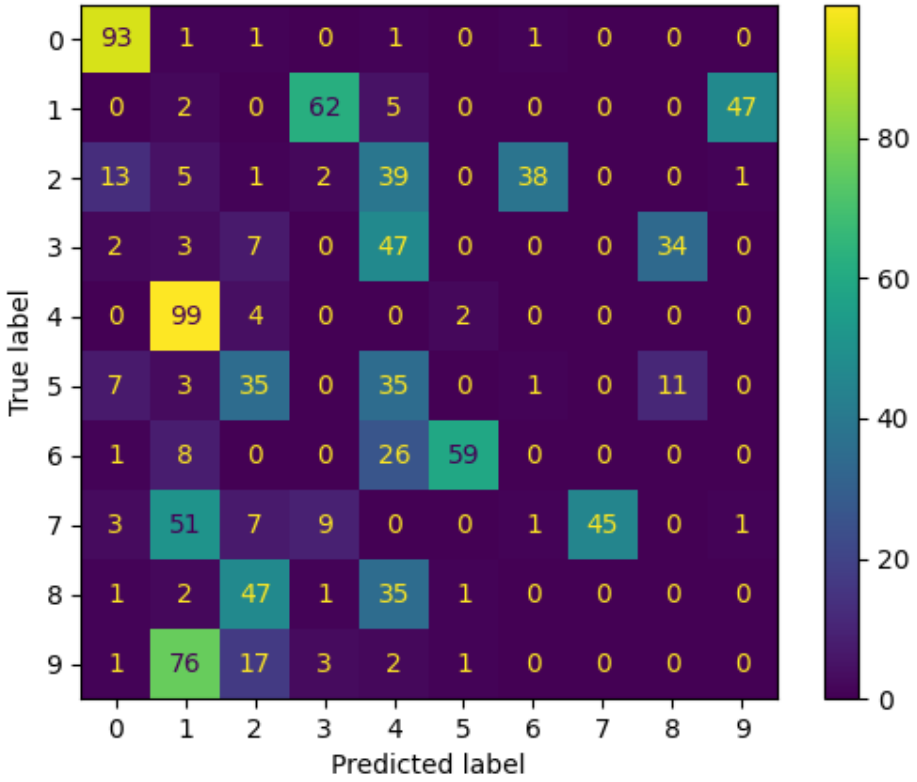
I have not been able to implement k-means and will not compare this to agglomerative clustering.

3

score for ward: 0.34203838268276526
score for complete: 0.1139682334943005
score for average: 0.04724055147950766
score for single: 0.00023274678616116512

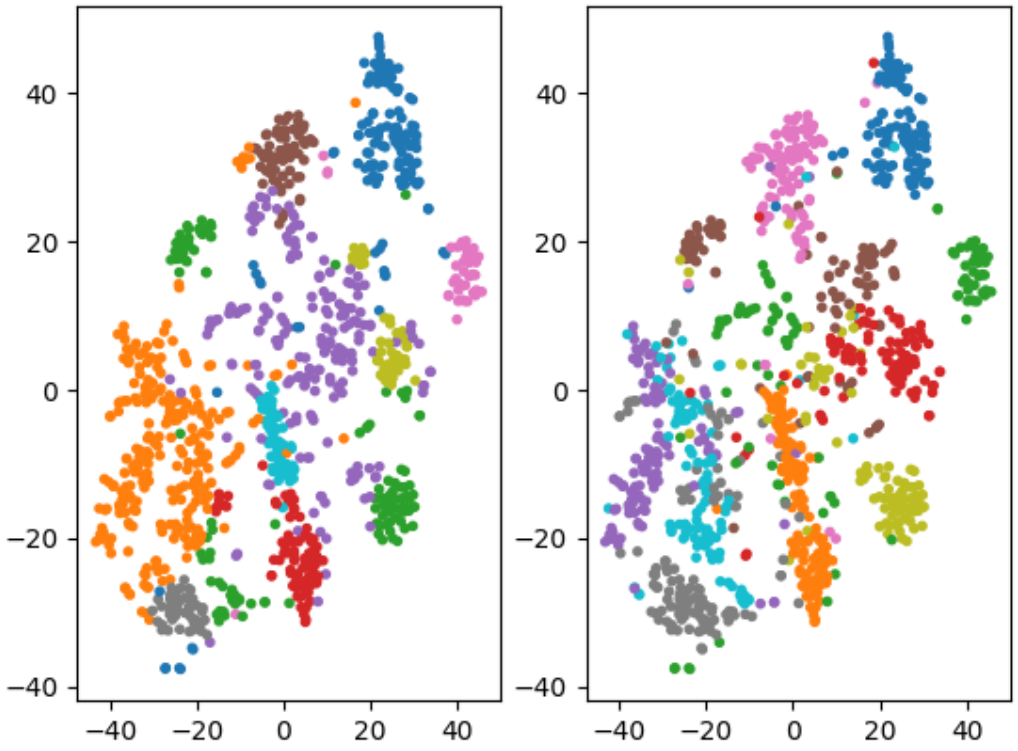
Ward's method performs better than the other methods.

4



4 and 0 seem to be classified really well, as well as 6 to some extent. 5 and 8 seem to be confused pretty easily. The model classifies 9 in the same cluster as 4. 7 is classified in this same cluster quite often as well. Surprisingly enough, 1 seems to be put into two predicted clusters, but both don't have any major overlap with other classes. I had expected 1 and 7 to be clustered together more often, because their shapes appear quite similar sometimes.

5



I think there's some classes the model clusters really well based on the graphs. Because the ARI score is measured over all classes and clusters, and not individually I feel like the ARI score might be more negatively impacted by the misplacing of some clusters than what would accurately reflect the quality of clustering based on the graph. However the graph is a 2d interpretation of a multidimensional clustering, so it could be that that leads to human misinterpretation.

6

score for hierarchical clustering with gaussian_blur_4x4 kernel: 0.288169813814006
score for hierarchical clustering with moldy_frikandel kernel: 0.4113651958055041

score for hierarchical clustering with sobel_0_3x3 kernel: 0.2681042270936365
score for hierarchical clustering with sobel_0_5x5 kernel: 0.3957797406351622
score for hierarchical clustering with sobel_0_7x7 kernel: 0.5232146961925138
score for hierarchical clustering with sobel_0_9x9 kernel: 0.3983815062303457

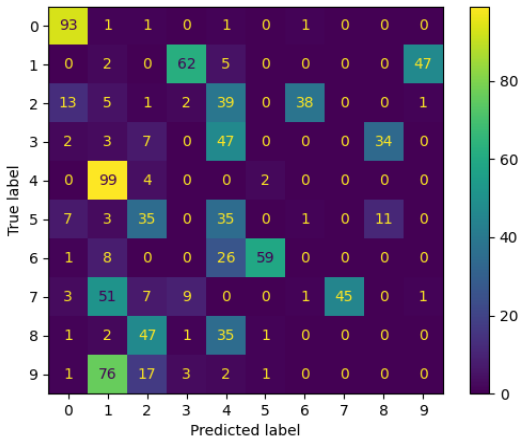
I picked the kernel size with the highest score on 0 degree rotation, and experimented with rotations between 0 and 90 degrees.

score for hierarchical clustering with sobel_0_7x7 kernel: 0.5232146961925138
score for hierarchical clustering with sobel_10_7x7 kernel: 0.42672440540941364

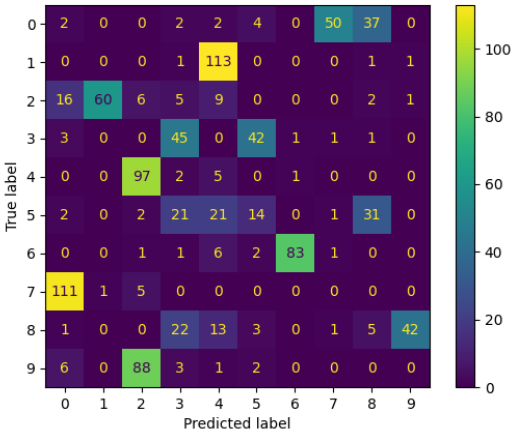
score for hierarchical clustering with sobel_20_7x7 kernel: 0.4933581986408891
score for hierarchical clustering with sobel_30_7x7 kernel: 0.41687287856709865
score for hierarchical clustering with sobel_40_7x7 kernel: 0.4446077775467421
score for hierarchical clustering with sobel_50_7x7 kernel: 0.37676720905293754
score for hierarchical clustering with sobel_60_7x7 kernel: 0.4414940757274717
score for hierarchical clustering with sobel_70_7x7 kernel: 0.3870726939504846
score for hierarchical clustering with sobel_80_7x7 kernel: 0.32487338456147213
score for hierarchical clustering with sobel_90_7x7 kernel: 0.40731354022233907

The rotation with the best score seems to be 0 degrees, even when decreasing the steps to 1 degree at a time.

For scoring per digit it might be a good idea to first look at the difference between confusion matrices of hierarchical clustering with and without sobel filters. The figure below shows the confusion matrix with and without filters.



No filter



sobel_0_7x7

Though somehow 9 and 4 are still grouped together after the filter, other classes are clustered way better than before. Interestingly enough 0 is now split into two clusters and 1 is nearly perfectly clustered as one.

Scoring predictions per digit proved harder than imagined. The adjusted rand score (the way I understand it) does not work well with subsets where the labels are all the same.

I made a very crude scoring system that calculated the percentage of a true label that the model has correctly clustered as one. This however fails to account for double clustering (e.g. 4 and 9 being incorrectly grouped as one) and is a very different way of scoring in general.

best rotation for digit 0 = sobel_80_7x7 score = 0.6804123711340206
best rotation for digit 1 = sobel_0_7x7 score = 0.9741379310344828
best rotation for digit 2 = sobel_0_7x7 score = 0.6060606060606061

best rotation for digit 3 = sobel_40_7x7 score = 0.7634408602150538
best rotation for digit 4 = sobel_40_7x7 score = 1.0
best rotation for digit 5 = sobel_10_7x7 score = 0.40217391304347827
best rotation for digit 6 = sobel_50_7x7 score = 0.9148936170212766
best rotation for digit 7 = sobel_0_7x7 score = 0.9487179487179487
best rotation for digit 8 = sobel_60_7x7 score = 0.8275862068965517
best rotation for digit 9 = sobel_60_7x7 score = 0.91

The results are still interesting in my opinion, but as to why certain rotations work better for certain numbers I currently have no meaningful explanation.

score for hierarchical clustering with gaussian_blur-1x1 kernel: 0.34203838268276526
score for hierarchical clustering with gaussian_blur-2x2 kernel: 0.3780770640174297
score for hierarchical clustering with gaussian_blur-3x3 kernel: 0.26453744296787046
score for hierarchical clustering with gaussian_blur-4x4 kernel: 0.288169813814006
score for hierarchical clustering with gaussian_blur-5x5 kernel: 0.33737896628023767
score for hierarchical clustering with gaussian_blur-6x6 kernel: 0.30592139602915164
score for hierarchical clustering with gaussian_blur-7x7 kernel: 0.3183971059201032
score for hierarchical clustering with gaussian_blur-8x8 kernel: 0.3309230691835175
score for hierarchical clustering with gaussian_blur-9x9 kernel: 0.2855444966513428
score for hierarchical clustering with gaussian_blur-10x10 kernel: 0.23440140945753135

For gaussian blur the best scoring filter is 2x2, surprisingly. Perhaps gaussian blur is just not very effective for this task and applying more blur makes it worse. However with that hypothesis we would expect a continuous drop in scores the bigger the filter, which is not the case.

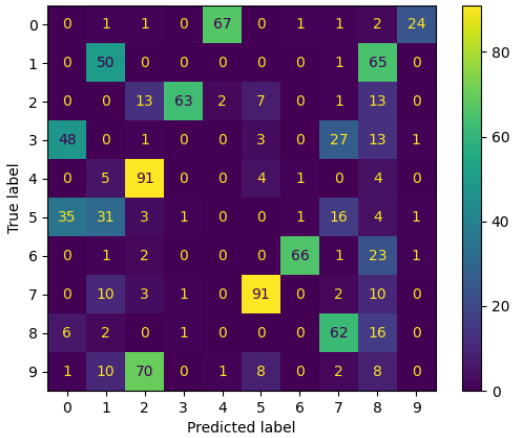
best filter for digit 0 = gaussian_blur-1x1 score = 0.9587628865979382
best filter for digit 1 = gaussian_blur-19x19 score = 0.5517241379310345
best filter for digit 2 = gaussian_blur-5x5 score = 0.7272727272727273
best filter for digit 3 = gaussian_blur-9x9 score = 0.8602150537634409
best filter for digit 4 = gaussian_blur-1x1 score = 0.9428571428571428
best filter for digit 5 = gaussian_blur-6x6 score = 0.5869565217391305
best filter for digit 6 = gaussian_blur-4x4 score = 0.723404255319149
best filter for digit 7 = gaussian_blur-5x5 score = 0.6666666666666666
best filter for digit 8 = gaussian_blur-11x11 score = 0.7011494252873564
best filter for digit 9 = gaussian_blur-1x1 score = 0.76

The crude sorted scores are probably quite meaningless, but at least somewhat expected. Digits that were already clustered really well (though be it in combination with another digit) didn't seem to benefit from high blurring, and digits that were classified in different clusters seemed to benefit from blurring more.

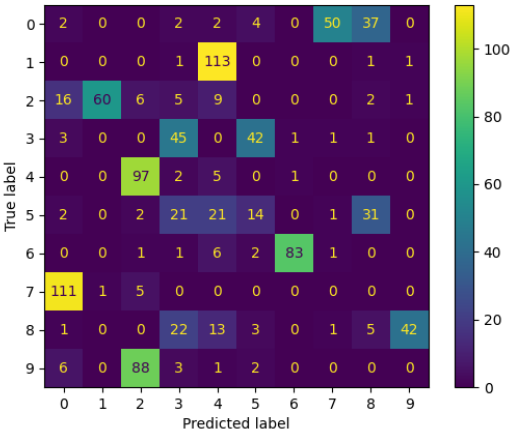
Bonus:

score for hierarchical clustering with moldy_frikandel kernel: 0.4113651958055041
score for hierarchical clustering with sobel_0_7x7 kernel: 0.5232146961925138

Sobel 7x7 seems to outperform moldy_frikandel. Below are the two confusion matrices next to each other:



moldy_frikandel



sobel_0_7x7

The narrower spread in sobel_0_7x7 seems to be the reason it outperforms moldy_frikandel with the adjusted random index. Moldy_frikandel does seem to have better defined clusters for some classes, but according to the metric, sobel_0_7x7 scores better.

7

Purity scores are apparently very similar to my “crude” metric used to calculate a score per digit.

purity score for agglomerative clustering without filters: 0.608
purity score for agglomerative clustering with sobel_0_7x7: 0.72
purity score for agglomerative clustering with moldy_frikandel: 0.658
purity score for agglomerative clustering with gaussian_blur_2x2: 0.643

No filters seem to perform the worst, and once again sobel_0_7x7 appears to score the best.