distanceComputation

Mathieu Lagrange

November 18, 2015

Introduction

This report documents the first demonstration of the use of the expLanes framework to conduct a computational experiment. The project distanceComputation is about the comparison of several distances over synthetic data.

Design

The project is divided into two processing steps:

1. **generate**: generation of the synthetic datasets
2. **compute**: computation of the distances over those datasets and performance metrics

Generate step: *generation of synthetic datasets*

Each dataset comprises a set of points in a space of given dimensionality. Those points are grouped into classes whose centroid are spread along the diagonal. The degree of overlap between classes is controlled by the spread of the points with respect to the centroid of its class. The needed factors of variability that are to be tested are:

- the number of classes
- the dimensionality of the space
- the spread of the data points of a given class

Some other factors are set to be constant (only one modality):

- the number of replications
- the number of data points per class.

This step outputs for each dataset the location of the data points (elements) as well as the class to which they belong to (class).

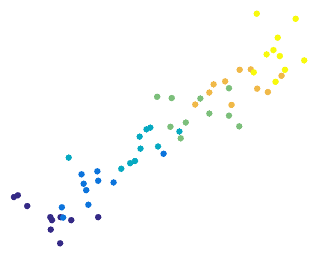

Figure 1: A dataset with average spread.

Compute step: *Computation of distances*

Once the datasets are generated, we can compute the distances between all points of a given dataset. The factor for this step is then:

1. the type of distances.

We consider three distance, the well-known Euclidean distance, with or without prior standardization over the dimensions of the space, and the cosine distance defined as:

$$d_c(x, y) = 1 - \frac{xy'}{\sqrt{(xx')(yy')}}$$

Observations

Since we consider distances between data points that each belong to a given class, ranking metrics are appropriate to evaluate the performance of those distances on the generated datasets [1]. More precisely, we consider the following metrics:

1. **map**: mean average precision
2. **precision**: precision at rank 5.

[1]

Definition of factors

Those factors and their corresponding modalities are defined in the file named `dicoFactors.txt` whose content is the following:

```
nbClasses === 2:2:6
nbDimensions === [2 5]
spread === 0:10:100
distance =2== {'euclidean', 'seuclidean', 'cosine'}
nbRealizations === 5
nbElementsPerClass === 10
```

Please note the 2 between the first two equals of the factor definition of factor `distance` which specify that this factor is needed only for the second processing step. Most the factor design discussed above is compactly displayed in Figure **??**. Modalitiesn`bRealizations` and `nbElementsPerClass` are fixed.

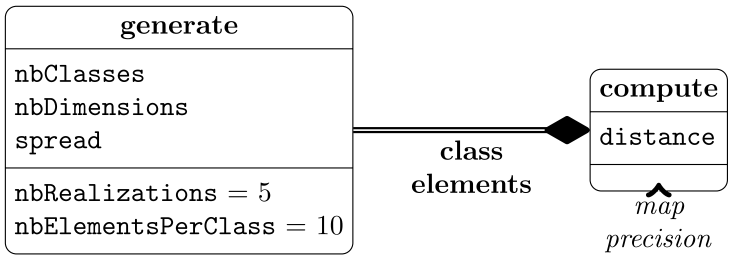

Figure 2: Factor and data flow graph.

Implementation

Each processing step is implemented in a given file.

Step 1: generate

The datasets are generated with the following code:

```
for m=1:setting.nbClasses
    elements = [elements; repmat(m, ...
        setting.nbElementsPerClass, ...
        setting.nbDimensions)...
        + randn(setting.nbElementsPerClass, ...
            setting.nbDimensions)*(setting.spread/100)];
    class = [class; m*ones(setting.nbElementsPerClass, 1)];
end

if all(setting.infoId(1:3) == [3 1 7]) && k==1
```

Step 2: compute

The data points are retrieved and, for some distances, pre-processed as follows:

```
elts = data.elements{k};
if strcmp(setting.distance, 'seuclidean')
    elts = bsxfun(@rdivide, elts, std(elts));
end
if strcmp(setting.distance, 'cosine')
    elts = bsxfun(@rdivide, elts, sqrt(sum(elts.^2, 2)));
end
```

Then, the actual distances are computed as follows:

```
for m=1:size(elts, 1)
    for n=m:size(elts, 1)
        switch setting.distance
            case {'euclidean', 'seuclidean'}
                d(m, n) =  norm(elts(m, :)-elts(n, :));
            case 'cosine'
                d(m, n) = 1-sum(elts(m, :).*elts(n, :));
        end
        d(n, m) = d(m, n);
```

Since the complete distance matrix is symmetric, only half the values are computed.

The observations, namely the 2 metrics are generated and stored as follows:

```
p = rankingMetrics(d, data.class{k});
obs.map(k) = p.meanAveragePrecision;
obs.precision(k) = p.precisionAt5;
```

Results

This report and all needed data can be generated using the following command:

```
distanceComputation('do', 0, 'report', 'rcvd');
```

Morphology of the datasets

We first want to study three factors that influence the performance of the different ranking metrics, namely the number of classes, the number of dimensions of the space and the intra class spread. To this matter, we set the metric to be the well known euclidean distance. Table 1 shows the impact of the number of classes. As the number of classes increases, the problem appears to become harder even though no statistical difference can be found.

Table 2 shows the impact of the number of dimensions. As this factor increases, the separability grows and consequently lead to a significant increase in performance.

Figure 3 shows the negative impact of the intra class spread on the MAP.

Those three displays are respectively generated by the following lines:

```
config = expExpose(config, 'table', 'mask', {0 2 7 1}, ...
    'obs', [1 2], 'percent', 0);
config = expExpose(config, 'table', 'mask', {3 0 7 1}, ...
    'obs', [1 2], 'percent', 0);
config = expExpose(config, 'linePlot', 'mask', {3 2 0 1 0 ...
    0}, 'obs', 1, 'expand', 'spread');
```

Distance Metrics

We chose to study three distances, respectively the Euclidean distance, the standardization Euclidean distance and the cosine distance. Given the morphology of the datasets, the standardization shall have a negligible impact on the performance. This is numerically verified by the experiment as shown on Table 3, where no statistical can be found over those two distances.

Considering the cosine distance have a significant negative impact on the performance, as the implied magnitude normalization is not meaningful for the datasets considered.

nbClasses	map (%)	precision (%)
2	**91±4**	**93±5**
4	85±2	87±3
6	85±4	**86**±5

Table 1: nbDimensions: 5, spread: 60, distance: euclidean

nbDimensions	map (%)	precision (%)
2	73±2	73±3
5	**85±4**	**86±5**

Table 2: nbClasses: 6, spread: 60, distance: euclidean

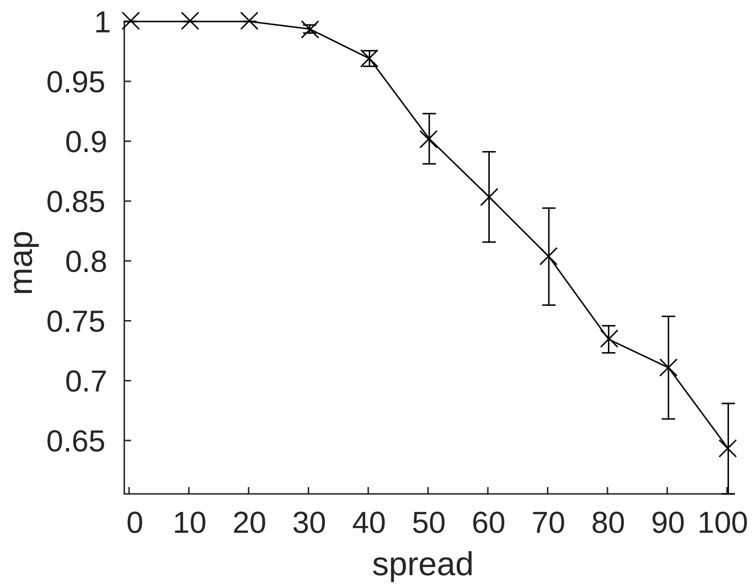

Figure 3: nbClasses: 6, nbDimensions: 5, distance: euclidean

distance	map (%)	precision (%)
euclidean	**85±4**	**86±5**
seuclidean	**85±4**	**86±5**
cosine	37±1	38±1

Table 3: nbClasses: 6, nbDimensions: 5, spread: 60