Image Classification using Topological Data Analysis

Melissa Goh, Yu-Shih Chen

March 25th 2020

Goals

- Dimension Reduction for Classification Models
  - Fight the overfitting problem
  - Speed up the training process
  - Common dimension reduction methods
    1. Convolution in Convolutional Neural Networks (CNN)
    2. Principle Component Analysis (PCA)
    3. Topological Data Analysis (TDA)
    4. TDAsweep
    5. Etc.
- Example : CNN

"C" in CNN

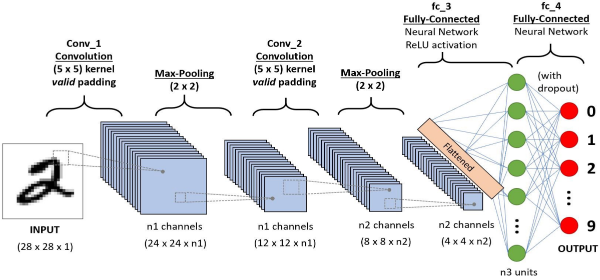
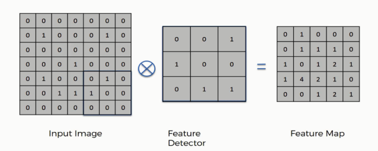


Figure – CNN Workflow



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

⊗

0	0	1
1	0	0
0	1	1

Feature Detector

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Figure – Convolution in CNN

Examples of TDA

- *Topological data analysis of high resolution diabetic retinopathy images* (Kathryn,G.,Robin,H.,Irina,M.,Cristina,M. May 24, 2019)

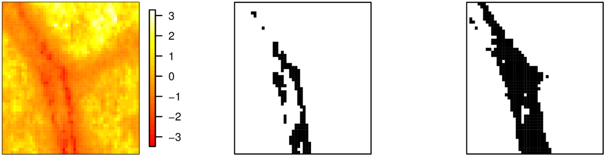


Figure – Persistent Homology on 50x50 Fundus Images

TDAsweep

- Components
  - Number of isolated pixel value(s)
    - e.g. there are 3 components in "1011101"
  - The 1s are the results after thresholding the pixel values
- The Sweeps
  - Finds the number of components in a pixel value matrix with respect to :
    1. Rows
    2. Columns
    3. Diagonals (NW to SE)
    4. Diagonal (NE to SW)

TDAsweep on MNIST

Output of TDAsweep on a random image from MNIST :

```
> ncol(img[1,])
> 784
> res <- test_one_img(img,label,28,28,F,20,2)
> res
> 0 0 0 1 2 2 2 2 1 1 1 1 1 0 0 0 0 0 0.5 2
2.5 2 2 1 0.5 0 0 0 2 2 2 2 2 1 1 0 0 0 0
0 0 2 1.5 1.5 1.5 1 1 1 0.5 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0.5 1 2 2 1.5 1 1 1 0.5
0 0 0 0 0 0 0 0 0 9    # last column is label
> ncol(res)
> 85
```

tdaImage

1. Specify images as a pixel intensity matrix, and labels as a vector.
2. Specify threshold for feature extraction.
3. Specify interval width, further decreasing the number of features.
4. Outputs a string of component counts in the row, column, and diagonals directions.
5. The set of output can then be fed into a ML method.

Threshold Value

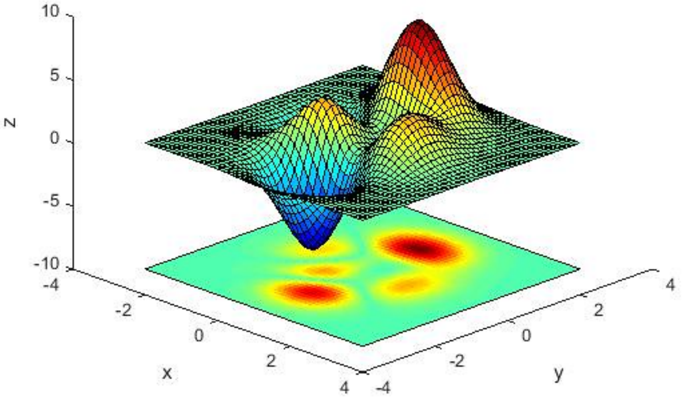


Figure – Slicing a Mountain

Example of Threshold

```
Threshold = 0:
    0 0 70 25 80 0      -> 1 component

Threshold = 26:
    0 0 70 0 80 0       -> 2 components

Threshold = 71:
    0 0 0 0 80 0        -> 1 component
```

Example of intervalWidth

```
#intervalWidth=1    #intervalWidth=2
...                  ...
3 3 2 0 0 1        1.5 2.5 2 0.5 0.5 1
0 2 2 1 1 1          ...
...
```

Takes the mean of every 2 rows, therefore further reducing the number of features in one direction by half.

TDA + SVM on MNIST

```
tstidx <- sample(nrow(mnist), 5000)
mn.tst <- mnist[tstidx,]
mn.trn <- mnist[-tstidx,]
# get TDA output
tdaout.trn <- tda_wrapper_func(mn.trn[,-785],
                    mn.trn[,785], 28, 28, F, 100, 2)
tdaout.tst <- tda_wrapper_func(mn.tst[,-785],
                    mn.tst[,785], 28, 28, F, 100, 2)
...
library('e1071')
system.time(svmout <- svm(labels~., data=tdrn, scale=F))
preds <- predict(svmout, tdst[,-85])
mean(preds == tdst$labels)  # 0.9602
```

Analysis

TDAsweep Analysis			
Parameters	MNIST(no sweep)	MNIST	MNIST
No. of Features	784	84	168
intervalWidth	NA	2	2
thresholds	NA	(100)	(100,175)
TDAsweep total time	NA	42min	78min
SVM train time	123min	14min	24min
Accuracy	97.9%	97%	97.86%

Table – Speed comparison of SVM before and after TDAsweep

- Note that the time for training has drastically decreased in exchange for a low cost of accuracy.
- TDAsweep is still time costly but is highly parallelizable

Analysis(Cont.)

- In general, for a $n \times m$ image ($n \times m$ features), TDAsweep reduced it to :

$$t \times (\lceil \frac{n}{i} \rceil + \lceil \frac{m}{i} \rceil + 2 \times \lceil \frac{n+m-1}{i} \rceil) \tag{1}$$

- where i and t denotes intervalWidth and |thresholds| respectively

Example of dimension reduction on one MNIST image :

```
> dim(test_set[1,])
[1]   1 784
tda_test_set <- tda_wrapper_func(test_set[1,],
test_y_true[1], nr=28, nc=28, rgb=FALSE, thresh=c(100)
, intervalWidth = 1)
> dim(tda_test_set)  # with label attached in the end
[1]   1 167
```

Next Step

- Try on colored image set, e.g., CIFAR-10.
- Make it faster with parallel computation - introduce clusters to distribute pre-processing work.
- Figures tend to be ragged. Add a minimum value for component length.