University of Groningen

Artificial Intelligence 2

Lab 3

Siger Steenstra
s2408791

October 19, 2016

Testing K-Means

In this section the influence of the prefetch threshold and the number of clusters on the hitrate and accuracy is investigated for the K-Means algorithm.

K	Threshold	hitrate	accuracy
2	0.01	0.999	0.223
2	0.1	0.958	0.264
2	0.2	0.881	0.336
2	0.35	0.763	0.437
2	0.5	0.664	0.519
2	0.65	0.578	0.596
2	0.8	0.446	0.706
2	0.9	0.330	0.773
2	0.95	0.224	0.807

K	Threshold	hitrate	accuracy
3	0.01	0.993	0.228
3	0.1	0.933	0.281
3	0.2	0.872	0.335
3	0.35	03746	0.429
3	0.5	0.654	0.495
3	0.65	0.551	0.572
3	0.8	0.471	0.613
3	0.9	0.342	0.696
3	0.95	0.212	0.672

K	Threshold	hitrate	accuracy
4	0.01	0.990	0.230
4	0.1	0.926	0.286
4	0.2	0.863	0.341
4	0.35	0.750	0.423
4	0.5	0.656	0.488
4	0.65	0.560	0.560
4	0.8	0.469	0.602
4	0.9	0.356	0.679
4	0.95	0.226	0.642

K	Threshold	hitrate	accuracy
5	0.01	0.980	0.237
5	0.1	0.921	0.294
5	0.2	0.863	0.343
5	0.35	0.769	0.430
5	0.5	0.681	0.503
5	0.65	0.599	0.562
5	0.8	0.508	0.603
5	0.9	0.383	0.699
5	0.95	0.294	0.693

K	Threshold	hitrate	accuracy
6	0.01	0.960	0.251
6	0.1	0.921	0.297
6	0.2	0.851	0.351
6	0.35	0.761	0.414
6	0.5	0.699	0.484
6	0.65	0.591	0.558
6	0.8	0.497	0.623
6	0.9	0.371	0.658
6	0.95	0.276	0.652

K	Threshold	hitrate	accuracy
7	0.01	0.970	0.240
7	0.1	0.907	0.297
7	0.2	0.849	0.344
7	0.35	0.756	0.416
7	0.5	0.641	0.492
7	0.65	0.578	0.526
7	0.8	0.489	0.602
7	0.9	0.385	0.635
7	0.95	0.306	0.617

K	Threshold	hitrate	accuracy
8	0.01	0.960	0.266
8	0.1	0.913	0.295
8	0.2	0.859	0.342
8	0.35	0.759	0.434
8	0.5	0.667	0.504
8	0.65	0.621	0.555
8	0.8	0.547	0.593
8	0.9	0.428	0.678
8	0.95	0.397	0.677

K	Threshold	hitrate	accuracy
9	0.01	0.938	0.260
9	0.1	0.910	0.298
9	0.2	0.852	0.339
9	0.35	0.735	0.414
9	0.5	0.644	0.471
9	0.65	0.601	0.522
9	0.8	0.473	0.566
9	0.9	0.377	0.634
9	0.95	0.356	0.625

K	Threshold	hitrate	accuracy
10	0.01	0.945	0.242
10	0.1	0.898	0.291
10	0.2	0.836	0.347
10	0.35	0.740	0.423
10	0.5	0.632	0.508
10	0.65	0.592	0.531
10	0.8	0.505	0.587
10	0.9	0.402	0.606
10	0.95	0.360	0.588

K	Threshold	hitrate	accuracy
11	0.01	0.931	0.282
11	0.1	0.916	0.296
11	0.2	0.850	0.347
11	0.35	0.752	0.424
11	0.5	0.666	0.499
11	0.65	0.628	0.530
11	0.8	0.516	0.588
11	0.9	0.433	0.650
11	0.95	0.414	0.651

K	Threshold	hitrate	accuracy
12	0.01	0.933	0.270
12	0.1	0.897	0.308
12	0.2	0.854	0.347
12	0.35	0.739	0.415
12	0.5	0.630	0.502
12	0.65	0.610	0.521
12	0.8	0.487	0.582
12	0.9	0.410	0.597
12	0.95	0.391	0.590

It is clear that with a low threshold, you have a very high hit-rate. This is quite logical, because nearly everything is prefetched. As the threshold increases, so does the accuracy, while the hitrate decreases. The fact that the hitrate decreases is quite logical because it becomes less likely for a request to be prefetched. The question is whether you want to prefetch everything you can, because in real life applications this means you need to store more data, with the risk that some (or worse, most) prefetched data goes unused. Personally I would opt for an intermediate accuracy, while maintaining a fair hitrate (finding a balance). Now, with clustering there never is a perfect solution. What I observed was that certain datapoints were always in small clusters, which probably means that there are some outliers. This means that with a small number of clusters the data might suffer the consequences of the outliers. So a few extra clusters might be necessary to account for this. I would like to have an accuracy of at least 0.5, and then see what the hit rate is. With 5 clusters and a threshold of 0.5 a hit rate of 0.681 is obtained. In my opinion this is the best setting. It loads a lot of your requests quicker (prefetched), while not unnecessarily prefetching URL's.

Testing Kohonen

First appropriate number of training epochs is determined. The best results are obtained with 500 epochs, but if time is essential you can also obtain good results with less epochs. Now the influence of the number of clusters and the value of the prefetch threshold are

investigated for their influence. The results are below:

N*N	Threshold	hitrate	accuracy
4	0.01	0.764	0.422
4	0.1	0.692	0.485
4	0.2	0.470	0.553
4	0.35	0.470	0.553
4	0.5	0.470	0.553
4	0.65	0.470	0.553
4	0.8	0.470	0.553
4	0.9	0.348	0.772
4	0.95	0.348	0.772

N*N	Threshold	hitrate	accuracy
9	0.01	0.723	0.478
9	0.1	0.665	0.502
9	0.2	0.520	0.643
9	0.35	0.520	0.643
9	0.5	0.520	0.643
9	0.65	0.520	0.643
9	0.8	0.520	0.643
9	0.9	0.405	0.757
9	0.95		

N*N	Threshold	hitrate	accuracy
16	0.01	0.722	0.482
16	0.1	0.673	0.502
16	0.2	0.521	0.647
16	0.35	0.521	0.647
16	0.5	0.521	0.647
16	0.65	0.521	0.647
16	0.8	0.521	0.647
16	0.9	0.419	0.760
16	0.95	0.419	0.760

N*N	Threshold	hitrate	accuracy
25	0.01	0.742	0.516
25	0.1	0.683	0.564
25	0.2	0.578	0.644
25	0.35	0.578	0.644
25	0.5	0.578	0.644
25	0.65	0.578	0.644
25	0.8	0.578	0.644
25	0.9	0.449	0.760
25	0.95	0.449	0.760

N*N	Threshold	hitrate	accuracy
36	0.01	0.740	0.496
36	0.1	0.701	0.547
36	0.2	0.596	0.647
36	0.35	0.596	0.647
36	0.5	0.596	0.647
36	0.65	0.596	0.647
36	0.8	0.596	0.647
36	0.9	0.481	0.752
36	0.95	0.481	0.752

It seems that as the number of clusters increases, the performance increases as well. This seems like a good thing, but if we are going to test on different data we might get vastly worse results, which might be an effect of overfitting. What is observable here is that the Kohonen-SOM algorithm is less influenced by the prefetch threshold. Without overfitting too much I think the best results are obtained with $N * N = 25$ and a very low threshold (0.01). The hitrate is 0.742 while the accuracy is 0.516. This means however that nearly all data has to be prefetched, which requires a lot of resources.

Comparing algorithms

From my results it seems that a SOM gives better results than K-Means. But, before simply concluding that SOM is definitely better, it might be a good idea to consider the additional parameters and what they mean. Yes, SOM gets better results, but at what cost? Nearly all data has to be prefetched, while only retaining a hitrate of 0.742. Therefore I think that K-Means is a better fit for the purpose of this assignment. It might of course also have to do with the input data, which almost definitely contains outliers (I feel like I have seen enough evidence). If the data was pruned and normalized a little more the results might be vastly influenced. However, the given data is probably a good reflection for real life problems we are facing.

When looking at the code for K-Means and SOM's, the algorithm for a SOM is defi-

nitely larger in terms of time complexity.

The difference in performance could also be explained by the fact that the algorithms have a different approach. In K-Means, each cluster center is recalculated based on its members. While in a SOM a data point can influence multiple cluster centers at once, namely the ones that are close (cluster centers might bounce around more than in K-Means, especially in the beginning of training).