CS6220 Unsupervised Data Mining

HW3A Dimmensionality Reduction, Supervised Classification

Make sure you check the syllabus for the due date. Please use the notations adopted in class, even if the problem is stated in the book using a different notation.

We are not looking for very long answers (if you find yourself writing more than one or two pages of typed text per problem, you are probably on the wrong track). Try to be concise; also keep in mind that good ideas and explanations matter more than exact details.

Submit all code files Dropbox (create folder HW1 or similar name). Results can be pdf or txt files, including plots/tabels if any.

"Paper" exercises: submit using Dropbox as pdf, either typed or scanned handwritten.

DATATSET **: SpamBase: emails (54-feature vectors) classified as spam/nospam**

DATATSET **: 20 NewsGroups : news articles**

DATATSET **: MNIST : 28x28 digit B/W images**

DATATSET **: FASHION : 28x28 B/W images**

https://en.wikipedia.org/wiki/MNIST_database
http://yann.lecun.com/exdb/mnist/
https://www.kaggle.com/zalando-research/fashionmnist

PROBLEM 1: Supervised Classification

6 Runs of Suoervised Training / Testing : 3 datasets (MNIST, Spambase, 20NG) x 2 Classification Algorithms (L2-reg Logistic Regression, Decision Trees). You can use a library for the classification algorithms, and also can use any library/script to process data in appropriate formats.
You are required to explain/analyze the model trained in terms of features : for each of the 6 runs list the top F=30 features. For the Regression these correspond to the highest-absolute-value F coefficients; for Decision Tree they are the first F splits. In particular for Decision Tree on 20NG, report performance for two tree sizes (by depths of the tree, or number of leaves, or number of splits)

PROBLEM 2 : PCA library on MNIST

A) For MNIST dataset, run a PCA-library to get data on D=5 features. Rerun the classification tasks from PB1, compare testing performance with the one from PB1. Then repeat this exercise for D=20
B) Run PCA library on Spambase and repeat one of the classification algorithms. What is the smallest D (number of PCA dimmesnsions) you need to get a comparable test result?

PROBLEM 3 : Implement PCA on MNIST

Repeat PB2 exercises on MNIST (D=5 and D=20) with your own PCA implementation. You can use any built-in library/package/API for : matrix storage/multiplication, covariance computation, eigenvalue or SVD decomposition, etc. Matlab is probably the easiest language for implementing PCA due to its excellent linear lagenbra support.

PROBLEM 4 : Pairwise Feature selection for text

On 20NG, run featurre selection using skikit-learn built in "chi2" criteria to select top 200 features. Rerun a classification task, compare performance with PB1. Then repeat the whole pipeline with "mutual-information" criteria.

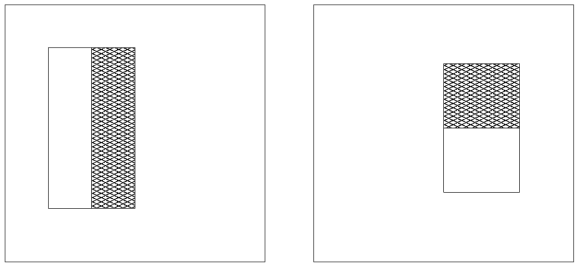
PROBLEM 5 : L1 feature selection on text

Run a strongL1-regularized regression (library) on 20NG, and select 200 features (words) based on regression coefficients absolute value. Then reconstruct the dateaset with only these features, and rerun any of the classification tasks,

PROBLEM 6 HARR features for MNIST :

Implement and run HAAR feature Extraction for each image on the Digit Dataset. Then repeat the classification task with the extracted features.

HAAR features for Digits Dataset :First randomly select/generate 100 rectangles fitting inside 28x28 image box. A good idea (not mandatory) is to make rectangle be constrained to have approx 130-170 area, which implies each side should be at least 5. The set of rectangles is fixed for all images. For each image, extract two HAAR features per rectangle (total 200 features):

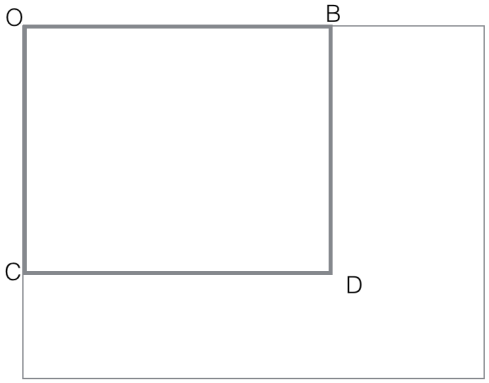


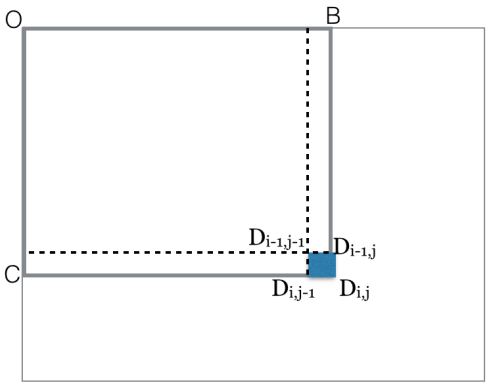
- the black horizontal difference black(left-half) - black(right-half)
- the black vertical difference black(top-half) - black(bottom-half)

You will need to implement efficiently a method to compute the black amount (number of pixels) in a rectangle, essentially a procedure black(rectangle). Make sure you follow the idea presented in notes : first compute all black (rectangle OBCD) with O fixed corner of an image. These O-cornered rectangles can be computed efficiently with dynamic programming

```
black(rectangle OBCD)= black(rectangle-diag(OD)) = count of black points in
OBCD matrix
for i=rows
for j=columns
    black(rectangle-diag(OD_ij)) = black(rectangle-diag(OD_i,j-1)) +
black(rectangle-diag(OD_i-1,j))
                                 - black(rectangle-diag(OD_i-1,j-1)) +
black(pixel D_ij)
```

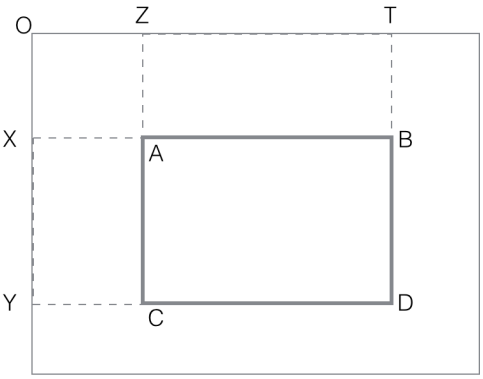
```
end for
end for
```





Assuming all such rectangles cornered at O have their black computed and stored, the procedure for general rectangles is quite easy:

```
black(rectangle ABCD) = black(OTYD) - black(OTXB) - black(OZYC) +
black(OZXA)
```



The last step is to compute the two feature (horizontal, vertical) values as differences:

```
vertical_feature_value(rectangle ABCD) = black(ABQR) - black(QRCD)
horizontal_feature_value(rectangle ABCD) = black(AMCN) - black(MBND)
```