class, say z_j
+ Inverse project back to the original HOG dimension and **add back the average HOG feature**, $\bar{h_j}$, we well get the approximation HOG per class, $\hat{h_j}$
+ Calculate Euclidean distance of the approximation HOG feature per class,$\hat{h_j}$ with HOG of the test image , h.

The class j with minimun euclidean distance is returned as predicted class.

TODO: pseudo code

Algorithm 2: Multi-category Classification using HOG + PCA

Input: K,Number of principle components to retained
Data: $X^{(j)}_{Train}, X_{Test}, Y_{Train}$, where $j = 0, 1, ..., \#Class$
Result: Classification prediction forX_{Test}, $\hat{Y_{Test}}$
begin
  foreach j **do**
    Step 1 : HOG feature
    Transform the raw data $X^{(j)}_{Train}$ into HOG feature matrix $H^{(j)} = [h^{(j)}_1 h^{(j)}_2 \cdots h^{(j)}_{n_j}]$
    Step 2 : Apply PCA
    Compute following: (Standardization procedure)
    - $\bar{h_j} = \frac{1}{n_j} \sum_{i=1}^{n_j} h^{(j)}_i$
    - $C_j = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} (h^{(j)}_i - \bar{h_j})(h^{(j)}_i - \bar{h_j})^T$

    Compute first K eigenvectors from C_j
    $\Longrightarrow T^{(j)}_K = [t^{(j)}_1 t^{(j)}_2 \cdots t^{(j)}_K]$
  Step 3 : Test data classification
  foreach $x \in X_{Test}$ **do**
    Compute HOG feature h of x
    foreach j **do**
      Compute
      - projection $z_j = T^{(j)}_K (h - \bar{h_j})$
      - approximation $\hat{h_j} = T^{(j)}_K z_j + \bar{h_j}$
      - Euclidean distance $E_j = ||h - \hat{h_j}||_2$

    Find $c \in j$ s.t. $E_c = \text{argmin} E_j$
    Append c to $\hat{Y_{Test}}$
  return $\hat{Y_{Test}}$

TODO: low/high res classification matrix

TODO: Interpret result

Bag of Feature

TODO: Flow chart

Image credit (BOF Flow Chart): Image adapted from Arénaire project, Laboratoire de l'Informatique du Parallélisme(LIP)

TODO: Explain model using flow chart

Bag of feature(BoF) is an implementation of bag of words from text mining in image classification. Instead of caputuring common words and build up character/ string code book, BoF classification is based on feature descriptors, of which there are various source of tecnique such as SIFT, HOG implemented in this report or can be replaced to PCA of an image,... etc. When a new test image needs to be classified, it counts the feauture of the image and **looks up** the code up and counts the frequencies of the feature descriptors, and feed the information of counts into classifier for classification.

- **Step (A)**:

First we scan over the image and pick region of area in interest for feature

- **Step (B)**:

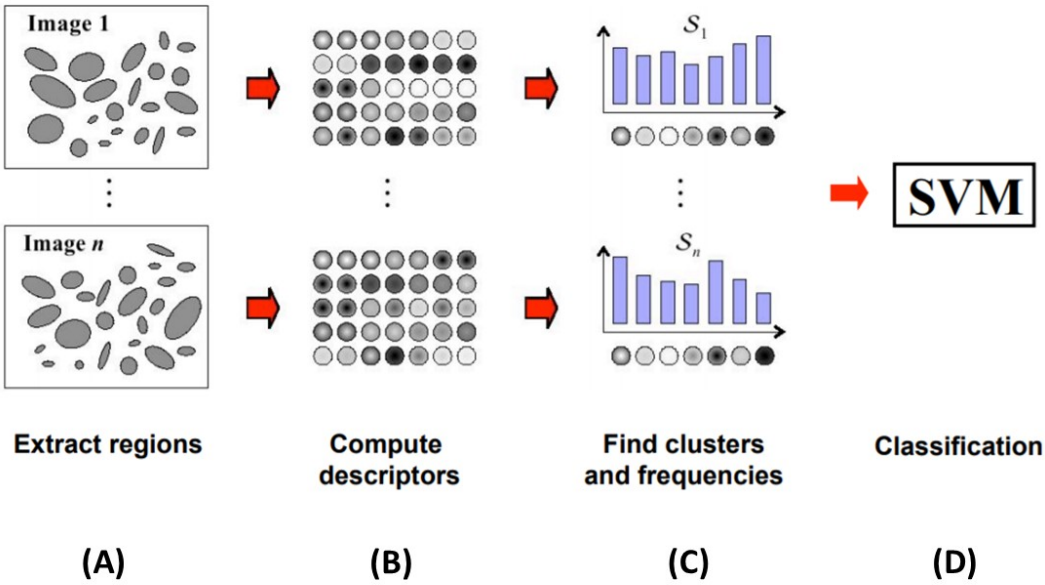


Figure 1: Bag Of Features: Flow Chart

For each image, we may find n_i desciptors, we collect all feature descriptors among all i images and using cluster technique(K-mean in the report) to label descriptors in to selected N clusters.

- **Step (C)**:

For each train data, we map the extracted n_i descriptors into N categories by following the cluster result and count the frequencies of each categories, the process will reduce the image into a vector of size N containing counts of categories, which can be read as a representive profile of an image.

- **Step (D)**:

For test images, we perform the same algorithm as in **Step A** and **Step C** so that the test image is processed into a length N vector, then use classifier(in our case SVM) for classfication result.

TODO: Pseudo code

Algorithm 3: Bag of Features

Input: N,Code-book size
Data: $X^{(j)}_{Train}, X_{Test}, Y_{Train}$, where $j = 0, 1, ..., \#Class$
Result: Classification prediction forX_{Test}, $\hat{Y_{Test}}$
begin
  i $\longleftarrow \#X_{Train}$
  foreach $x_i \in X_{Train}$ **do**
    Step 1 : Feature extraction
    - Extract regions
    - Compute descriptors

    Add extracted descriptors, $\{d_{i1}, d_{i2}, \cdots\}$ into **descriptor sets**, D
  Step 2 : Build Code Book
  For D, find N cluster using clustering method s.t. Code Book has N clusters
  foreach $d \in D$ **do**
    Assign descriptors $\{d_{i1}, d_{i2}, ..., d_{..}\}$ cluster label $\Longrightarrow$ $\{d^{(0)}_{i1}, d^{(3)}_{i2}, ..., d^{(j)}_{..}\}$, where $j \in \{0, 1, ..., N-1\}$
    Feature Dictionary, $F_j$contains descriptors with cluster label j
  for i **do**
    Step 3 : Compute Frequencies
    Compute frequencies of extracted descriptors in the Code Book, F

    foreach j **do**
      $S_{ij} = \#\left\{d^{(j)}_{i\cdot} \in F_j\right\}$
  Step 4 : Test data classification
  foreach $x \in X_{Test}$ **do**
    Perform **Step 1 + Step 3** $\Longrightarrow$ Classifier (*e.g.* SVM)
    Append classifier result to $\hat{Y_{Test}}$
  return $\hat{Y_{Test}}$

BoF with HOG

TODO: Explain BoF with HOG

TODO: BoF with HOG, high-/low- classification matrix

TODO: Bof with HOG interpret result

SIFT

TODO: Introduce SIFT

BoF with SIFT

TODO: Explain BoF with SIFT

TODO: BoF with SIFT, high-/low- classification matrix

TODO: Bof with SIFT interpret result

Feature Descriptors + SVM

TODO: flow chart

TODO: HOG + SVM

TODO: HOG + SVM, high-/low- classification matrix

TODO: HOG + SVM interpret result