STAT639:
Final Report on Machine Learning Tasks

John M. Niehaus*

UIN: 225009469

April 23, 2020

*Graduate student, Texas A&M University, Department of Political Science

1 Supervised Learning

For the supervised learning task, I train six classifiers: k-nearest neighbors, naïve bayes (NB), logistic elastic net, random forest, support vector machine, and stochastic gradient boosted trees. Furthermore, for KNN I investigate both standardized and non-standardized data for the task. All models are tuned using grid-search (where applicable), and repeat, nested cross-validation.[1] Repeat, nested cross validation has two advantages: first, it permits me to simultaenously tune optimal parameters and estimate test error; and second, the repetition of this process reduces the variance on the estimate of the test error due to CV sampling. Briefly, the process can heuristically be understood as follows:

Divide the training data into K_o outer testing folds. Then, for $k_o \in \{1,2,...,K_o\}$, do:

1. Divide the remaining $K_o - 1$ outer folds into K_i inner folds
2. Do K_i-fold cross validation on the K_i inner folds to get the optimal tuning parameters, $\hat{\beta}_o$
3. Estimate test error using $\hat{\beta}_o$ on k_o

After this process is completed, average the K_o estimates of the test error. Then, repeat the above process the R times, averaging the R estimated test errors so as to reduce variance due to CV sampling. Additionally, the set of tuning parameters that is selected in each of the K_i inner CV loops is the optimal set of tuning parameters. In my case, I use $K_o = 6$, $K_i = 5$, and R=5. **Following this process yields a estimated test-error of 21.6%.**

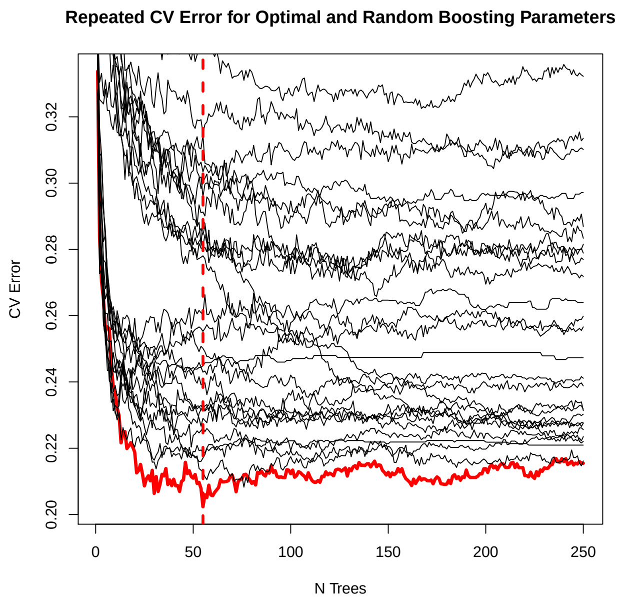
In my case, the inner cross-validation iterations are not completely stable. That is, they each produce slightly different optimal tuning parameters. However, all of them select a stochastic gradient boosted tree, and all of the estimated cross-validation errors for these boosted trees are roughly the same. Table 1 in the appendix demonstrates this fact, where each row is the optimal set of parameters and corresponding inner-CV error when a particular outer-fold is left out.

Because the parameters are unstable but the cross-validation errors are roughly the same, it seems likely that the procedure will still generalize well to test data. In hindsight, some parameters in a boosted tree interact with each other, which may lead to this instability. For example, a higher learning rate tends to imply fewer trees, so the effects of these parameters

[1]See the appendix for the exact grids used for each method

are not independent of each other. In order to overcome this parameter instability, I run another set of cross-validation on the entire training data in order to select the optimal parameters to a boosted tree, but I retain my estimated test error from the nested CV. Because all inner CV loops selected a boosted tree, and their respective CV estimates are roughly 10 percentage points lower than those of any other method, I only consider boosted trees on this additional CV grid-search.

In more detail, stochastic gradient boosting comes from Friedman (2002), and differs from typical gradient boosting by permitting the random sampling of both observations and covariates during the fitting process, analogously to Random Forest. However, **due to the small sample size, I only subsample the covariates, not the observations**. Second, I also impose a minimum loss reduction required in order to honor a split. That is, a split must reduce the loss by some minimum threshold in order to be honored. This is done to reduce over-fitting. Aside from these two modifications, the algorithm is the same as boosting as we have already seen it: at each iteration in the process, we fit a tree that up-weights observations that were fit poorly on the previous iteration, and then add all these trees together.[2] The CV curve for the optimal set of parameters from this second CV grid-search, as well as the curves for a random selection of parameters from my grid are shown below. **The optimal boosting curve has 55 trees, a learning rate of .0001, a feature sampling ratio of 0.25, and a max depth of 9,**



[2]In this sense, boosting fits an additive model (Friedman, Hastie and Tibshirani 2001, Ch. 10)

2 Unsupervised Learning

For the unsupervised learning task, I considered four clustering algorithms: K-means, Gaussian mixture models, hierarchical clustering, and density based scanning. However, in order to effectively cluster these data, it is necessary to first reduce the number of dimensions, as the raw data have 784 dimensions, which implies that all of the observations will be grouped in the corners of this hypercube. As a result, euclidean distance breaks down.

In order to reduce the dimensionality of this data, I used principle components analysis (PCA), and then retained a subset of the PCs according to the proportion of variance explained. Figure 1 below demonstrates the cumulative proportion of variance explained, as well as the scree plots. Moving from left to right, the X-axis is merely trimmed in order to magnify local information. The red, green, and blue lines indicate 5, 20, and 40 PCs, respectively.

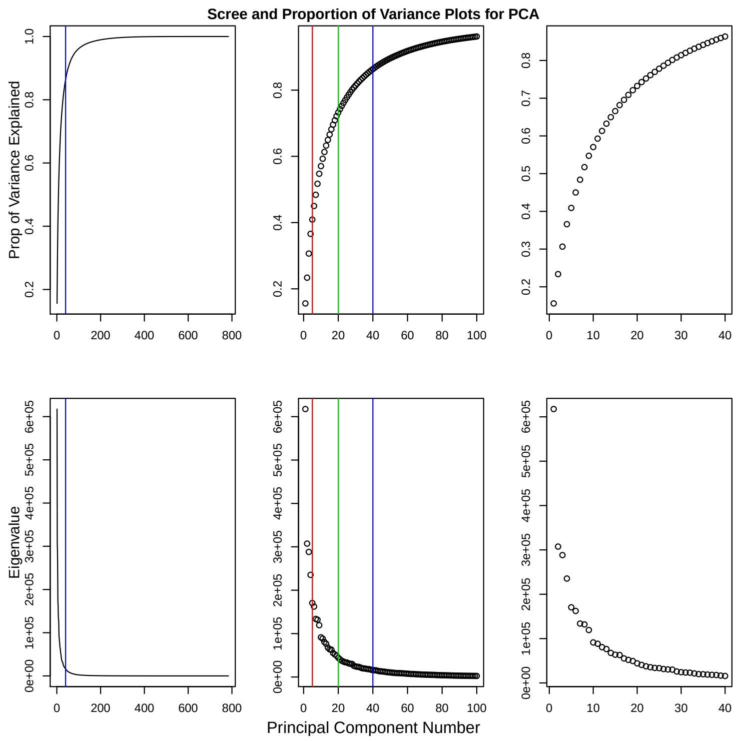


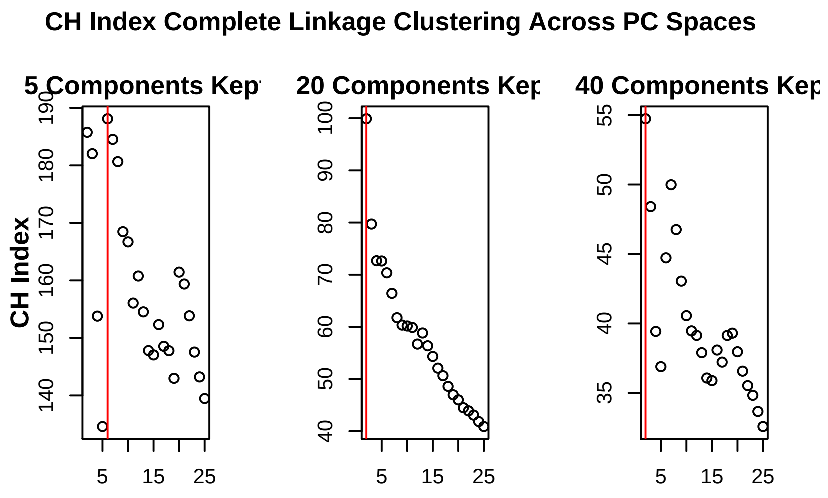
Figure 1

I then use each clustering algorithm on each of these PC spaces. However, due to space constraints, I show only the results from two algorithms: hierarchical with single linkage, and DBSCAN. Additional results can be found in the appendix.

To determine the number of clusters, K, I use the CH index (Caliński and Harabasz 1974).[3] The CH index is defined as:

$$CH(K) = \frac{BSS/K-1}{TWSS/n-K}$$

Intuitively, we seek the value of K such that $CH(K)$ is maximized. In this way, we maximize between cluster distance, while minimizing within cluster distance. Values of the CH-index across PC spaces and values of K for hierarchical, complete linkage clustering are shown below.



Both the 20 and 40 component spaces agree on two clusters, while the 5 component space suggests 6 clusters.

For DBSCAN, the number of clusters is not decided on ex-ante. However, I need to decide on the epsilon and minimum points parameters to the algorithm. To do so, I lay down a grid for each parameter, and do grid-search to determine the set of parameters that maximizes CH for DBSCAN.[4] The results are found in the following table.

	Eps	Min. Points	N Clusters
5 PCs	595	40	2
20 PCs	1255	40	2
40 PCs	1455	25	3

As can be seen above, DBSCAN results in a clustering solution with 2 or 3 clusters, depending on the PC space. **This result, combined with that of the complete linkage hierarchical clustering leads me to conclude that there are 2 clusters.**

[3]However, see Charrad et al. (2014) for details on 30 other indices, and their implementations in R.
[4]See the appendix for the method used to determine this grid.

References

Caliński, Tadeusz and Jerzy Harabasz. 1974. "A dendrite method for cluster analysis." *Communications in Statistics-theory and Methods* 3(1):1–27.

Charrad, Malika, Nadia Ghazzali, Veronique Boiteau and Azam Niknafs. 2014. "NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set." *Journal of Statistical Software* .

Friedman, Jerome H. 2002. "Stochastic gradient boosting." *Computational statistics & data analysis* 38(4):367–378.

Friedman, Jerome, Trevor Hastie and Rob Tibshirani. 2010. "Regularization paths for generalized linear models via coordinate descent." *Journal of statistical software* 33(1):1.

Friedman, Jerome, Trevor Hastie and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1 Springer series in statistics New York.

Rahmah, Nadia and Imas Sukaesih Sitanggang. 2016. Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra. In *IOP Conference Series: Earth and Environmental Science*. Vol. 31 IOP Publishing p. 012012.

Appendix

A Grids

Among perhaps the simpler methods, the grid for KNN consists of all $k \in [1, 50]$, while naïve bayes has no grid. For the GLMs, the elastic-net mixing parameter between LASSO and Ridge (typically denoted α) was spaced by 0.1 on the unit interval, while the regularization parameter, λ, was chosen according to the method described in Friedman, Hastie and Tibshirani (2010, 8), wherein the maximum value of λ is found such that all coefficients will be set to zero (or very close to zero, for Ridge).[5] In order to ensure sufficient grid coverage for all possible cross-validation splits of the data, I randomly drew 5000 samples of the training data of equal size to that considered in the cross-validation step, and then computed the maximum and minimum values of lambda for each of these samples according to Friedman, Hastie and Tibshirani (2010, 8). Finally, I took the maximum value from all of the λ values derived this way as the upper boundary for the grid-search. For random forest, I chose a grid for the number of variables to consider at each split by first starting with the default reccomendations of $m = \sqrt{p}$, then $m = \sqrt{p} + 50$, and then a sequence from 100 to p, spaced by 50. For support vector machines, I considered a linear, polynomial, and radial kernel. For each kernel, I used the same cost grid, with powers from 10^{-3} up to 10^{5}. Additionally, for the radial kernel I used a γ of 10^{-5} to 10^{5}, and for the polynomial kernel, I considered all degrees from 2 to 10. Lastly, for stochastic gradient boosting, I considered up to 500 trees, learning rates of $.0001, .001, .01, .1$, and 0.3, a minimum loss reduction required to make a split of $0, 0.01, 0.1, 0.5$ and 1, proportion of features to consider at each split of $\sqrt{p}/p, 0.25, 0.5, 0.75$ and 1, and lastly, all depths from 1 to 9, spaced by 2.

[5] Although the R package `glmnet` will select a grid automatically when calling `cv.glmnet`, I chose to manually lay down a grid so that each iteration of cross validation is consistent in the grid values used.

B Proof of CV Stability in Inner CV Boosting

	λ	Depth	Col. Sample	γ	Trees	CV
Outer Fold 1	0.0001	9	0.25	0.01	137	0.1993
Outer Fold 2	0.01	9	0.5	0	44	0.1972
Outer Fold 3	0.01	7	0.25	0	71	0.1841
Outer Fold 4	0.001	9	0.25	0.5	32	0.2055
Outer Fold 5	0.01	7	0.25	1	24	0.2009
Outer Fold 6	0.01	7	0.5	1	28	0.1668

$\boldsymbol{\lambda}$ = learning rate, $\boldsymbol{\gamma}$ = loss reduction required for split, "**col. sample**"= proportion of features considered at each split

Table 1: Optimal Inner Parameters for One Repitition

C Additional Clustering Results

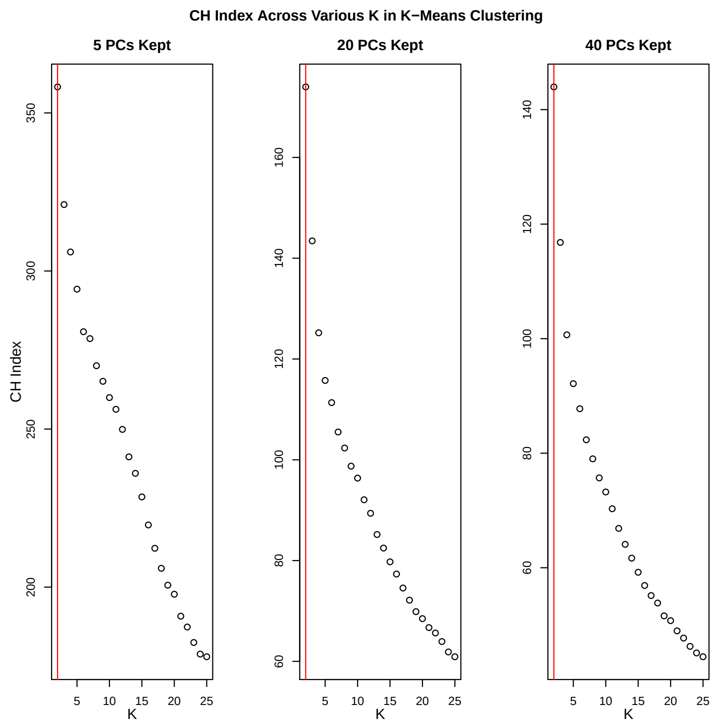
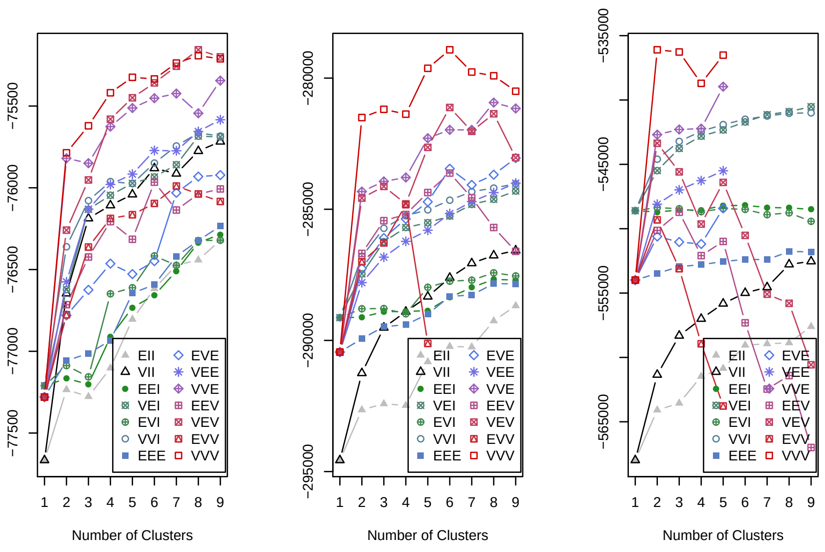
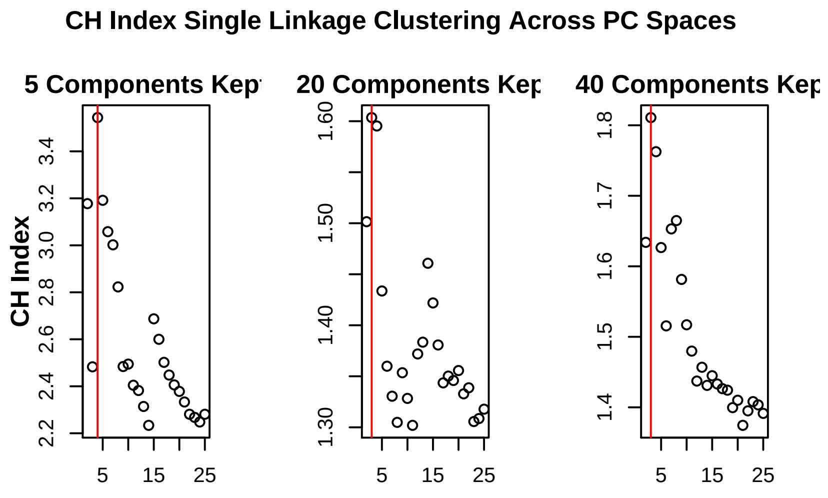
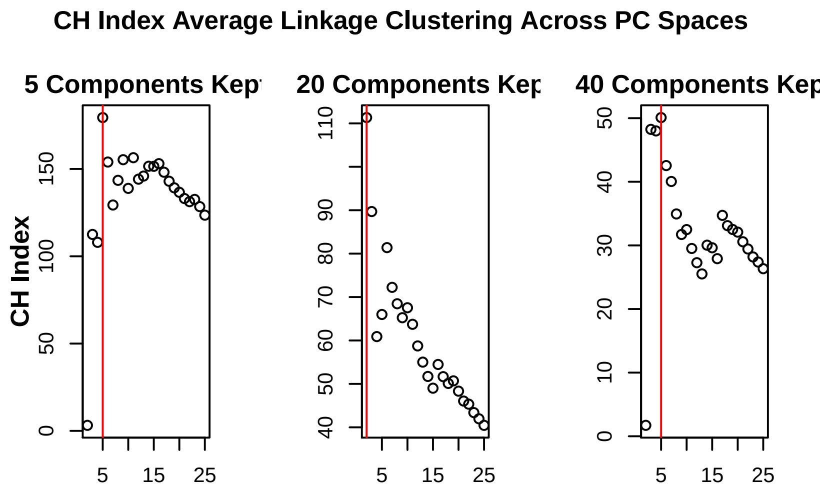


Figure 2: BICs across Restrictions and PC Spaces for GMM





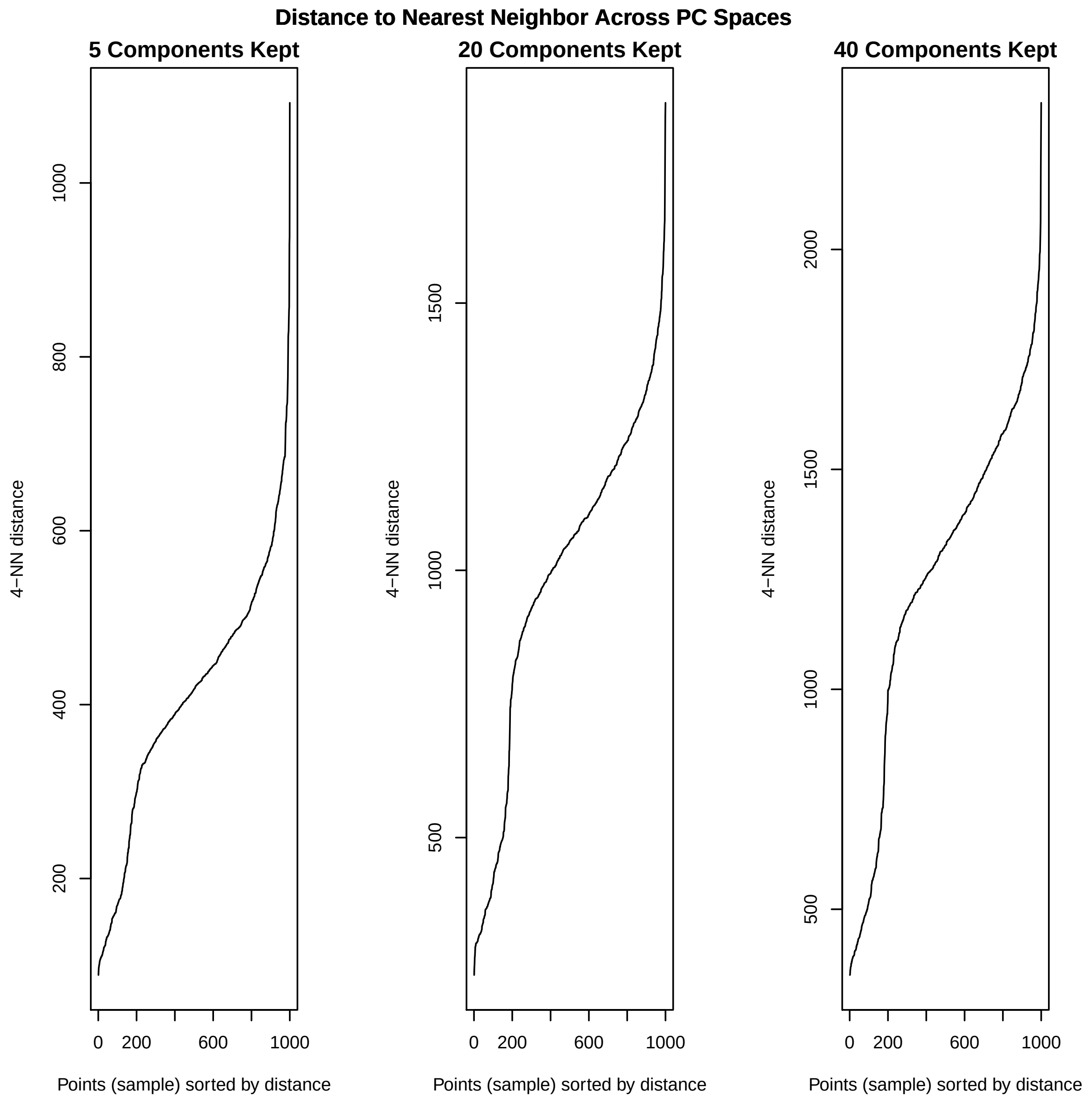
CH Index Single Linkage Clustering Across PC Spaces
5 Components Kep
20 Components Kep
40 Components Kep
CH Index



CH Index Average Linkage Clustering Across PC Spaces
5 Components Kep
20 Components Kep
40 Components Kep
CH Index

D Grid for DBSCAN

The grid for DBSCAN was chosen by following the advice of Rahmah and Sitanggang (2016). Namely, the 4 nearest neighbor distances were sorted in ascending order and plotted. Then, a grid is layed down at the point of maximum curvature in this plot, as this point of curvature indicates when the distances between observations is rapidly shifting.


Distance to Nearest Neighbor Across PC Spaces
5 Components Kept
20 Components Kept
40 Components Kept
4−NN distance
Points (sample) sorted by distance