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

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

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

$$p(x|\mu,\pi) = \sum_z p(x|z,\mu)\cdot p(z|\pi) = \sum_z \left(\prod_{k=1}^{K} \pi_k^{z_n}\right)\left(\prod_{k=1}^{K} p(x|\mu_k)^{z_k}\right) = \sum_z \prod_{k=1}^{K} \pi_k^{z_n} p(x|\mu_k)^{z_k} = \sum_{k=1}^{K} \pi_k p(x|\mu_k)$$

The last equality follows from z being a vector of binary values, where only one z_k can have a value of 1 at any given time (the other values are $0s$).

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

$$\begin{aligned}
0 &= \frac{\partial \mathbb{E}_z[\ln p(\mathrm{Z},\mathrm{X}|\mu,\pi)]}{\partial \mu_{ki}} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left\{\sum_{i=1}^{D}\frac{x_{ni}}{\mu_{ki}} - \frac{1-x_{ni}}{1-\mu_{ki}}\right\} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left\{\sum_{i=1}^{D}\frac{x_{ni}(1-\mu_{ki}) + \mu_{ki}(x_{ni}-1)}{\mu_{ki}(1-\mu_{ki})}\right\} \\
&= \sum_{n=1}^{N}\gamma(z_{nk})\left\{x_n(1-\mu_k) + \mu_k(x_n-1)\right\} \\
&= \sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_k) \\
&= \sum_{n=1}^{N}\gamma(z_{nk})(x_n) - \sum_{n=1}^{N}\gamma(z_{nk})(\mu_k) \\
\mu_k &= \frac{\sum_{n=1}^{N}\gamma(z_{nk})(x_n)}{\sum_{n=1}^{N}\gamma(z_{nk})} \\
&= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})x_n
\end{aligned}$$

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

$$\begin{aligned}
\frac{\partial J}{\partial \mu_k} &= \frac{\partial \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2}{\partial \mu_k} \\
&= -2 \sum_{n=1}^{N} r_{nk} (x_n - \mu_k)
\end{aligned}$$

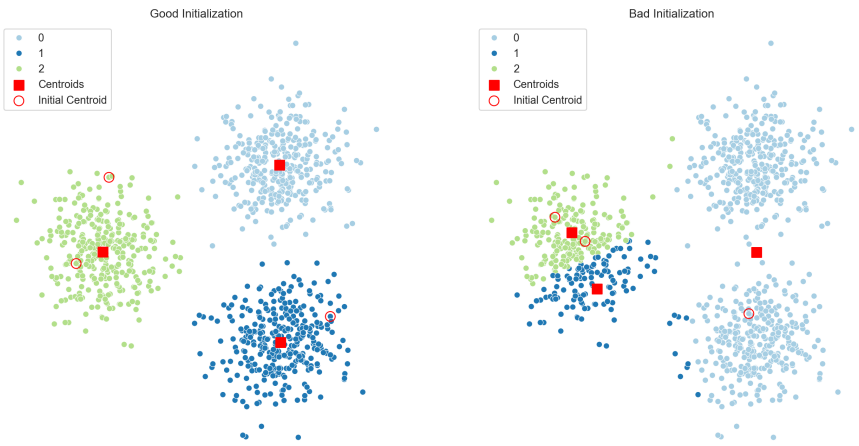
Set to zero and solve for μ_k

$$\begin{aligned}
-2 \sum_{n=1}^{N} r_{nk} (x_n - \mu_k) &= 0 \\
2 \sum_{n=1}^{N} r_{nk} x_n - 2 \sum_{n=1}^{N} r_{nk} \mu_k &= 0 \\
\sum_{n=1}^{N} r_{nk} x_n &= \sum_{n=1}^{N} r_{nk} \mu_k
\end{aligned}$$

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}}$$

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 clusters on the left are well defined, with all the points assigned to the correct clusters. By contrast, the clusters on the right are not ideal: many points are clustered incorrectly, resulting in some clusters being big and with high variance, and others being small and with lower variance. This difference is due to the initialization of the cluster centroids: if the 3 means are initialized in the 3 different clusters, k means will run smoothly; but if 2 or more means are initialized in the same cluster, they algorithm will not perform well, as the centroid can't 'move' to the correct cluster (the correct points are too far, and since they are assigned to a mean via Euclidean Distance, they will necessarily be misclassified).

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?

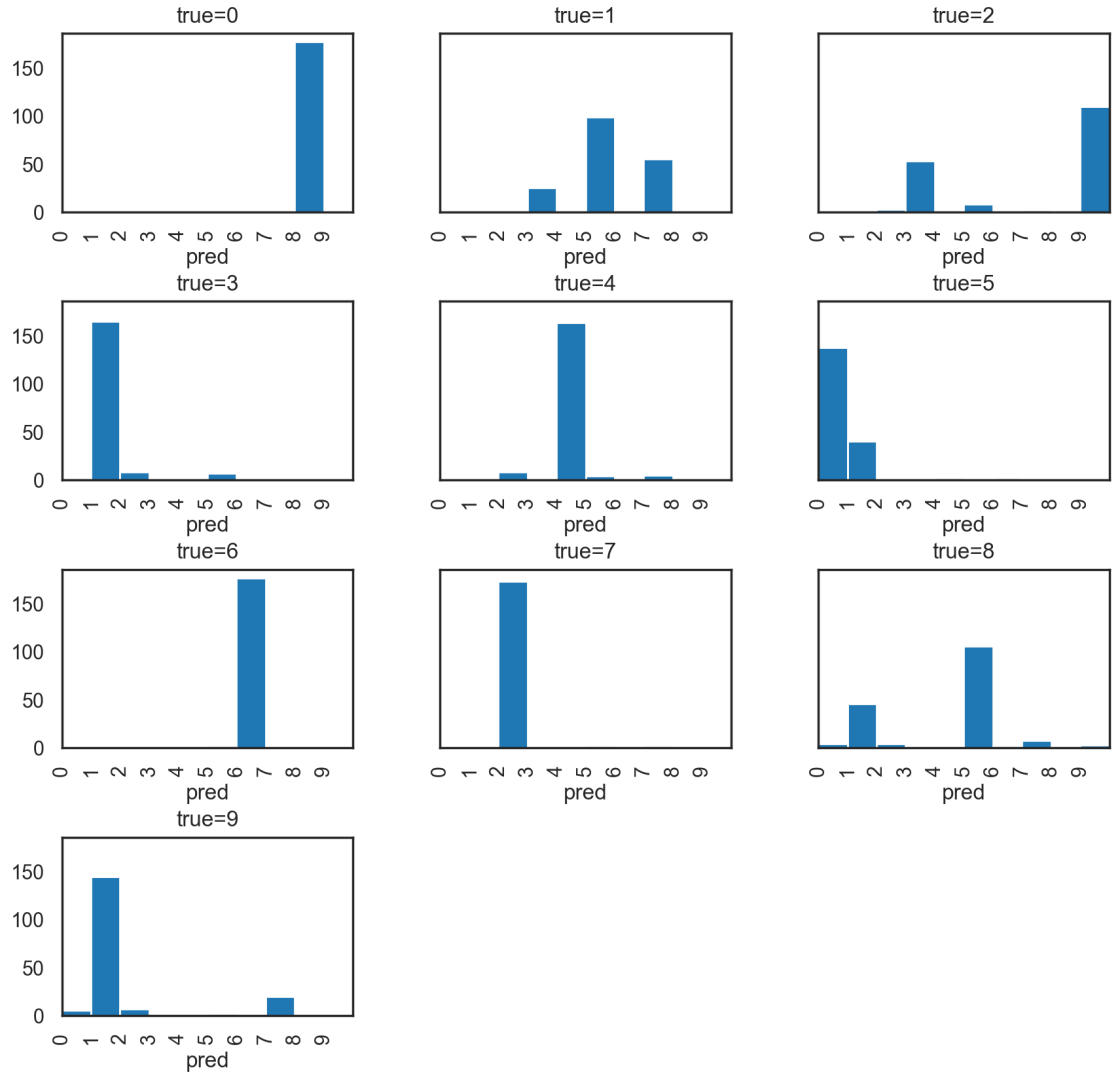
Points: 0.2

The right value for k is 3; this is clear from the plot, which shows an elbow at k=3. This means that there is a sharp improvement in the distortion measure from k = 1, 2 to k = 3, whereas at higher values of k (k=4, k=5 etc.), the distortion measure does not decrease significantly. Hence the optimal choice for k is 3.

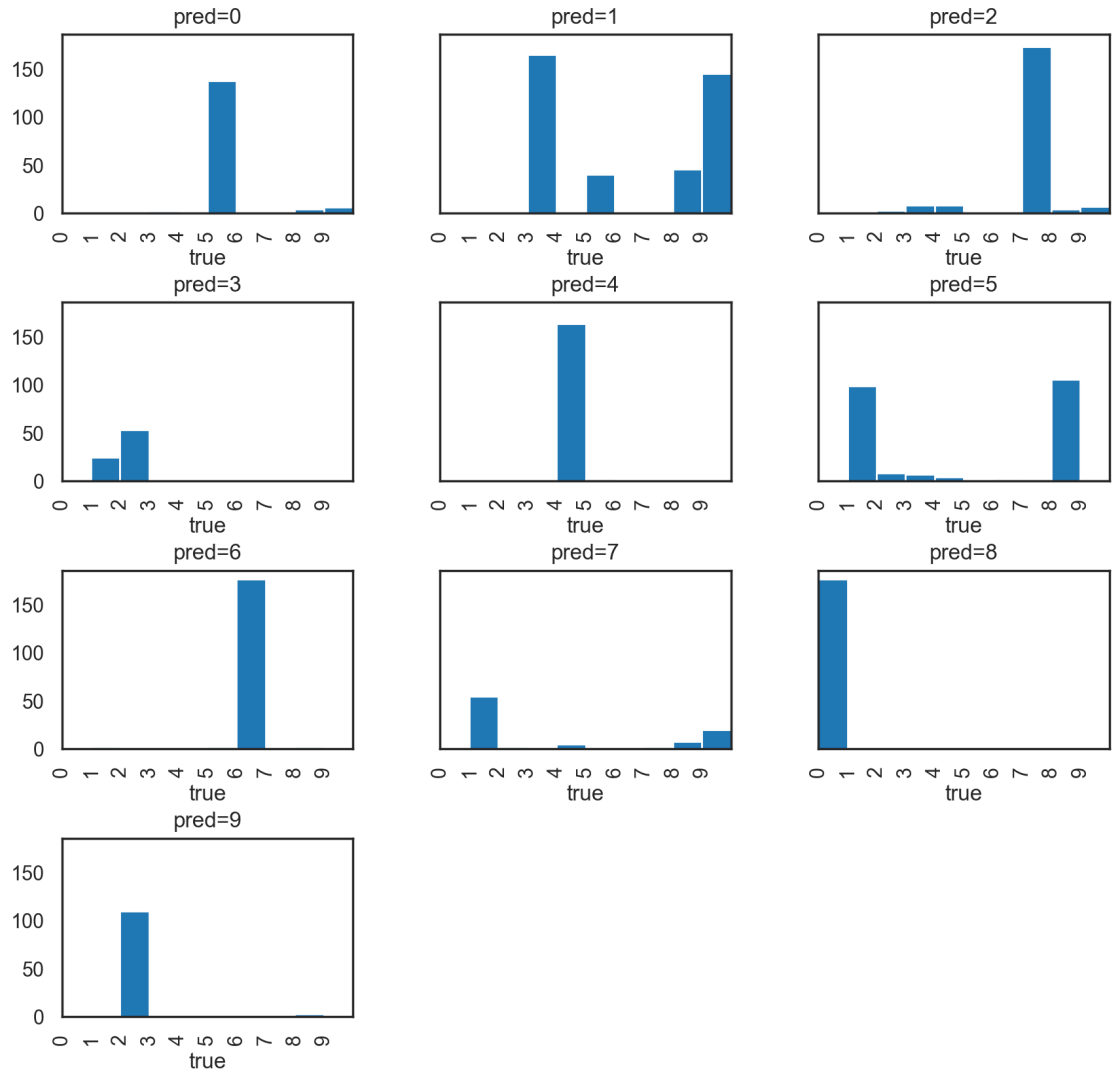
0.5.2 Q3.2 Interpret the result

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 Index is defined as $R = \frac{a+b}{a+b+c+d}$. It is a metric used to compare 2 partitions ($X$and Y) of the same set of data points. The index can be interpreted as the number of agreement between X and $Y(a + b)$, over the the total number of predictions ($a + b + c + d$, where $c + d$ is the number of disagreements between X and Y). Some agreements, however, can be due to chance alone. The adjusted Rand Index is a modified version of the Rand Index which takes into account this possibility. It is defined as $AR = \frac{R-E[R]}{max(R)-E[R]}$, where R is the Random Index and E is the expectation. After comparing the true cluster labels to the KMeans clusters, I obtained R = 0.9204799387248979 and AR = 0.596907083127223. The adjusted score is much lower than the unadjusted Rand Index, suggesting that a percentage of the agreements between the true labels and the KMeans-assigned labels is due to chance alone.

2. the KMeans algorithm is not good at predicting the true label for hand-written digits: the KMeans-generated clusters are not an accurate representation of the true labels. This is evident from the adjusted Rand Index (comparison of true and predicted labels) of 0.597. Meaning, on average, KMeans assigns a point to the correct cluster 59% of the time. The plots are also insightful: for example, 0 gets consistently misclassified by KMeans into 8.

3. KMeans works best with distinct clusters that have similar variance and spherical shape. In the case of hand-written digits, many clusters overlap, as some digits are especially similar to one another (for example 0 and 8, 1 and 7). These are the hardest classifications for KMeans, which the algorithm often gets wrong. Some digits, however, have a characteristic shape which makes them separable form the others. 6 and 4 are two examples of digits that KMeans clusters successfully. Additionally, the very nature of the dataset of hand-written digits is such that different digits could have different variance and cluster shape, which KMeans is not suited to handle.

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.

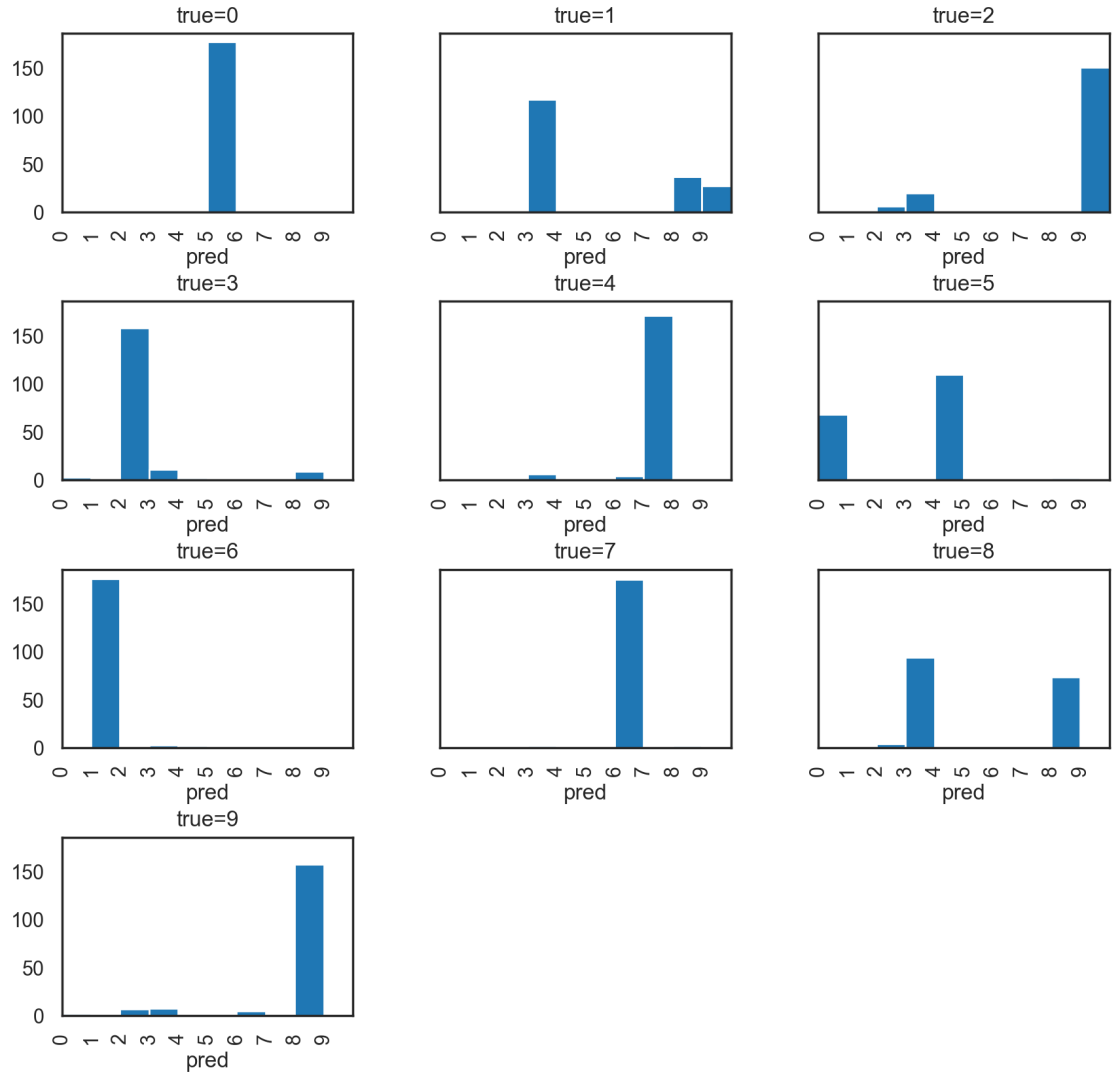
Points: 0.6

KMeans does not fit well to the new dataset because the clusters are not compact, they do not have similar variance, and they are not spherical. These features impede the correct clustering of the data, because they bias the Euclidean Distance metric at the core of this KMeans implementation, which assigns point to the closest centroid. Indeed, in this scenario, 2 points belong to the same cluster even if they are far, but the algorithm can not recognize it. In addition, since KMeans's objective is to minimize the sum of squared distances within the cluster, the algorithm will prefer clusters with smaller variance, and potentially misclassify data points as belonging to the low-variance clusters (as observed above).

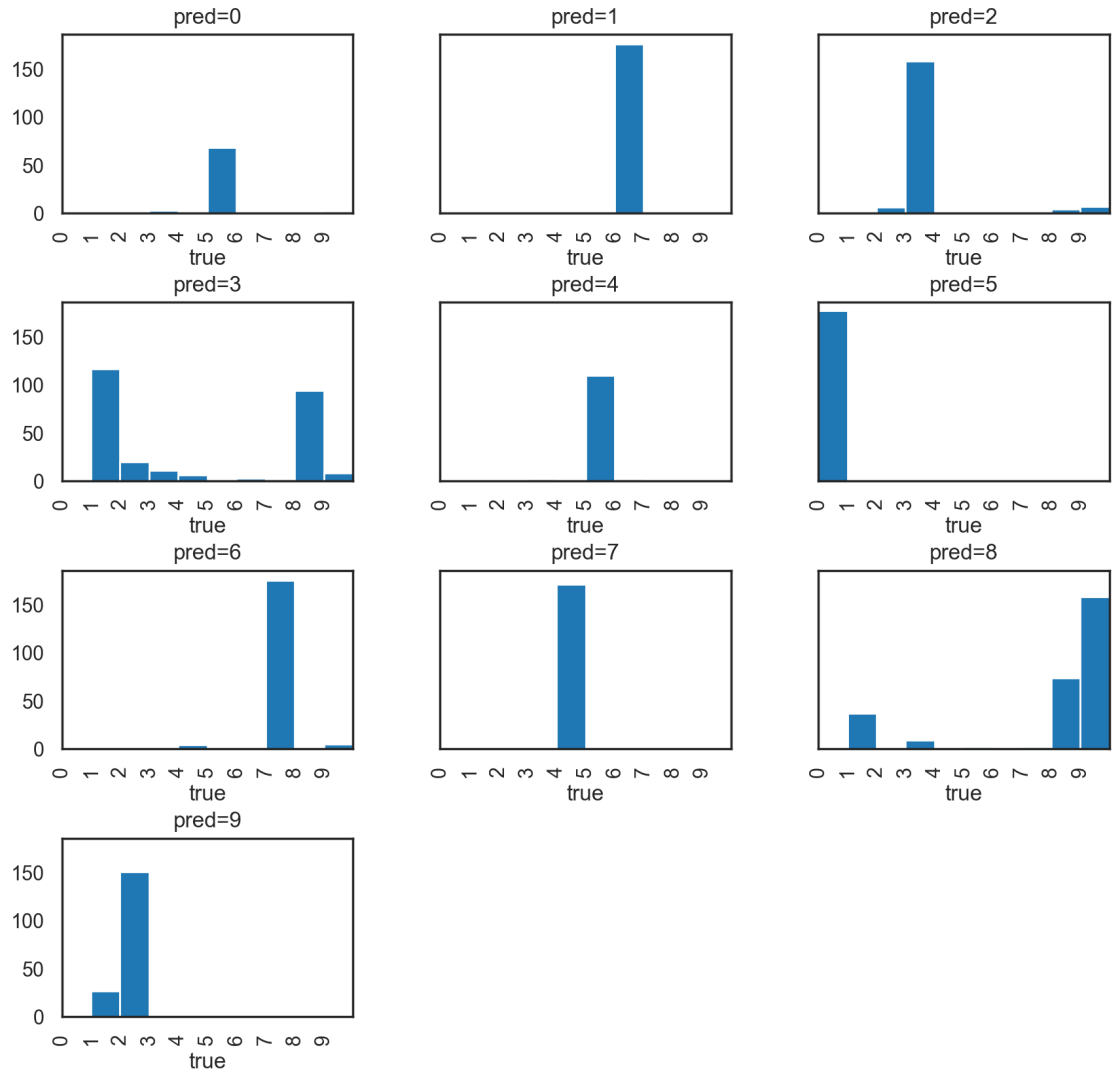
0.7 Q5.2 Interpret the GMM results

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 [ ]: 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 [ ]: 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. The Gaussian Mixture Model is a better predictor of the true scores than KMeans as indicated by the adjusted Rand indexes of, respectively, 0.68 and 0.59. Nonetheless, GMM is not a satisfactory classifier of hand-written digits. For example, the predicted cluster of $2s$ contains mostly pictures of $4s$, and the predicted cluster of $5s$ contains only $0s$. Some clusters, however, are significantly more accurate than the KMeans prediction, as in the case of the digit 7.
2. The `tied` argument passed into the Gaussian Mixture Models constrains all the clusters to share the same covariance matrix, meaning the shape of all the clusters should be similar, with possibly different means and spatial orientations. By contrast, KMeans assumes spherical covariance, so it constrains the clusters to have the same variance in all dimensions. Since GMM performs better than KMeans on the MINST dataset, and given the 'tied' parameter on the GMM model, the true clusters have, probably, different variability in different directions (not spherical covariance).