0.1 Q1.1 Mixture of Distributions

Consider a mixture distribution of the form

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|k)$$

where you may assume the elements of x are discrete, and $\vec{\pi}$ is a k dimensional vector that satisfy the following condition $\sum_k \pi_k = 1$. Denote the mean and covariance of $p(x|k)$ by μ_k and Σ_k, respectively. Show that the mean of the mixture distribution is given by the following equation:

$$\mathbb{E}[\mathrm{x}] = \sum_{k=1}^{K} \pi_k \mu_k$$

To get there you may want to start with the definition of the mixture $p(x)$ above. Reminder, the generic definition of expectation:

$$\mathbb{E}[\mathrm{x}] = \sum_x x p(x)$$

Points: 0.3

$$\begin{aligned}
p(x) &= \sum_{k=1}^{K} \pi_k p(x \mid k) \\
\mathbb{E}[\mathrm{x}]_k &= \mu_k = \sum_x x p(x \mid k) \\
\mathbb{E}[\mathrm{x}] &= \sum_x x p(x) \\
&= \sum_x x \sum_{k=1}^{K} \pi_k p(x \mid k) \\
&= \sum_{k=1}^{K} \pi_k \sum_x x p(x \mid k) \\
&= \sum_{k=1}^{K} \pi_k \mu_k
\end{aligned}$$

Illustration of Bernoulli Mixture Model on MNIST dataset We illustrate the Bernoulli mixture model to model handwritten digits from the MNIST dataset. Here the digit images have been turned into binary vectors by setting all elements whose values exceed 0.5 to 1 and setting the remaining elements to 0. A sample of this binary-valued MNIST is shown below for just the digits 2, 3, and 4. In other words, the only two possible pixel intensities of the MNIST is now 0 (fully black) and 1 (fully white). In this setting, we can consider each individual pixel as a bernoulli random variable.

This approach would fit each digit with a Bernoulli parameter vector as long as the number of pixels in an image. And we'd converge to the correct parameter vectors by running iterations of the EM algorithm.

0.2 Q1.2 Mixture of Bernoulli Distribution

Consider the joint distribution of latent and observed variables for the Bernoulli distribution obtained by forming the product of $p(x|z, \mu)$ (given by Bishop equation (9.52)):

$$p(x|z, \mu) = \prod_{k=1}^{K} p(x|\mu_k)^{z_k}$$

and $p(z|\pi)$ (given by Bishop equation (9.53)):

$$p(z|\pi) = \prod_{k=1}^{K} \pi_k^{z_k}$$

Show that if we marginalize this joint distribution with respect to z, then we obtain the following equation:

$$p(x|\mu, \pi) = \sum_{k=1}^{K} \pi_k p(x|\mu_k)$$

Hint: you might want to read the pages of Bishop noted above. Also when we say "marginalize with respect to z", you should review what that means and start with the definition of marginalizing over a latent variable.

Points: 0.3

$$
\begin{aligned}
p(x|z,\mu) &= \prod_{k=1}^{K} p(x|\mu_k)^{z_k} \\
p(z|\pi) &= \prod_{k=1}^{K} \pi_k^{z_k} \\
\text{Joint Propability : } p(x,z|\pi,\mu) &= \prod_{k=1}^{K} \pi_k^{z_k} p(x \mid \mu_k)^{z_k} \\
\text{Marginal Probability W.R.T. Z: } p(x|\pi,\mu) &= \sum_z p(x, z \mid \pi, \mu) \\
&= \sum_z \prod_{k=1}^{K} \pi_k^{z_k} p(x \mid \mu_k)^{z_k} \\
&= \sum_z \prod_{k=1}^{K} \pi_k^{z_k} p(x \mid \mu_k)^{z_k} \\
p(x|\pi,\mu) &= \sum_{k=1}^{K} \pi_k p(x \mid \mu_k)
\end{aligned}
$$

Z is one hot encode, so only z_k where k=label is equal to 1}

Turns into a summation because after taking the product with respect to K, you find that the element z_k is always 0 unless K=target_label which will make $z_k = 1$. So when you marginalize with respect to z, you find that the different permutations of z will make each permutation term equal to $\pi_k p(x \mid \mu_k)$. At the end, you sum all of these terms with respect to K because there are K permutations of latent variable z.

0.3 Q1.3 Mixture of Bernoulli Distribution M-step

Show that if we maximize the expected complete-data log likelihood function (Bishop equation (9.55))

$$\mathbb{E}_z[\ln p(\mathrm{Z},\mathrm{X}|\mu,\pi)] = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left\{\ln \pi_k + \sum_{i=1}^{D}[x_{ni}\ln \mu_{ki} + (1-x_{ni})\ln(1-\mu_{ki})]\right\}$$

for a mixture of Bernoulli distributions with respect to μ_k, we obtain the M-step equation below:

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})x_n,$$

that is, the k-th mean's MLE is a $\gamma(z_{nk})$ weighted mean of the entire dataset.

Hint: you will want to start with the idea that we always go to when we want to maximize a convex function (as log likelihood is indeed convex). Make sure you are doing this process with respect to the variable you are trying to maximize.

Points: 0.4

$$\mathbb{E}_z[\ln p(\mathrm{Z},\mathrm{X}|\mu,\pi)] = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left\{\ln \pi_k + \sum_{i=1}^{D}[x_{ni}\ln \mu_{ki} + (1-x_{ni})\ln(1-\mu_{ki})]\right\}$$

$$\frac{\mathbb{E}_z[\ln p(\mathrm{Z},\mathrm{X}|\mu,\pi)]}{\partial \mu} = 0$$

0.4 Question 2.1 Update Rule for μ_k

Once we minimizing J w.r.t. r_{nk} while fixing μ_k, we will continue to minimize J w.r.t. μ_k fixing r_{nk} fixed. Prove that

$$\hat{\mu}_k = \arg\min_{\mu_k} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

Hint: Take the derivative of J w.r.t. μ_k, and find the critical point.

Points: 0.4

$$\hat{\mu}_k = \arg\min_{\mu_k} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

$$||x_n - \mu_k||^2 = (x_n - \mu_k)^2$$

$$\frac{||x_n - \mu_k||^2}{\partial \mu} = \frac{(x_n - \mu_k)^2}{\partial \mu} = -2(x_n - \mu_k)$$

This summation: $\sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 = r_{nk} ||x_n - \mu_k||^2$ where k=correct_cluster

$$\frac{\partial \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2}{\partial \mu} = -2 r_{nk}(x_n - \mu_k)$$

$$\frac{\partial \hat{\mu}_k}{\partial \mu} = 0 = \sum_{n=1}^{N} -2 r_{nk}(x_n - \mu_k)$$

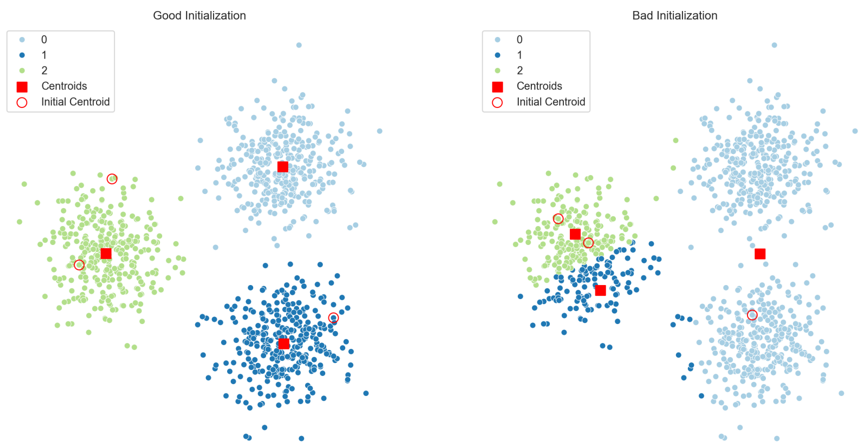
$$0 = \sum_{n=1}^{N} -2 r_{nk}(x_n - \mu_k) = -2(\sum_n r_{nk} x_n - \sum_n r_{nk} \mu_k)$$

$$\sum_n r_{nk} x_n = \sum_n r_{nk} \mu_k$$

$$\frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} = \mu_k$$

0.5 Question 2.3 Effects of Centroid Initialization

Take a look at the K-Mean cluster.



What is the differences between the clusters in the left and the clusters in the right? In what ways does the initialization of the centroids affect the clusters we got?

Points: 0.4

The differences are that the left side has the most correct centroids while the right side has incorrect centroids. Initializations do matter because if this were to be related to a minimization of a loss function. It would seem that we initialized near a local minima on the right side and initialized closer to the global minima on the left side.

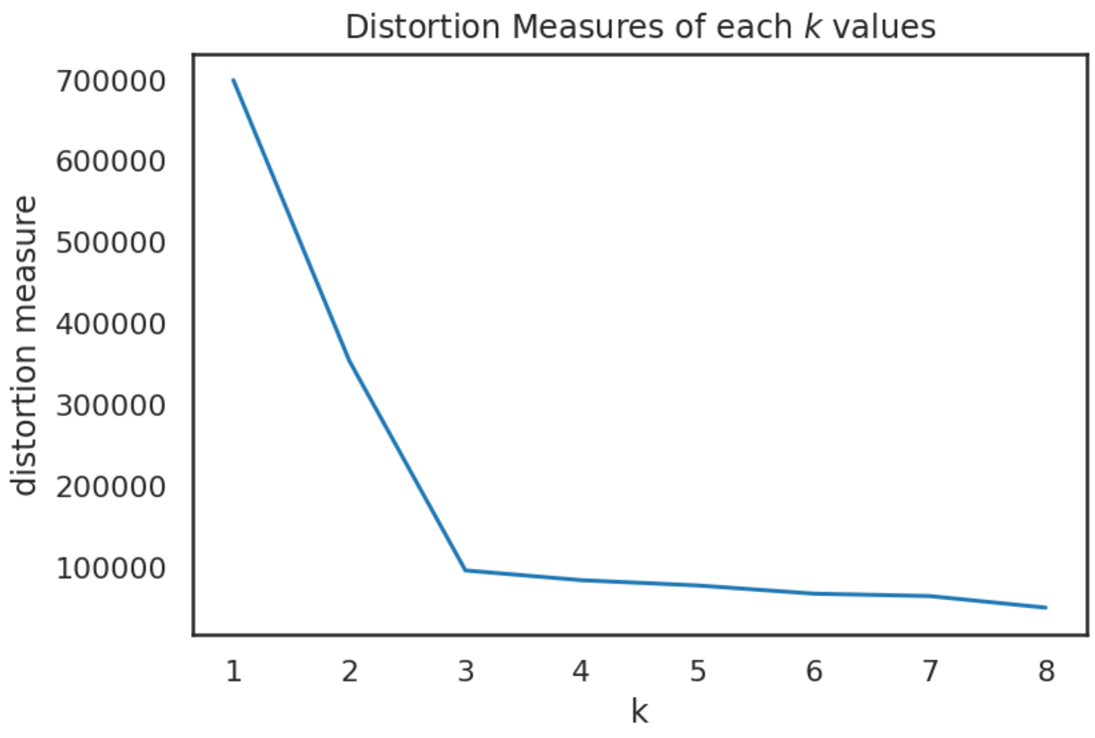
0.5.1 Question 2.4.2 Choose the Right K

In real world scenarios, we as data scientists oftentimes do not have access to the knowledge of the true *k* in the data generating process. Therefore, in this section, we explore how to choose the right value of the *k* to achieve the desired clusters.

```
In [9]: # Fit K-Means on different values of k
        costs = []
        for k in range(1, 9):
            # instantiate KMeans Object
            km = MyKMeans(k=k)
            # Get out the membership array
            labels = km.fit(points)
            costs.append(distortion_measure(points, km.centroids))
```

```
In [10]: # Visualize the Distortion Measure.
         plt.plot(range(1, 9), costs)
         plt.xlabel("k")
         plt.ylabel("distortion measure")
         plt.title("Distortion Measures of each $k$ values")
```

```
Out[10]: Text(0.5, 1.0, 'Distortion Measures of each $k$ values')
```



If your implementations is sound, you should have a plot looks like this.

Which is the right K to choose? And why?

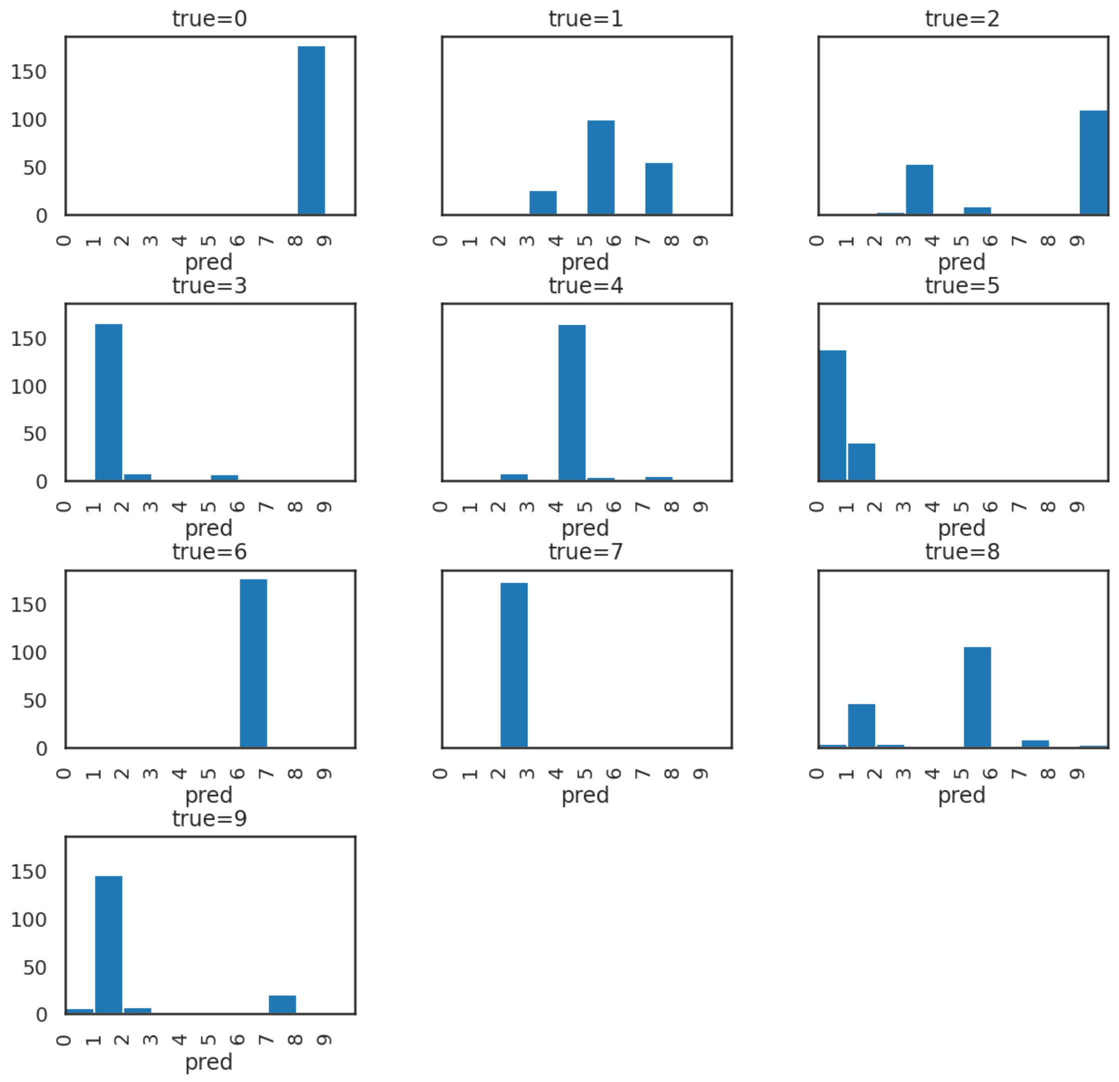
Points: 0.2

I would choose 8 because it has the lowest distortion.

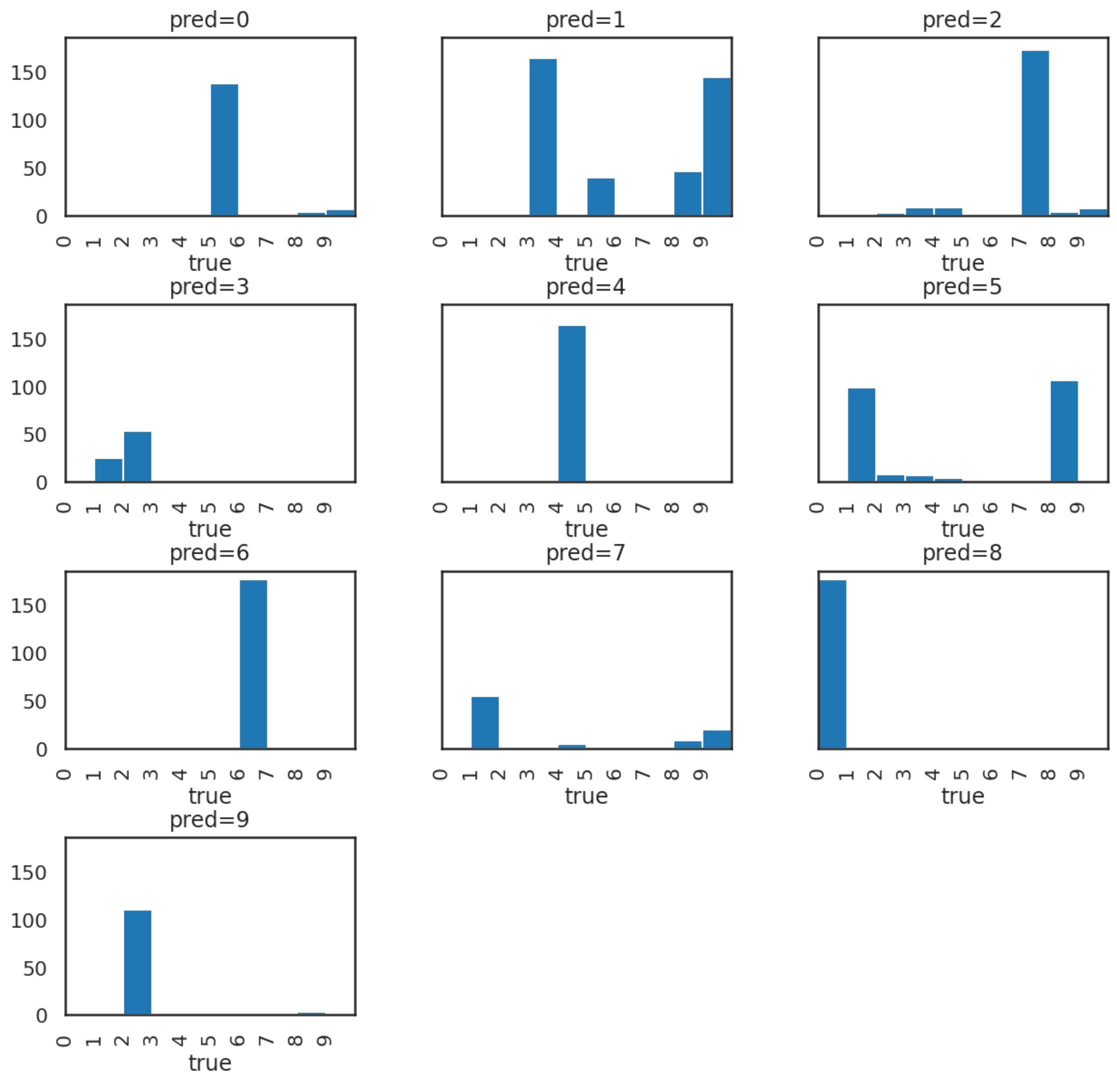
0.5.2 Q3.2 Interpret the result

The next two cells have visualizations that you can do to see what is going on with the predicted clusters and how they relate to the true labels and vice versa. Take a look.

```
In [14]: axs=kmeans_pred.hist(column='pred',by='true',sharey=True, figsize=(10,10), bins=range(11));
         for ax in axs.ravel():
             ax.set_xlim((0,10))
             ax.set_xticks(range(10))
             old=ax.get_title()
             ax.set_title('true='+old)
             ax.set_xlabel('pred')
```



```
In [15]: axs=kmeans_pred.hist(column='true',by='pred',sharey=True, figsize=(10,10), bins=range(11));
         for ax in axs.ravel():
             ax.set_xlim((0,10))
             ax.set_xticks(range(10))
             old=ax.get_title()
             ax.set_title('pred='+old)
             ax.set_xlabel('true')
```



OK, now in the next cell discuss your results above.

Please give us some insights… think deeply about the plots and empirical results in light of what you know about the clustering method, the metrics, and the visualizations. If you just copy/paste some definitions and say "yes/no" you will not be getting full points.

1. Describe what Rand score and Adjusted Rand score are. I suggest you look at the Wikipedia page as well as the scikit-learn docs. Describe the difference between the metrics. Interpret the numbers you got for Rand vs Adjusted Rand in light of what the difference is in the metrics.
2. Do you think that the clustering produced by KMeans is any good for predicting the true label? Why or why not… please include evidence from the viz and the scores above.
3. Regardless of what you said about the prediction above, do you think that the clustering can tell you anything about the data… like are there particular true labels that are MORE likely to be confused with others? What else might we learn from the clustering?

Points: 0.6

1) The rand score is a scoring from 0 to 1 that describes the basic similarity of 2 different clusterings. A score of 0 means they are completely different while a score of 1 means they are exactly the same. The adjusted rand score is an extension of the rand score that fixes a flaw in rand score's method. The flaw is that rand score's method doesn't take into account that the similarity could be by chance. Adjusted rand score accounts for the probability of the 2 clusterings randomly agreed with each other with a scoring between -1 and 1. A score of -1 means that the clusterings could be negatively correlated, a score of 0 would mean that they are randomly agree with each other by chance, and 1 means that they are perfectly the same. The rand score is 0.9204799387248979, which we can interpret as the clustering from our predictions as being almost the same as the optimized solution of the sklearn library's implementation. The adjusted rand score is 0.596907083127223, which is close to saying that these 2 clusterings agree with each other by chance but leans more towards the belief that these clusterings are similar.

2) I think it does predict well because inside the graphs for predicting 0, 2, 4, 6, 7, 8 and 9 there is a visually consistent labeling for that label inside a specific cluster, a cluster that stands out more than the others. For example, when predicting a label=2 there are clusters that have label=2 assigned to them but are insignificant compared to cluster 7's grouping of the points that are labeled 2.

3) The clustering can tell us that labels 3 and 9 look similar when looking at them as data samples or the euclidean distances between cluster means for points labeled 3 or 9 can be predicted as each other because their points are close to each other. I also believe that labels 1 and 5 are the most likely label to be confused due to the number of points labeled as 1 or 5 are not in consistent enough clusters.

0.6 Q4.2 Interpret the results of K-means from Q3.1

According to the result of Q3.1, does K-means fit well to this new dataset? Why or why not? Explain.

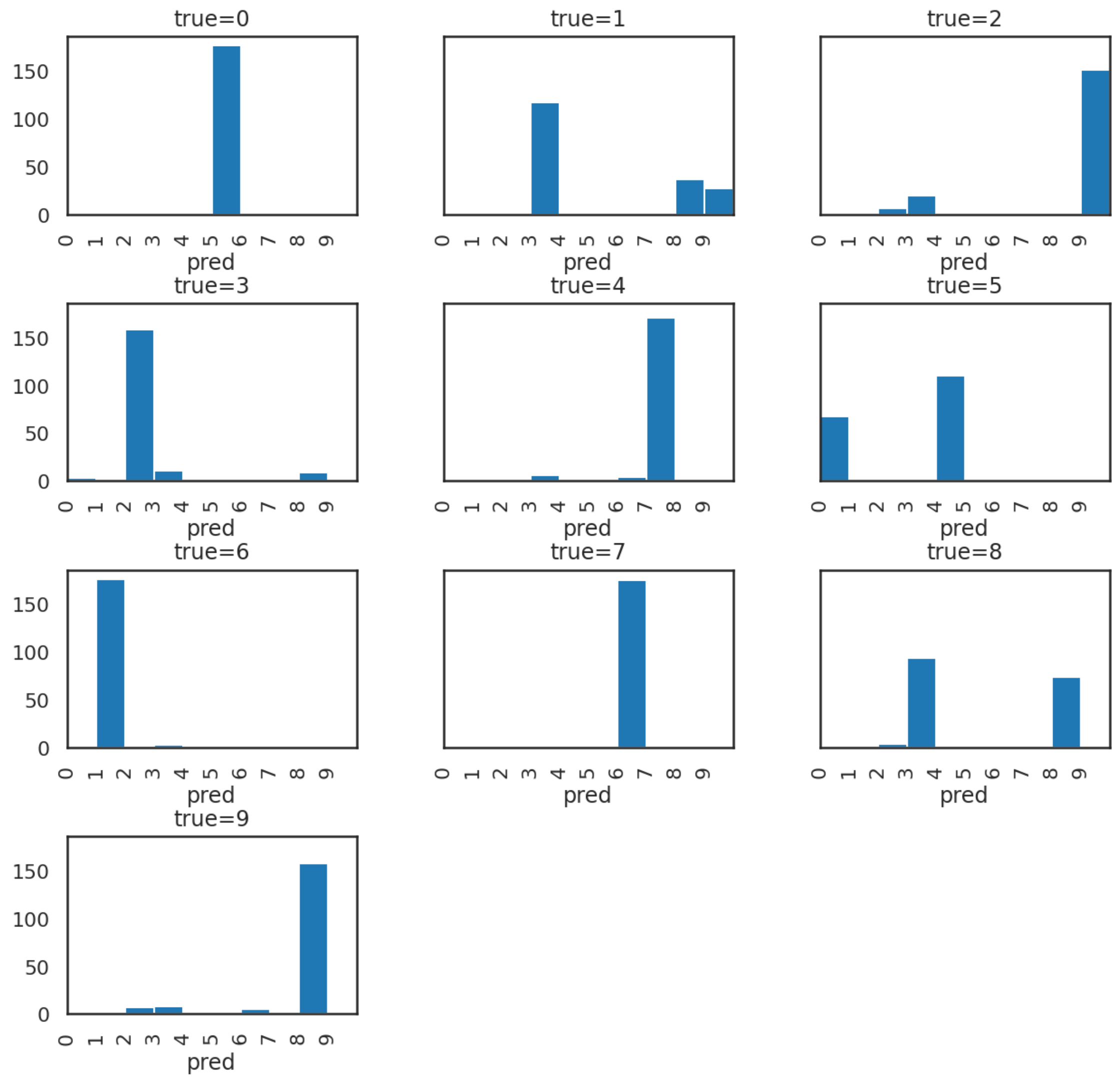
Points: 0.6

I would say this K-means fits well because for the most part, the clusterings are correct and a small amount of points, small in comparison to the whole dataset, or in the wrong cluster. Looking at the visuals, around 70% accurate.

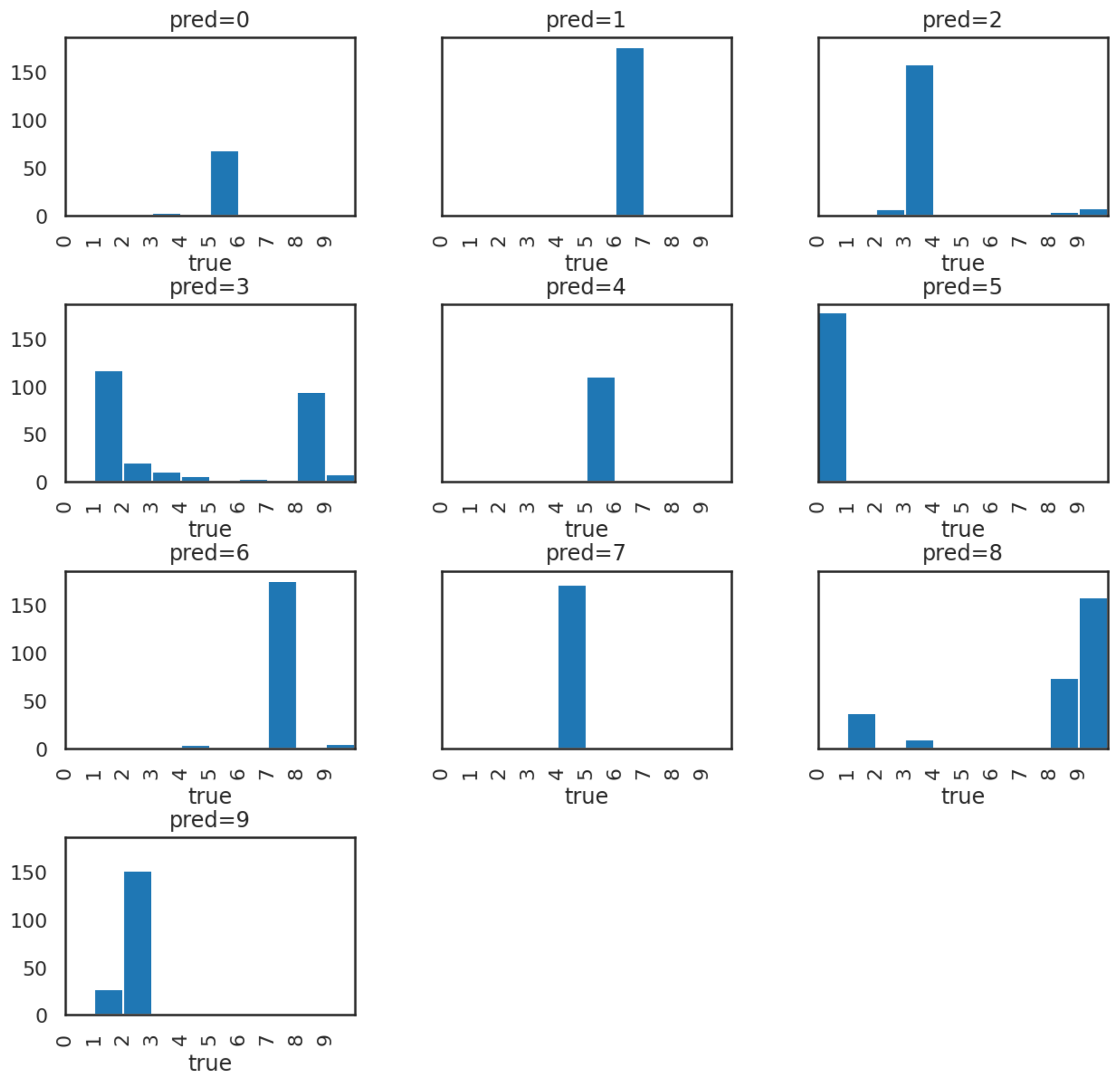
0.7 Q5.2 Interpret the GMM results

The next two cells have visualizations that you can do to see what is going on with the predicted clusters and how they relate to the true labels and vice versa. Take a look.

```
In [29]: axs=gmm_pred.hist(column='pred',by='true',sharey=True, figsize=(10,10), bins=range(11));
         for ax in axs.ravel():
             ax.set_xlim((0,10))
             ax.set_xticks(range(10))
             old=ax.get_title()
             ax.set_title('true='+old)
             ax.set_xlabel('pred')
```



```
In [30]: axs=gmm_pred.hist(column='true',by='pred',sharey=True, figsize=(10,10), bins=range(11));
         for ax in axs.ravel():
             ax.set_xlim((0,10))
             ax.set_xticks(range(10))
             old=ax.get_title()
             ax.set_title('pred='+old)
             ax.set_xlabel('true')
```



OK, now in the next cell discuss your results above.

Please give us some insights… think deeply about the plots and empirical results in light of what you know about the clustering method, the metrics, and the visualizations. If you just copy/paste some definitions and say "yes/no" you will not be getting full points.

1. Do you think that the clustering produced by the Mixture Model is any good for predicting the true label? Why or why not… please include evidence from the viz and the scores above. Compare the clustering produced by the Mixture Model with the one produced by the KMeans.
2. Given the covariance argument passed to the Mixture Model, and the fact that the Mixture Model produced better results, what does that imply about the strucuture of the problem?

Points: 0.6

1) I think the mixture model is better, because there are more consistent labeling to clusters. What I mean by this is that there aren't as many cases of a certain label being a part of multiple clusters instead of just one consistent one that strongly suggests that whatever point in this cluster is highly likely to be this label N.
2) The problem was structured to be a case where the data could be described best as a mixture of Gaussian models. While a K-Means model could predict up to a certain accuracy, it wouldn't do better than a mixture model. Meaning the data itself was structured in a manner of mixture models.