CS361 PROJECT

Qikai Yang

Content Menu

1. Introduction of designing

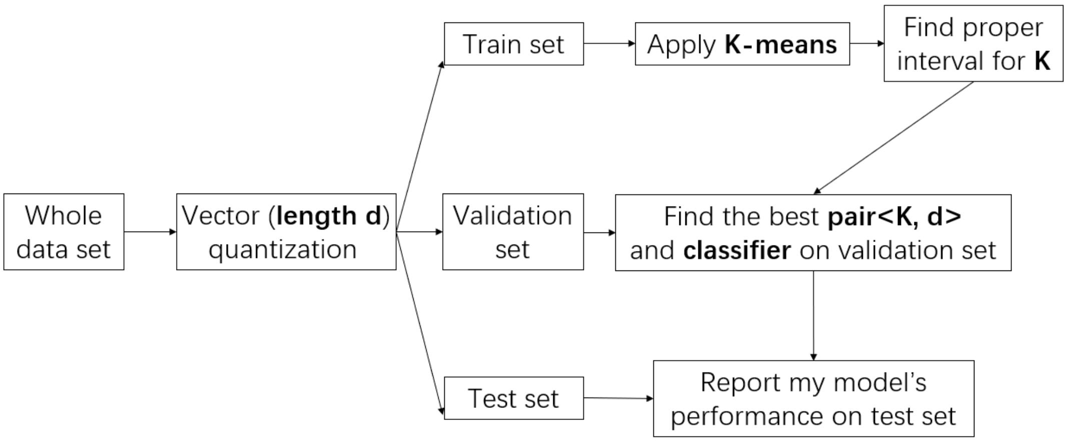
According to the requirements of this project on website. I have applied all the steps and created a framework to show how I design my model as below in the second part.

There are two variables that I would like to emphasize here. In my report, I name the number of cluster centers as **k** and the length of vector quantization as **d**. The length of vector quantization is defined as the number of rows taken as a whole in each time.

Also, I split the data set into 3 parts: train set, validation set and test set. The ratio is 65% : 17.5% : 17.5%.

2. Main Structure

This is the picture of my whole design process.



Whole data set
Vector (length d) quantization
Train set
Apply K-means
Find proper interval for K
Validation set
Find the best pair<K, d> and classifier on validation set
Test set
Report my model's performance on test set

3. How to choose proper **length of vector quantization (d)**

First of all, we have to take a look at the minimum number of rows in each txt file. No matter how **d** changes, it can’t exceed the minimum number of rows in each txt file. If **d** is too large, the txt file with shortest length will be cut off to zero because its length is less than **d** and all its data will be discarded.

After I check all the txt files’ length, the minimum number of rows is **125**. In order to avoid too much data to be discarded during the vector quantization, I set the maximum number of **d** as $\mathrm{int}\left(\frac{125}{2}\right) = 62$.

And I set the **d**’s value changing **10** for each step. Therefore, the initial **d**’s value should be selected from this list **[10,20,30,40,50,60]**

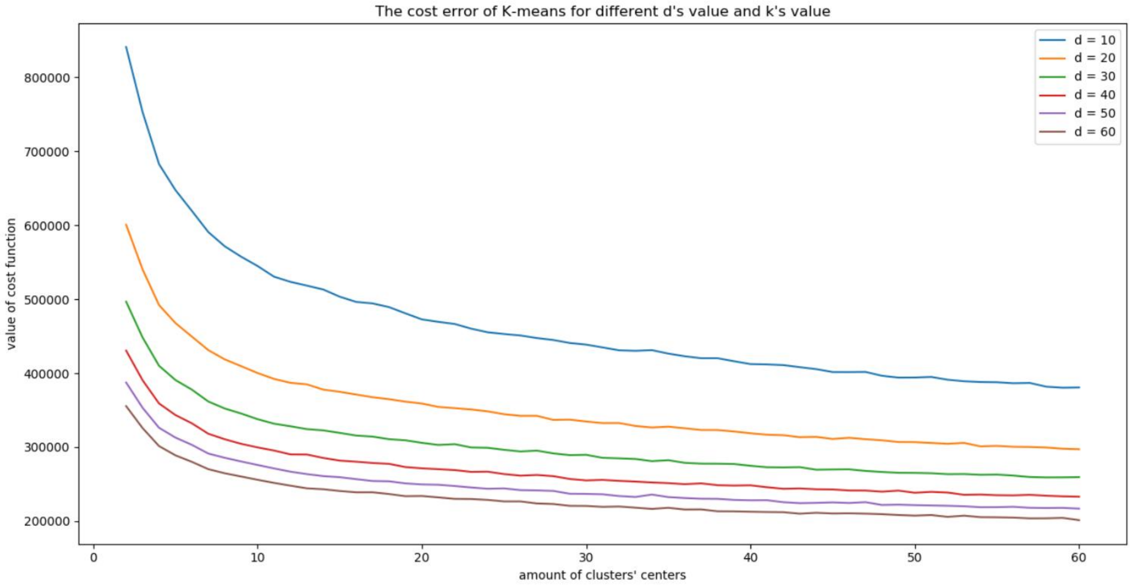
$$\mathrm{d} \in [10,20,30,40,50,60]$$

4. How to choose proper **amount of cluster centers (k)**

To start with, my conclusion is we should set the value of **k** in the interval of **[25,35]**. There are two reasons.

Reason 1

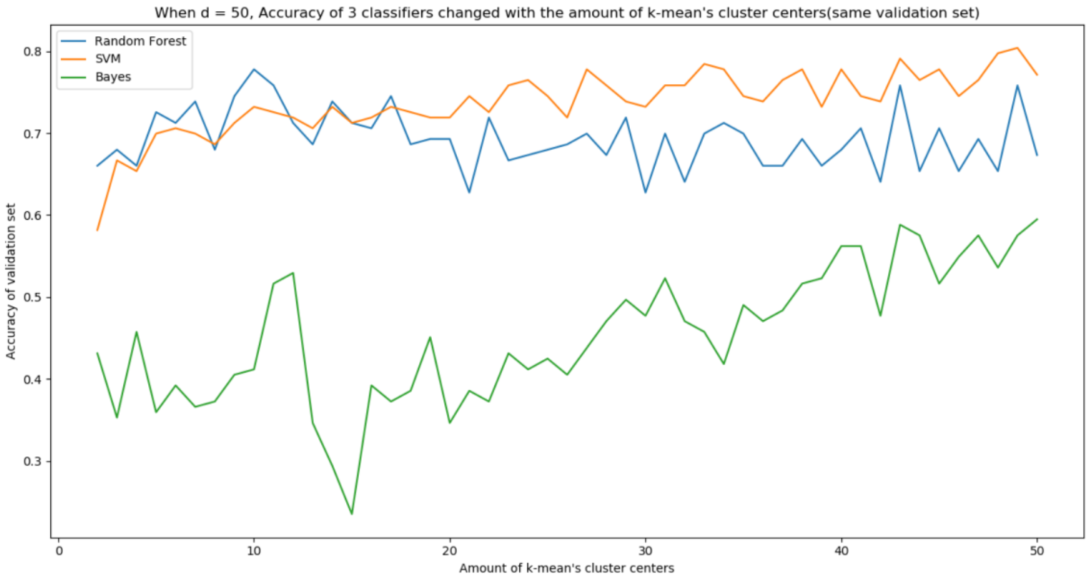
I have calculated and plotted the cost error of K-means on the set of **d** which I have chosen in the third part. And I find that for each **d**, as **k** increases, when **k** is greater than 25, the cost error begins to become stable. Therefore, **k** should be at least 25. (The figure is as below)



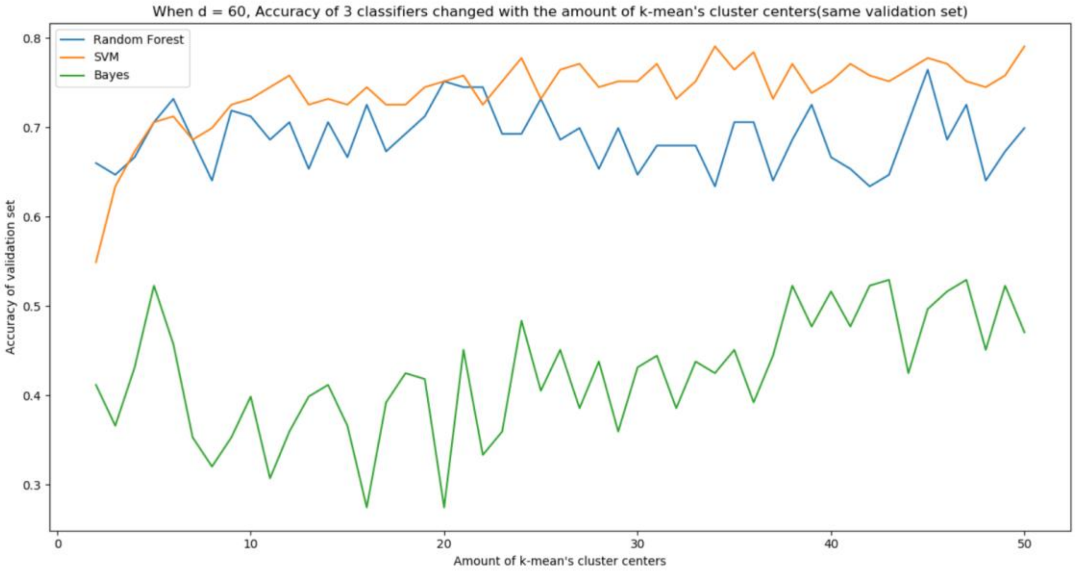
(As the figure shows, when $\mathrm{k} \geq 25$, whichever **d** is, the cost error becomes stable and doesn't change dramatically)

Reason 2

I have made several experiments on the accuracy on the validation set to see if the accuracy will change dramatically as **k** increases. The result shows that when **k** is more than 30, the accuracy doesn't change dramatically as **k** increases. The figure below shows the accuracy of three classifiers when **d** is equal to 50 and 60 (I didn't list **d** = 10, 20, 30 and 40 here, but they are very similar). We can tell this easily from the figures.



(When d = 50)



(When d = 60)

As we all know, the larger **k** is, the longer time will be taken. Therefore, it's unnecessary to search the whole space of **k** and we can just focus on searching smaller area of **k.**

To make everything simple, I set $\mathbf{k} \in [25,35]$.

$$k \in [25,35]$$

Another thing is that, during the process of finding best pair**<k, d>**, I find that the classifier **SVM** performs the best. Therefore, I discard using Random Forrest and Naïve Bayes.

5. Complete process

Based on the interval we set for **k** and **d** as well as the **best classifier**, we begin our process.

$$d \in [10,20,30,40,50,60]$$

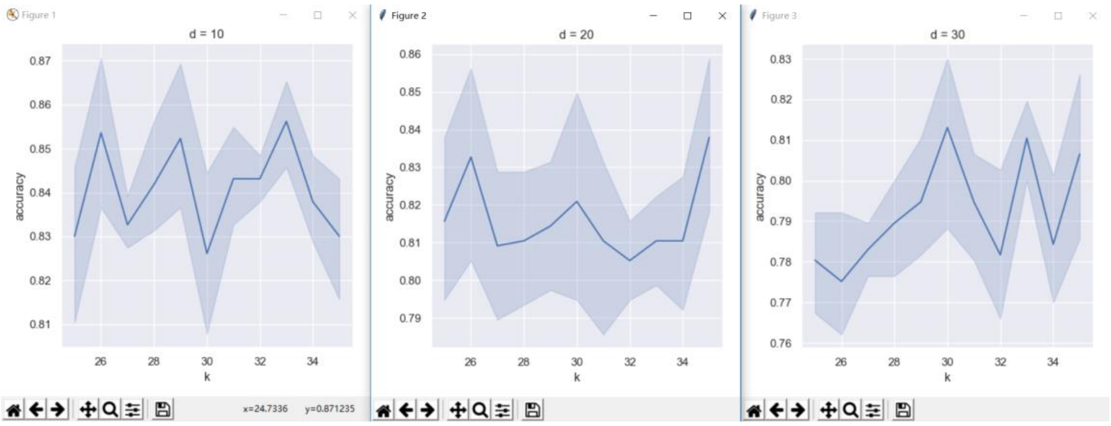
$$k \in [25,35]$$

The best classifier: SVM

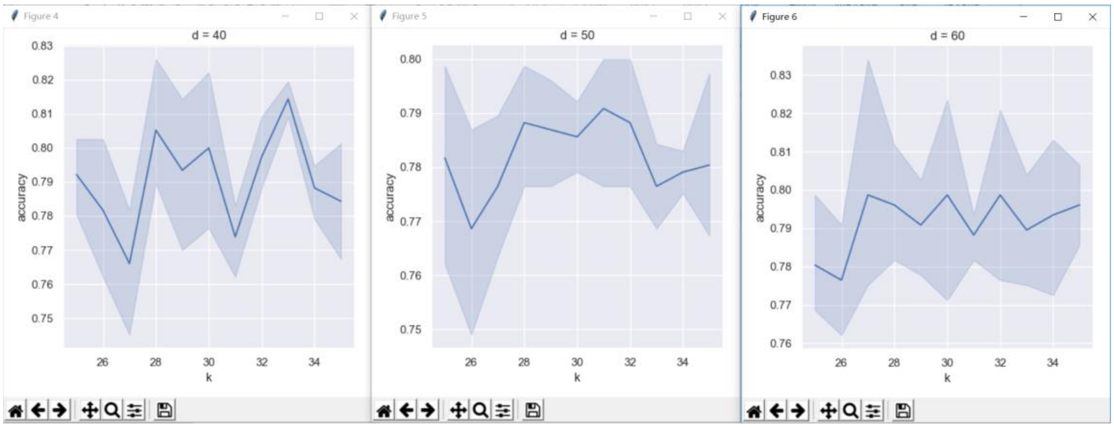
When I trained the K-means model on the train set and compare the accuracy of each pair**<k**, **d>**, I plotted them out to see which pair**<k**, **d>** performs the best. The figures are as follow:

One thing important:

Because the K-means method has some uncertain random, I run every pair**<k, d>** for 5 times and compare their mean value to find the best pair.



(When d = 10, 20 and 30)



(When d = 40, 50 and 60)

As we can see the best pair**<k, d>** is <33, 10>, whose average accuracy is 0.856209.

6. Final evaluation

After completing all the tasks above, I use the model with the best parameters to test its accuracy on my test set. The final model is:

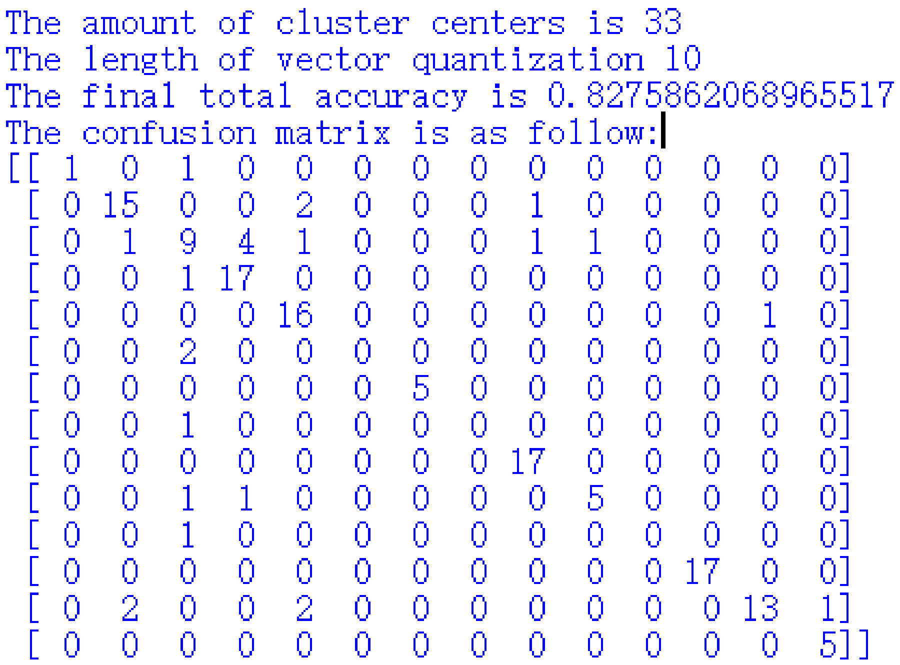
$$< \mathrm{k}, \mathrm{d} > = < 33{,}10 >$$

Model: SVM

The final total accuracy is 0.8275862068965517

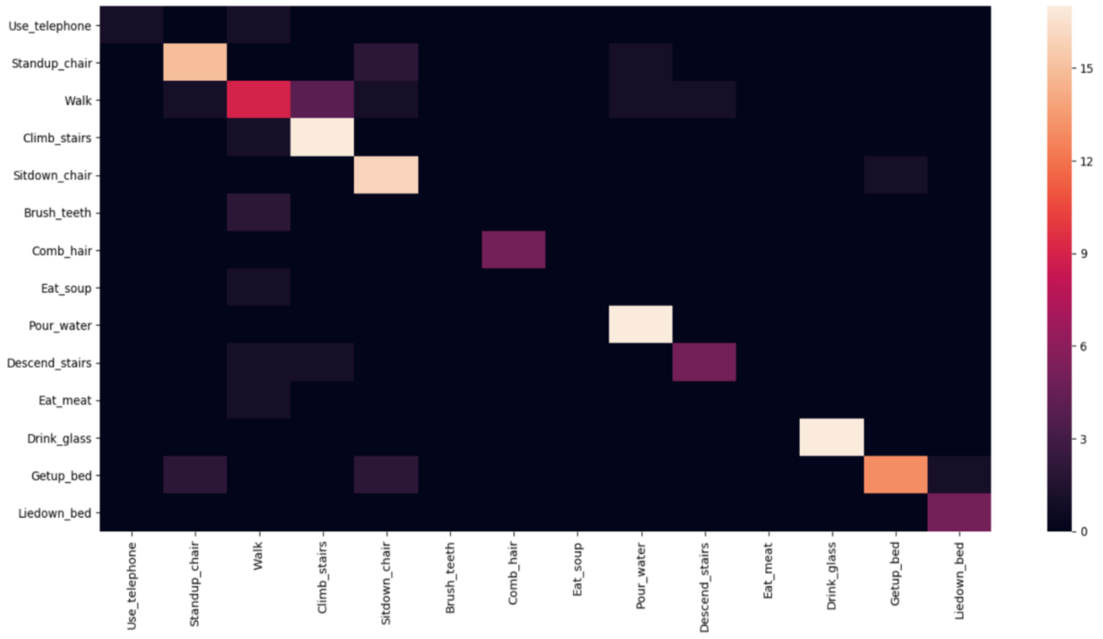
The final total error rate is 0.1724137931

The final confusion matrix is:



```
The amount of cluster centers is 33
The length of vector quantization 10
The final total accuracy is 0.8275862068965517
The confusion matrix is as follow:
[[ 1  0  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 15  0  0  2  0  0  0  1  0  0  0  0  0]
 [ 0  1  9  4  1  0  0  0  1  1  0  0  0  0]
 [ 0  0  1 17  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 16  0  0  0  0  0  0  0  1  0]
 [ 0  0  2  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  5  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 17  0  0  0  0  0]
 [ 0  0  1  1  0  0  0  0  0  5  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 17  0  0]
 [ 0  2  0  0  2  0  0  0  0  0  0  0 13  1]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  5]]
```

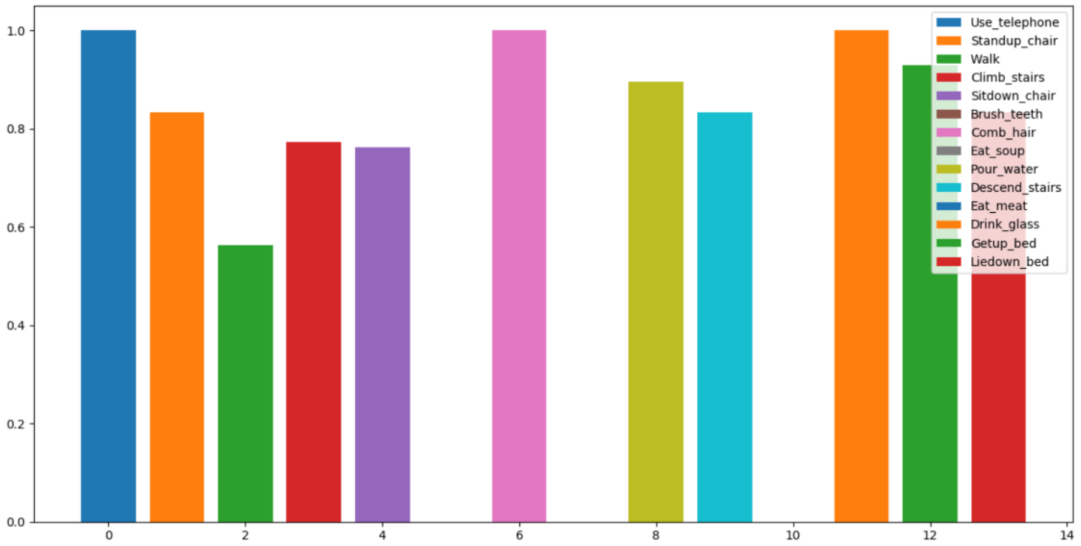
If we construct a heatmap for this matrix, it appears like this:



The specific accuracy for each label is as follow: (If it is nan, it means there is no item with such label in the test set)

Predicted as	***Each label’s prediction accuracy***
Use_telephone	100%
Standup_chair	83.33%
Walk	56.25%
Climb_stairs	77.27%
Sitdown_chair	76.19%
Brush_teeth	Nan
Comb_hair	100%
Eat_soup	Nan
Pour_water	89.47%
Descend_stairs	83.33%
Eat_meat	Nan
Drink_glass	100%
Getup_bed	92.85%
Liedown_bed	83.33%

If we use a bar-plot to see how our model works on each label, the result looks like this:



7. Further improvement

In my model, I think there are some parts that can be improved.

First of all, Although I have experimented several **d**’s value, it is apparent that such amount of experiments is not enough. The list [10,20,30,40,50,60] can only be used to roughly find the best **d**. Limited to the running time, I can’t make the interval smaller. If any extra information can be provided, **d**’s value can be more accurate.

Secondly, even though I tried to reduce the running time. My codes still run for a quite long time. If the search space for **k** and **d** can be dramatically reduced, the running time can decrease so that the whole method can be more efficient.