CS181-S22

Assignment #4
Due: 11:59pm EST, March 25, 2022

Homework 4: SVM, Clustering, and Ethics

Introduction

This homework assignment will have you work with SVMs, clustering, and engage with the ethics lecture. We encourage you to read Chapters 5 and 6 of the course textbook.

Please submit the **writeup PDF to the Gradescope assignment 'HW4'**. Remember to assign pages for each question.

Please submit your **LaTeX file and code files to the Gradescope assignment 'HW4 - Supplemental'**.

Problem 1 (Fitting an SVM by hand, 10pts)

For this problem you will solve an SVM by hand, relying on principled rules and SVM properties. For making plots, however, you are allowed to use a computer or other graphical tools.

Consider a dataset with the following 7 data points each with $x \in \mathbb{R}$ and $y \in \{-1, +1\}$:

$$\{(x_i, y_i)\}_{i=1}^{7} = \{(-3, +1), (-2, +1), (-1, -1), (0, +1), (1, -1), (2, +1), (3, +1)\}$$

Consider mapping these points to 2 dimensions using the feature vector $\boldsymbol{\phi}(x) = (x, -\frac{8}{3}x^2 + \frac{2}{3}x^4)$. The hard margin classifier training problem is:

$$\begin{aligned} &\min_{\mathbf{w}, w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad &y_i(\mathbf{w}^\top \boldsymbol{\phi}(x_i) + w_0) \geq 1, \ \forall i \in \{1, \ldots, n\} \end{aligned}$$

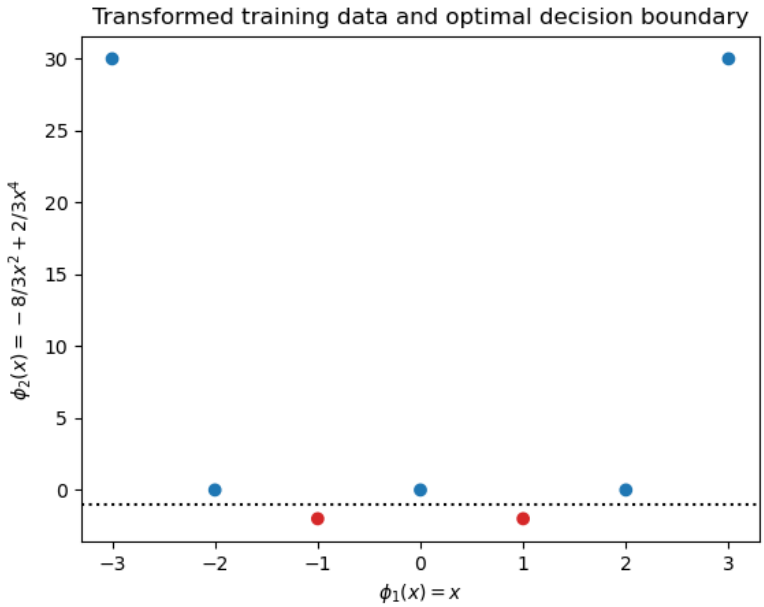
Make sure to follow the logical structure of the questions below when composing your answers, and to justify each step.

1. Plot the transformed training data in $\mathbb{R}^2$ and draw the optimal decision boundary of the max margin classifier. You can determine this by inspection (i.e. by hand, without actually doing any calculations).
2. What is the value of the margin achieved by the optimal decision boundary found in Part 1?
3. Identify a unit vector that is orthogonal to the decision boundary.
4. Considering the discriminant $h(\boldsymbol{\phi}(x); \mathbf{w}, w_0) = \mathbf{w}^\top \boldsymbol{\phi}(x) + w_0$, give an expression for *all possible* $(\mathbf{w}, w_0)$ that define the optimal decision boundary from 1.1. Justify your answer.

   Hint: The boundary is where the discriminant is equal to 0. Use what you know from 1.1 and 1.3 to solve for $\mathbf{w}$ in terms of w_0. (If you solve this problem in this way, then w_0 corresponds to your free parameter to describe the set of all possible $(\mathbf{w}, w_0)$.)
5. Consider now the training problem for this dataset. Using your answers so far, what particular solution to $\mathbf{w}$ will be optimal for the optimization problem?
6. What is the corresponding optimal value of w_0 for the $\mathbf{w}$ found in Part 5 (use your result from Part 4 as guidance)? Substitute in these optimal values and write out the discriminant function $h(\boldsymbol{\phi}(x); \mathbf{w}, w_0)$ in terms of the variable x .
7. Which points could possibly be support vectors of the classifier? Confirm that your solution in Part 6 makes the constraints above tight—that is, met with equality—for these candidate points.
8. Suppose that we had decided to use a different feature mapping $\boldsymbol{\phi}'(x) = (x, -\frac{31}{12}x^2 + \frac{7}{12}x^4)$. Does this feature mapping still admit a separable solution? How does its margin compare to the margin in the previous parts? Based on this, which set of features might you prefer and why?

Solution

1. By inspection, the optimal decision boundary of the max margin classifier is the line where the second dimension is equal to -1.



2. The value of the margin is 1, since the datapoints with $-2 \leq x_i \leq 2$ are all distance 1 from the optimal decision boundary.

3. The unit vector $(0,1)^\top$ is orthogonal to the decision boundary.

4. The decision boundary is the line where the discriminant is equal to 0, which occurs here when $\phi_2(x) = -1$. Plugging this in to the discriminant equation,

$$\begin{aligned} 0 &= h(\boldsymbol{\phi}(x); \mathbf{w}, w_0) \\ &= \mathbf{w}^\top \boldsymbol{\phi}(x) + w_0 \\ &= w_1\phi_1(x) + w_2\phi_2(x) + w_0 \\ &= w_1\phi_1(x) - w_2 + w_0. \end{aligned}$$

Since $\mathbf{w}$ is orthogonal to the decision boundary and thus parallel to $(0,1)^\top$, we have $w_1 = 0$, leaving

$$w_2 = w_0.$$

Therefore, the set of all possible $(\mathbf{w}, w_0)$ that define the optimal decision boundary is

$$\{(\mathbf{w}, w_0) : w_1 = 0, w_2 = w_0\}.$$

5. The training problem is

$$\min_{\mathbf{w}, w_0} \frac{1}{2}||\mathbf{w}||_2^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^\top \phi(x_i) + w_0) \geq 1, \forall i \in \{1, \ldots, n\}.$$

From Part 4, we must have $w_1 = 0$ and $w_2 = w_0$, so the constraint equation reduces to

$$y_i(w_2\phi(x_i) + w_2) \geq 1$$

$$y_i w_2(\phi(x_i) + 1) \geq 1.$$

Examining the training set, the minimal w_2 that satisfies this constraint for all training points is $w_2 = 1$, giving

$$\mathbf{w} = (0,1)^\top.$$

6. From Part 4, the corresponding optimal value of w_0 is simply

$$w_0 = w_2 = 1.$$

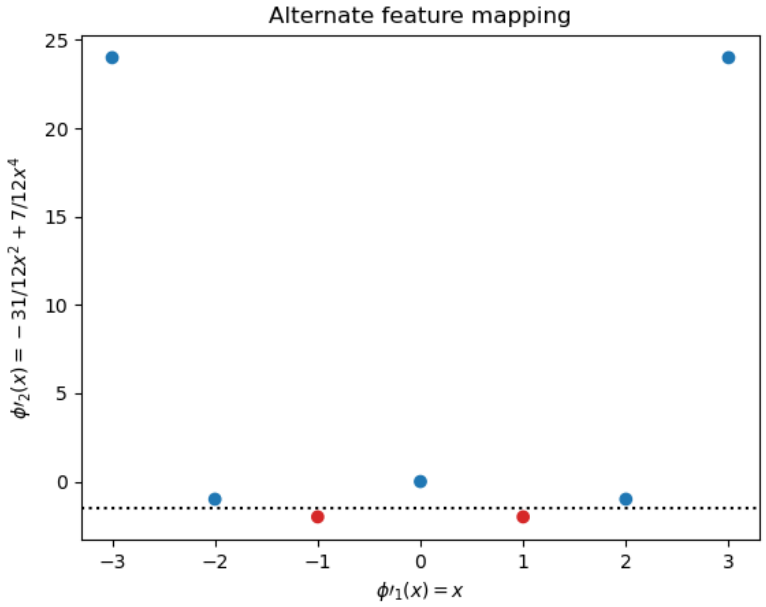
This makes the discriminant function

$$\begin{aligned} h(\boldsymbol{\phi}(x); \mathbf{w}, w_0) &= \phi_2(x) + 1 \\ &= -\frac{8}{3}x^2 + \frac{2}{3}x^4 + 1. \end{aligned}$$

7. Only the points $i \in \{2, 3, 4, 5, 6\}$, which have $x_i \in \{-2, -1, 0, 1, 2\}$, could possibly be support vectors of the classifier, since if either of the points $i \in \{-3, 3\}$ were on the margin boundary, some of the training points would have to be misclassified. The constraints for the possible support vector points are

$$\begin{aligned} y_2(h(\phi(-2); \mathbf{w}, w_0)) &= 1(0 + 1) &&= 1 \\ y_3(h(\phi(-1); \mathbf{w}, w_0)) &= -1(-2 + 1) &&= 1 \\ y_4(h(\phi(0); \mathbf{w}, w_0)) &= 1(0 + 1) &&= 1 \\ y_5(h(\phi(1); \mathbf{w}, w_0)) &= -1(-2 + 1) &&= 1 \\ y_6(h(\phi(2); \mathbf{w}, w_0)) &= 1(0 + 1) &&= 1 \end{aligned}$$

8. The alternate feature mapping plot is:



This still admits a separable solution, since there is space for a line between the classes, including an optimal line where the second dimension is equal to -1.5. The margin, however, is only 0.5, which is smaller than in the original feature mapping. Based on this, I would prefer the original set of features ϕ, because they allow for a larger margin and therefore a greater degree of separation between the classes, ideally making the boundary more robust to future data.

Problem 2 (K-Means and HAC, 20pts)

For this problem you will implement K-Means and HAC from scratch to cluster image data. You may use `numpy` but no third-party ML implementations (eg. `scikit-learn`).

We've provided you with a subset of the MNIST dataset, a collection of handwritten digits used as a benchmark for image recognition (learn more at http://yann.lecun.com/exdb/mnist/). MNIST is widely used in supervised learning, and modern algorithms do very well.

You have been given representations of MNIST images, each of which is a 784×1 greyscale handwritten digit from 0-9. Your job is to implement K-means and HAC on MNIST, and to test whether these relatively simple algorithms can cluster similar-looking images together.

The code in `T4_P2.py` loads the images into your environment into two arrays – `large_dataset`, a 5000x784 array, will be used for K-means, while `small_dataset`, a 300x784 array, will be used for HAC. In your code, you should use the ℓ_2 norm (i.e. Euclidean distance) as your distance metric.

Important: Remember to include all of your plots in your PDF submission!

Checking your algorithms: Instead of an Autograder file, we have provided a similar dataset, `P2_Autograder_Data`, and some visualizations, `HAC_visual` and `KMeans_visual`, for how K-means and HAC perform on this data. Run your K-means (with $K = 10$ and `np.random.seed(2)`) and HAC on this second dataset to confirm your answers against the provided visualizations. Do **not** submit the outputs generated from `P2_Autograder_Data`. Load this data with `data = np.load('P2_Autograder_Data.npy')`.

1. Starting at a random initialization and $K = 10$, plot the K-means objective function (the residual sum of squares) as a function of iterations and verify that it never increases.
2. For $K = 10$ and for 3 random restarts, print the mean image (aka the centroid) for each cluster. There should be 30 total images. Code that creates plots for parts 2, 3, and 4 can be found in `T4_P2.py`.
3. Repeat Part 2, but before running K-means, standardize or center the data such that each pixel has mean 0 and variance 1 (for any pixels with zero variance, simply divide by 1). For $K = 10$ and 3 random restarts, show the mean image (centroid) for each cluster. Again, present the 30 total images in a single plot. Compare to Part 2: How do the centroids visually differ? Why?
4. Implement HAC for min, max, and centroid-based linkages. Fit these models to the `small_dataset`. For each of these 3 linkage criteria, find the mean image for each cluster when using 10 clusters. Display these images (30 total) on a single plot.

   How do the "crispness" of the cluster means and the digits represented compare to mean images for k-means? Why do we only ask you to run HAC once?

   Important Note: For this part ONLY, you may use `scipy`'s `cdist` function to calculate Euclidean distances between every pair of points in two arrays.
5. For each of the HAC linkages, as well as one of the runs of your k-means, make a plot of "Number of images in cluster" (y-axis) v. "Cluster index" (x-axis) reflecting the assignments during the phase of the algorithm when there were $K = 10$ clusters.

   Intuitively, what do these plots tell you about the difference between the clusters produced by the max and min linkage criteria?

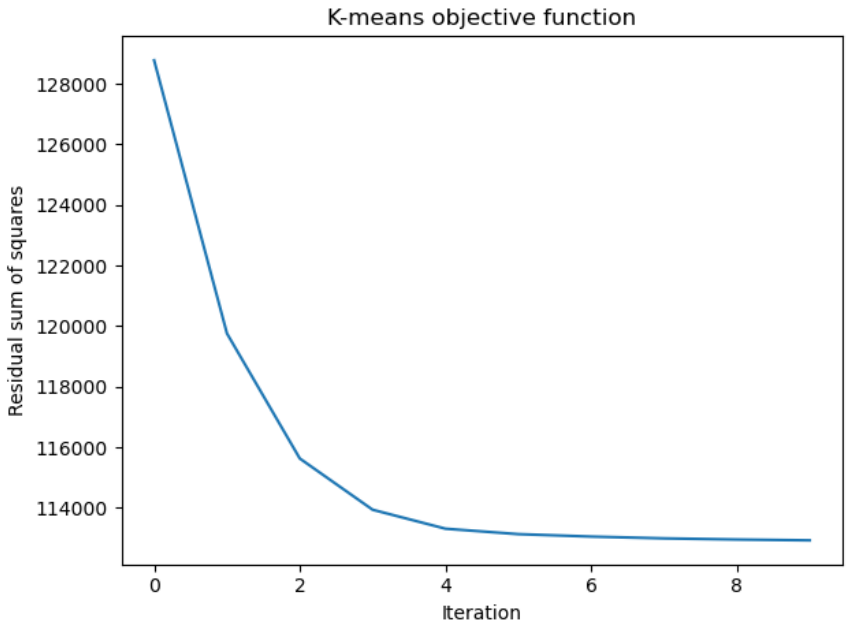
   Going back to the previous part: How does this help explain the crispness and blurriness of some of the clusters?

Problem 2 (cont.)

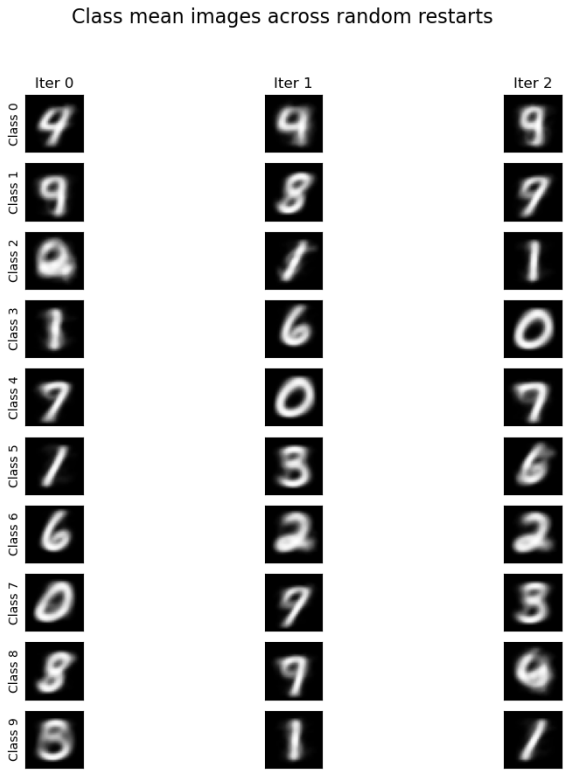
6. For your K-means with $K = 10$ model and HAC min/max/centroid models using 10 clusters on the `small_dataset` images, use the `seaborn` module's `heatmap` function to plot a confusion matrix between each pair of clustering methods. This will produce 6 matrices, one per pair of methods. The cell at the ith row, jth column of your confusion matrix is the number of times that an image with the cluster label j of one method has cluster i in the second method. Which HAC is closest to k-means? Why might that be?
7. Suppose instead of comparing the different clustering methods to each other, we had decided to compute confusions of each clustering method to the *true* digit labels (you do *not* have to actually compute this). Do you think how well the clustering match the true digits is reasonable evaluation metric for the clustering? Explain why or why not.

Solution

1. The objective function never increases:



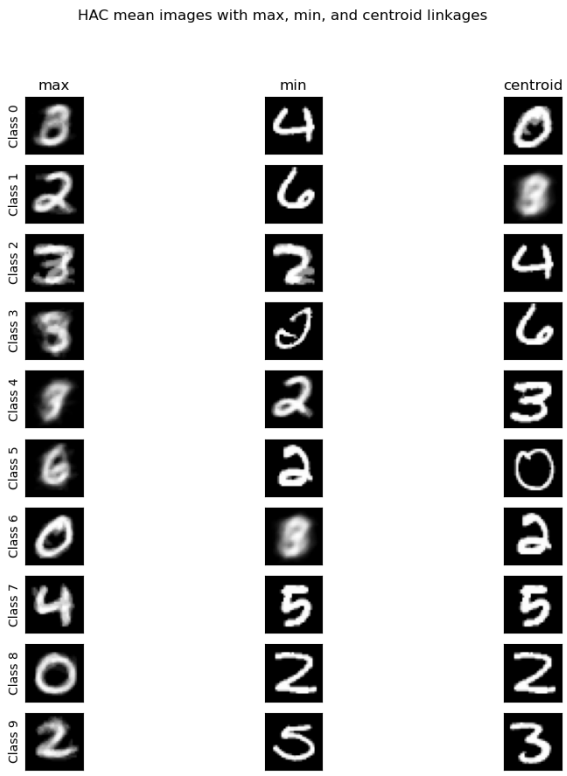
2. The mean images are:



3. The mean images, using standardized data, are:

The most obvious visual difference in comparison to the non-standardized data from Part 2 is that the outer regions of the images that were previously black are now grey, since with standardized data these regions have mean values around zero. In contrast, regions with higher variance, where some source images are white and others black, are the most extreme parts of the mean images visually. In terms of the accuracy of the actual classes generated by K-Means, the standardized data seems to be comparable to the non-standardized data, with both data sets resulting in some clear classes for digits like 0, 1, and 6, and more muddled classes for other digits.

4. The HAC mean images are:



The cluster means for max-linkage are slightly blurrier than those for K-Means, but for min- and centroid-linkage are generally much crisper than those for K-Means. However, one of the mean images for each of the min- and centroid-linkage types is quite blurry.

We only need to run HAC once since, after selecting a linkage criterion, it is a deterministic algorithm, meaning the clusters would be the same in every iteration.

5. See Figure 1.

   Intuitively, the plots show that the max-linkage criterion produced a variety of clusters of unequal, but still considerable, sizes. By contrast, the min-linkage criterion produced one cluster containing almost all of the images, with the rest of the clusters being much smaller. Presumably the small clusters contain those images which are farthest away from any other image.

   Going back to the previous part, this explains why many of the mean images in the min- and centroid-linkage models were very crisp, while one image was quite blurry. The crisp images are because many of the classes in these models contain only one or a few images, all of which are quite similar to each other, giving them a very crisp mean. By contrast, the class with most of the images has a very blurry mean, since it is comprised of images of many different digits. In a less extreme sense, the same logic helps explain why some of the max-linkage model mean images are crisper than others.

6. See Figure 2.

   Max-linkage HAC is the closest to K-Means, which we can see from the non-negligible number of shared images for each class. By contrast, the min- and centroid-linkage HAC models have large vertical bars on their heatmaps which indicate that almost all of their images are in the same class, regardless of

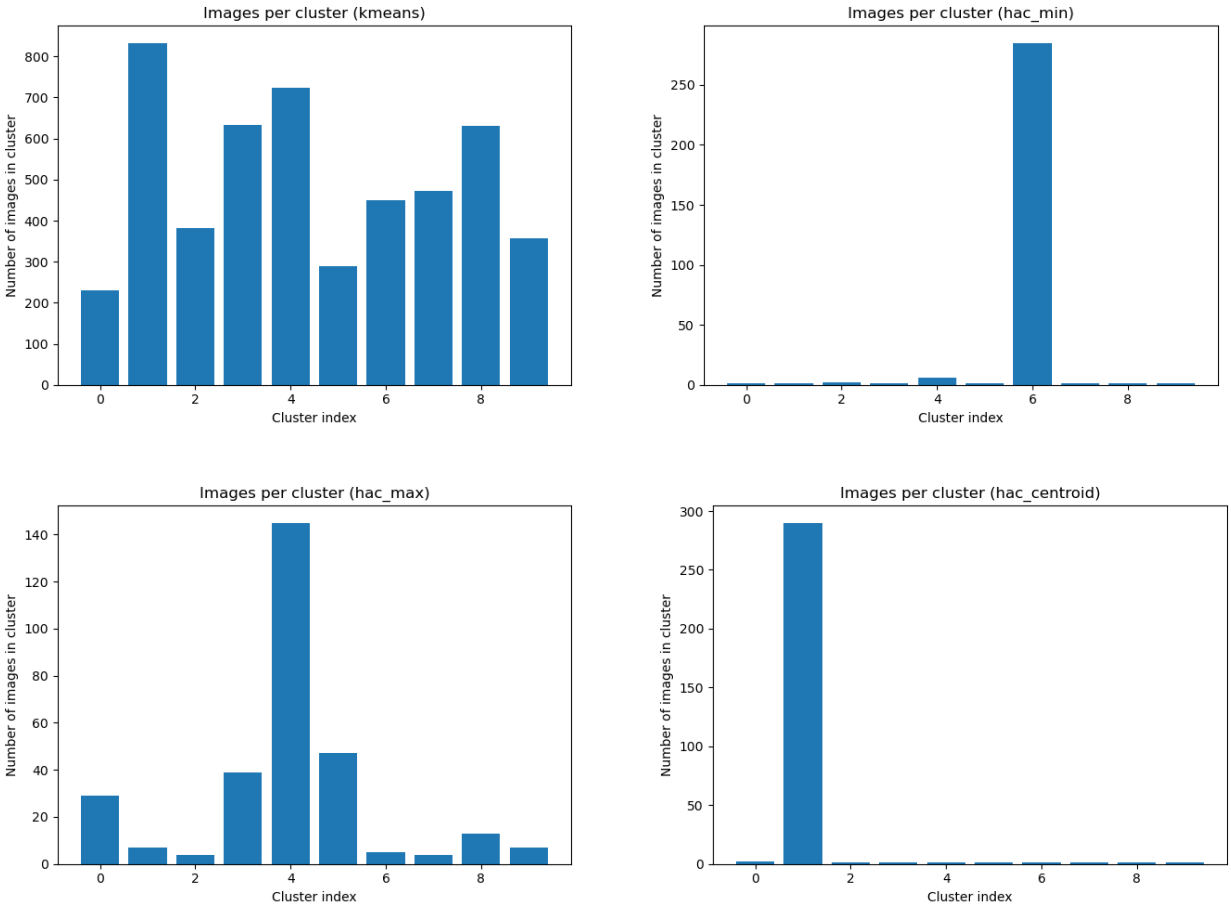


Figure 1: Number of images per cluster

the class K-Means puts them in.

Max-linkage HAC is closest to K-Means because K-Means tends to group the data points into clusters that are hypersphere-like in order to minimize the distances of points to their means. Similarly, the HAC max-linkage criterion tends to group the data points into hypersphere-like clusters in order to minimize the distance of new points to the farthest-away point in the sphere. By contrast, the min- and centroid-linkage critera can produce stringer clusters, or in this case can group most of the data points into one cluster when they are relatively evenly spread out in space, because they can start with one cluster and gradually expand it across the space to include most of the data points without incurring a substantial penalty for having a large, spread-out cluster.

7. Yes, it does seem like comparing the clusterings to the true digit labels is a reasonable evaluation metric since our goal from the outset was to find a clustering method that could correctly classify the images into digits as well as possible. Comparing against the true digit labels would give us an empirical measure of how well each clustering method identifies the true clusters of images into digits.

Problem 3 (Ethics Assignment, 5pts)

Select a real-life outcome in Artificial Intelligence or Machine Learning that you believe is morally wrong. You can select your own outcome from the news or select one of the outcomes in the options below:

- COMPAS, a case management tool predicting recidivism that flagged "blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend" (Angwin 2016).
- An NLP algorithm filled in the inference "Man is to ____ as woman is to ____" with "Man is to computer programmer as woman is to homemaker" (Bolukbasi et al, 2016).
- http://www.survivalofthebestfit.com/game: a game that exemplifies algorithmic bias in resume screening
- IBM Diversity in faces: insufficient training data for darker-skinned faces
- Other Unfair Algorithms: Algorithms of Oppression (a really good book with tons of examples), VI-SPDAT, Allegheny Family Screening Tool

Draw a causal chain that resulted in this outcome and circle the choice points that were the largest contributors to the outcome. At each morally relevant choice point, write two alternative decisions that could have prevented the outcome.

Solution

Outcome: An NLP algorithm filled in the inference "Man is to ____ as woman is to ____" with "Man is to computer programmer as woman is to homemaker" (Bolukbasi et al, 2016).

We present the causal chain as a list in chronological order. The choice point that is the largest contributor to the outcome is underlined. Morally relevant choice points are followed by alternative decisions.

1. Development of natural language processing algorithms for applications like speech recognition and generation.
2. <u>The concept of word embedding is developed. Word embeddings transform words into vectors in a high-dimensional space that relates words by similarity along various dimensions.</u>
   (a) Alternative 1 (Naive): Don't use word embeddings, and instead use "one-hot" or "bag of words" representations of text. This avoids word representations having any gender bias, because the representations of words have no relationship to their meaning. However, this denies us the important benefits of word embeddings, not only for comparing similar words, but also for ease of NLP training and the representation of large vocabularies.
   (b) Alternative 2: Recognize that word embeddings have the potential to encode both actual similarities and stereotypical similarities, and proceed to develop the theory of word embeddings with this in mind.
3. Embeddings like word2vec train on large corpora using "second-order" techniques, which uncover both implicit meaning and implicit bias in texts, rather than "first-order" techniques, which simply count the number of times words occur near each other. First-order techniques might not, for example, not consider the word "nurse" feminine, since the phrase "male nurse" occurs more often in the world than "female nurse," because the "female" part is often left implicit when describing a "female nurse." Second-order techniques uncover the implicit bias and consider the word "nurse" feminine.
   (a) Alternative 1 (Naive): Use only first-order techniques. However, as in the example of "nurse," this could create bias in the other direction, where "nurse" becomes associated with males. Additionally, first-order techniques severely limit the expressivity of the resulting embeddings.

(b) Alternative 2: Attempt to hardcode gender-neutrality into training algorithms for word embeddings, so that bias is removed in the training process.

4. Researchers train use word2vec to train an embedding called w2vNEWS on a corpus of Google News texts consisting of 3 million English words, placing them in 300 dimensions. Even though the news articles are primarily written by professional journalists and should ideally exhibit little gender bias, bias still occurs in the embeddings.

   (a) Alternative 1 (Naive): Systematically review the corpus by hand for gender bias, both implicit and explicit, and edit articles to remove it. However, this could be impractically time consuming. Additionally, there are no guarantees that the human reviewers would find or successfully correct all bias due to their own biases. It would not be possible to carry this process out using ML, since the reviewing model would itself have bias if not trained on unbiased data.

   (b) Alternative 2: De-bias word embeddings after training using the techniques developed by Bolukbasi et al. This is probably the best alternative of all those considered. In the future, this de-biasing process could hopefully be automatically applied to all word embeddings after they are generated to make all future embeddings free of gender- and other-biases.

5. An NLP algorithm filled in the inference "Man is to ____ as woman is to ____" with "Man is to computer programmer as woman is to homemaker"

Name

Alex Encalada-Stuart

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?

No one and no

Did you attend office hours for help with this homework?

No

Calibration

Approximately how long did this homework take you to complete (in hours)?

16

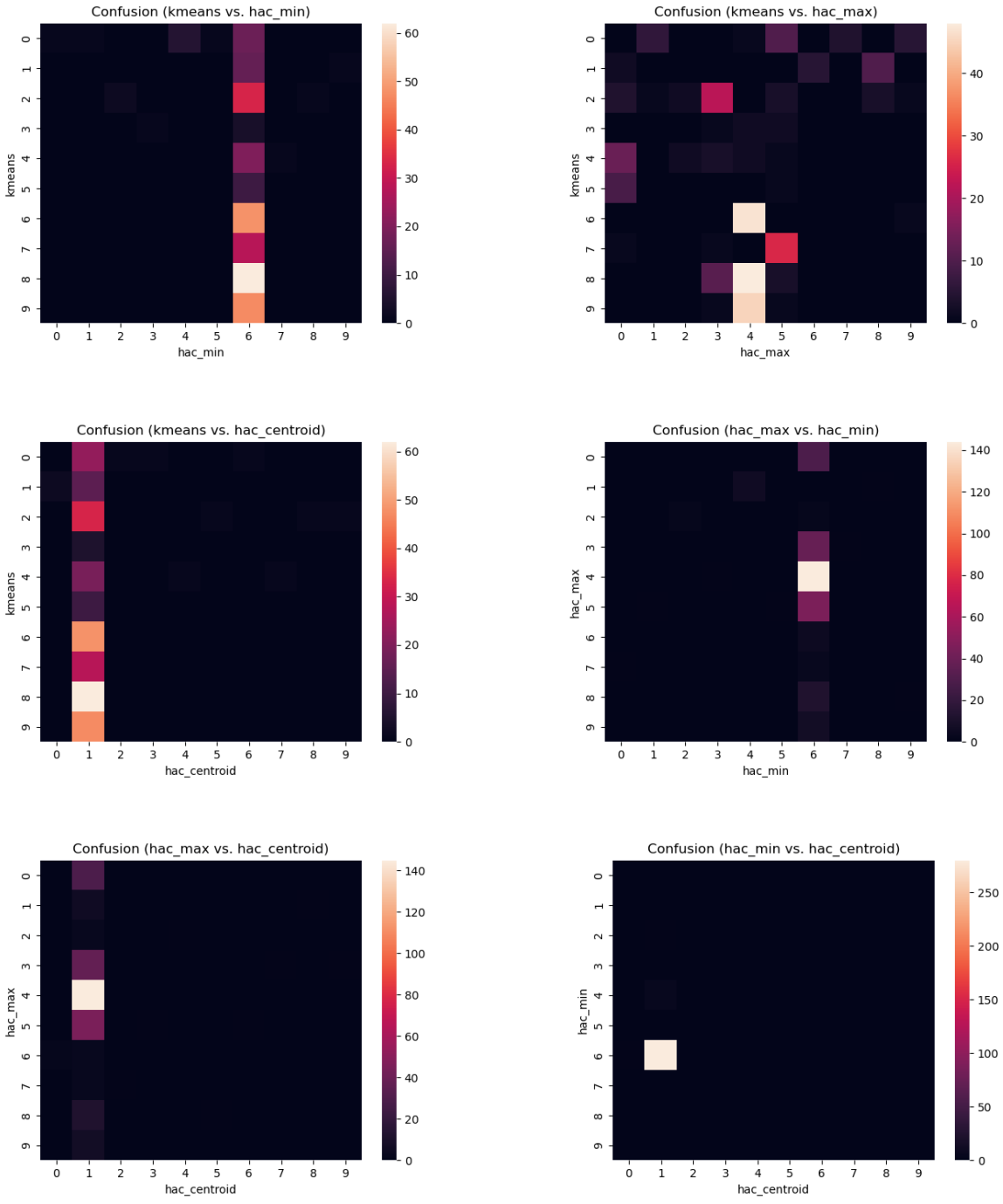


Figure 2: Confusion matrices