# Homework 1

Chujie Chen
cchen641@gatech.edu

September 26, 2019

## 1 Probability

### (a)

Total number of woman staff = $50 * 50\% + 75 * 60\% + 100 * 70\% = 140$
Number of woman staff worked at store C = $100 * 70\% = 70$

$$\boxed{P = \frac{70}{140} = \frac{1}{2}}$$

### (b)

From Bayes's law, we have

$$\begin{aligned} P(hasDisease|+) &= \frac{P(hasDisease, +)}{P(+)} \\ &= \frac{P(+|hasDisease)P(hasDisease)}{P(+|hasDisease)P(hasDisease) + P(+|healthy)P(healthy)} \\ &= \frac{95\% \times 0.5\%}{95\% \times 0.5\% + 1\% \times (1 - 0.5\%)} \\ &\boxed{\approx 32.31\%} \end{aligned}$$

### (c)

Atlanta Braves (AB) can end up with 87, 88, 89 or 90 wins. San Francisco Giants (SFG) / Los Angeles Dodgers (LAD) can end up with 89 / 86, 88 / 87, 87 / 88 or 86 / 89 wins, respectively.
For AB, probabilities of different wins are

$$\begin{aligned} P(AB = 90) &= \frac{1}{2^3} = \frac{1}{8} \\ P(AB = 89) &= \binom{3}{2}\frac{1}{2^2}\frac{1}{2} = \frac{3}{8} \\ P(AB = 88) &= \binom{3}{1}\frac{1}{2^1}\frac{1}{2^2} = \frac{3}{8} \\ P(AB = 87) &= \frac{1}{2^3} = \frac{1}{8} \end{aligned}$$

For SFG and LAD, similarly, we have

$$P(SFG=89, LAD=86) = \frac{1}{2^3} = \frac{1}{8}$$
$$P(SFG=88, LAD=87) = \binom{3}{2}\frac{1}{2^2}\frac{1}{2} = \frac{3}{8}$$
$$P(SFG=87, LAD=88) = \binom{3}{1}\frac{1}{2^1}\frac{1}{2^2} = \frac{3}{8}$$
$$P(SFG=86, LAD=89) = \frac{1}{2^3} = \frac{1}{8}$$

Since we don't care whether it's SFG beat LAD or LAD beat SFG. We can rewrite above probabilities into two cases:

$$P(one\ 89, one\ 86) = P(SFG=89, LAD=86) + P(SFG=86, LAD=89) = \frac{1}{4}$$
$$P(one\ 88, one\ 87) = P(SFG=88, LAD=87) + P(SFG=87, LAD=88) = \frac{3}{4}$$

Thus, AB can win with 90, 89, 88 wins. The final probability is

$$\begin{aligned} P(AB\ win) =& P(AB=90) + P(AB=89) \times (P(one\ 89, one\ 86) \times P(ABwinInPlayoff) \\ &+ P(one\ 88, one\ 87)) + P(AB=88) \times P(one\ 88, one\ 87) \times P(ABwinInPlayoff) \\ =& \boxed{\frac{19}{32}} \end{aligned}$$

For $P(ABwinInPlayoff) = \frac{1}{2}$.

### (d)

Based on what we did above,

$$\begin{aligned} P(hasPlayoff) &= P(AB=89) \times P(one\ 89, one\ 86) + P(AB=88) \times P(one\ 88, one\ 87) \\ &\boxed{= \frac{3}{8}} \end{aligned}$$

## 2 Maximum Likelihood

### (a)

Since it's i.i.d., we have the joint probability:

$$P(x_1, x_2, ..., x_n|\lambda) = \prod_{i=0}^{n} P(x_i = k_i)$$

More specifically, we have log likelihood function:

$$\begin{aligned} \mathcal{L} &= logP(x_1, x_2, ..., x_n|\lambda) \\ &= \sum_i logP(x_i = k_i) \\ &= \sum_i [log\lambda^{k_i} + log(e^{-\lambda}) - log(k_i!)] \\ &= \sum_i [k_i log\lambda - \lambda - log(k_i!)] \end{aligned}$$

After taking the first derivative:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{\sum_i k_i}{\lambda} - \sum_i 1 = \frac{\sum_i k_i}{\lambda} - n \hat{=} 0$$

We can get the maximum likelihood estimator of $\lambda$ is

$$\boxed{\hat{\lambda} = \frac{1}{n}\sum_i k_i = \bar{X}}$$

## (b)

We can write log likelihood function as below

$$\begin{aligned} \mathcal{L} &= log f(x_1, x_2, ..., x_k; n, \theta_1, \theta_2, ..., \theta_k) \\ &= log(n!) - \sum_j log(x_j!) + \sum_j x_j log(\theta_j) \end{aligned}$$

In order to get the maximum with constraints, we use Lagrange multiplier and get

$$\mathscr{L} = \mathcal{L} + \lambda(\sum_j \theta_j - 1)$$

Then take the derivative, we have

$$\frac{\partial \mathscr{L}}{\partial \theta_j} = \frac{x_j}{\theta_j} + \lambda \hat{=} 0$$

Thus, $\hat{\theta_j} = -\frac{x_j}{\lambda}$ where $\lambda = -n$ since

$$\sum_j x_j = -\lambda \sum_j \theta_j = -\lambda \hat{=} n$$

As a result, the estimator

$$\boxed{\hat{\theta}_j = \frac{x_j}{n}}$$

## (c)

Since data samples are i.i.d., we can get the joint pdf by multiplying them together.

$$f(x_1, x_2, ..., x_n; \mu, \sigma^2) = \frac{1}{\sigma^n (2\pi)^{\frac{n}{2}}} exp\{-\frac{1}{2\sigma^2}[(x_1 - \mu)^2 + (x_2 - \mu)^2 + ... + (x_n - \mu)^2]\}$$

Then we can have log likelihood function:

$$\mathcal{L}(\mu, \sigma^2 | x_1, ..., x_n) = -nlog\sigma - \frac{n}{2}log(2\pi) - \frac{1}{2\sigma^2}[(x_1 - \mu)^2 + ... + (x_n - \mu)^2]$$

The dirivatives are:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mu} &= \frac{1}{\sigma^2}[(x_1 - \mu) + ... + (x_n - \mu)] \\ &= \frac{1}{\sigma^2}(\sum_i x_i - n\mu) \hat{=} 0 \\ \frac{\partial \mathcal{L}}{\partial \sigma} &= -\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_i [(x_i - \mu)^2] \hat{=} 0 \end{aligned}$$

We can have estimators then

$$\begin{aligned} \hat{\mu} &= \frac{\sum_i x_i}{n} = \boxed{\bar{X}} \\ \hat{\sigma}^2 &= \boxed{\frac{\sum_i (x_i - \hat{\mu})^2}{n}} \end{aligned}$$

## 3 Principal Component Analysis

### (a)

Since we have

$$\begin{aligned} J &= \frac{1}{N}\sum_n ||x^n - \tilde{x}^n||^2 \\ &= \frac{1}{N}\sum_n (x^n - \tilde{x}^n)^T(x^n - \tilde{x}^n) \end{aligned}$$

By taking the derivative over $z_j^n$ and set the result to be 0. We can have

$$\frac{\partial J}{\partial z_j^n} = -\frac{1}{N}\sum_n [u_j^T(x^n - \tilde{x}^n) + (x^n - \tilde{x}^n)^T u_j] \hat{=} 0$$

Which means we want the projected sum of lengths (can be negative) of $(x_i^n - \tilde{x}_i^n)$ to be zero. Apparently, we require that (notice that $u_1, ..., u_D$ are orthonormal bases)

$$\begin{aligned} 0 &= \sum_n (x^n - \tilde{x}^n)^T u_j \\ &= \sum_n [\sum_i^D ({x^n}^T u_i)u_i - \sum_{i=1}^M z_i^n u_i + \sum_{i=M+1}^D b_i u_i] u_j \\ &= \sum_n ({x^n}^T u_j - z_j^n) \end{aligned}$$

Thus, for $j = 1, ..., M$. We need $\boxed{z_j^n = {x^n}^T u_j}$.

### (b)

Almost the same but with small differences, we have

$$\frac{\partial J}{\partial b_j} = -\frac{1}{N}\sum_n [u_j^T(x^n - \tilde{x}^n) + (x^n - \tilde{x}^n)^T u_j] \hat{=} 0$$

The condition becomes,

$$\sum_n ({x^n}^T u_j - b_j) = 0$$

Thus,

$$\boxed{b_j = \frac{1}{N}\sum_{n=1}^N {x^n}^T u_j = \bar{x}^T u_j}$$

where $j = M + 1, ..., D$.

### (c)

$$\tilde{x}^n = \boxed{\sum_{i=1}^M ({x^n}^T u_i)u_i + \sum_{i=M+1}^D (\bar{x}^T u_i)u_i}$$

$$x^n - \tilde{x}^n = \boxed{\sum_{i=M+1}^D (x^n - \bar{x})^T u_i u_i}$$

### (d)

We can rewrite $J$

$$\begin{aligned}
J &= \frac{1}{N}\sum_n ||x^n - \tilde{x}^n||^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D} u_i^T(x^n-\bar{x})(x^n-\bar{x})^T u_i \\
&= \sum_{i=M+1}^{D} u_i^T[\frac{1}{N}\sum_{n=1}^{N}(x^n-\bar{x})(x^n-\bar{x})^T]u_i \\
&= \sum_{i=M+1}^{D} u_i^T S u_i
\end{aligned}$$

So the optimization problem becomes

$$\min_{u_i} J = \min_{u_i} \sum_{i=M+1}^{D} u_i^T S u_i$$

Eigenvectors:

$$Su = \lambda u$$

For $i = M+1, ...., D$, this is to pick eigenvectors $u_i$ with **smallest** $(D-M)$ eigenvalues. While for $i = 1, ..., M$, $u_i$ are the rest $(M)$ orthogonal bases.

## 4 Clustering

### (a)

When using Euclidean distance

$$||x^n - \mu^k||^2 = (x^n - \mu^k)^T(x^n - \mu^k)$$

We let the derivative to be zero

$$\frac{\partial J}{\partial \mu^k} = 0$$

And get

$$\sum_{n=1}^{N} r^{nk}(x^n - \mu^k) = 0$$

This will give us

$$\boxed{\mu^k = \frac{\sum_n r^{nk} x_n}{\sum_n r^{nk}}}$$

### (b)

**Finiteness:**
$N$ data points have finite assignments into $K$ clusters:

$$number\ of\ ways\ of\ partitions\ \leq K^N$$

**Strictly decreasing:**
1. For the cluster assignement

$$\pi(i) = argmin_{j=1,...,k}||x^i - c^j||^2$$

we are making sure the new Euclidean distance between the data point and the belonging cluster is no greater than the old distance, which is

$$||x^i - c^{\pi(i)}_{current}||^2 \leq ||x^i - c^{\pi(i)}_{previous}||^2$$

making the

$$J_{current} \leq J_{previous}$$

2. For the center adjustment

$$c^j = \frac{1}{|\{i : \pi(i) = j\}|} \sum_{i:\pi(i)=j} x^i$$

which is also

$$c^j = argmin_c \sum_{i:\pi(i)=j} ||x^i - c||^2$$

for Euclidean distances. This also makes the objective functino decrease

$$J_{current} \leq J_{previous}$$

Along with what we have at the beginning, the volume is finite and discrete. **Thus, the K-means algorithm converges.** Moreover, we check if the cluster centers change or not to continue or stop the iteration gives us finite iterations.

### (c)

It is **Average linkage**. By comparing the average distance between elements of each cluster

$$D(g_i, g_j) = min_{x \in g_i, y \in g_j} \frac{1}{mp} \sum_i \sum_j ||x_i - y_j||$$

that is similar to the objective of K-means.

### (d)

**Single linkage** would successfully separate the two moons. Because each data point in one moon is close to nearby data points in the same but not the other moon. We must carefully link data points that are close enough to each other.

## 5 Programming: Image compression

### 1

**Representatives of each cluster:** I am using 'centroid' which is a K $\times$ $size(pixels, 2)$ matrix. Each row represents a centroid, and each column in that row is a color number. At the very beginning, I randomly chose K data points as centroids. And during the assignment, I modify the 'class' (a N * 1 vector) to record assignments.

**What distance measures I tried:** I tried Euclidean / $L1$ / 'cosine' / 'inf' -distance measures. And I choose $L1$ distance (Manhattan distance). I stop the iteration when all centroids don't move.

**More details:** Before the iteration, as elaborated above, I randomly select K data points as initial centroids. For the iteration, as long as any centroid moves from last loop, do following: (1) get the cluster assignment done by connecting data points with their nearest cluster. The distance is Manhattan distance. (2) adjust the center by choosing the middle elements of colors among data points in its cluster. More details can be found here: https://www.geeksforgeeks.org/find-a-point-such-that-sum-of-the-manhattan-distances-is-minimized/

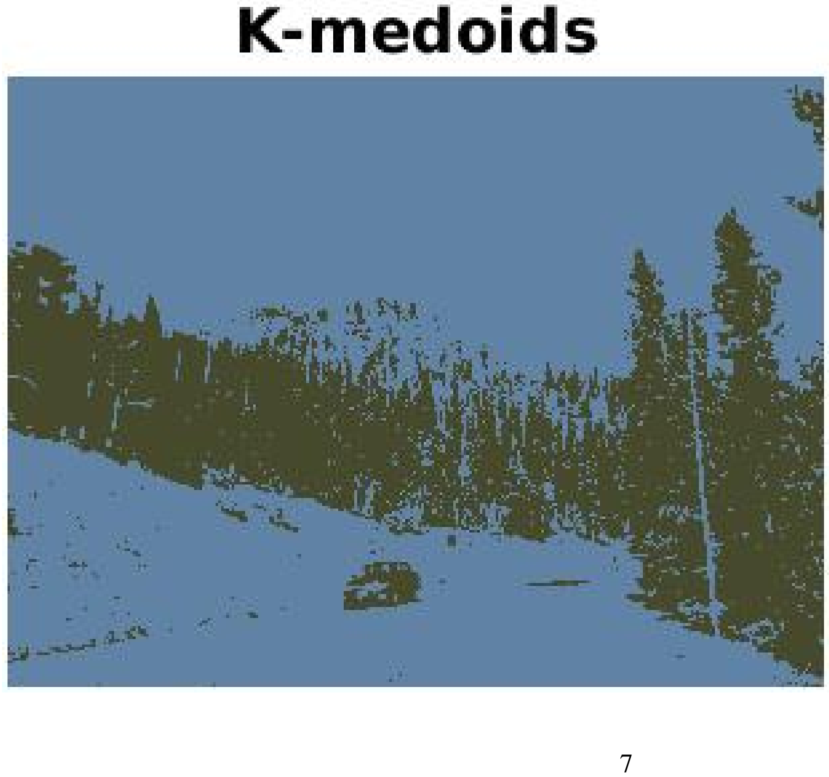
## 2 Original picture:

original.jpg (310 × 260)

## 3

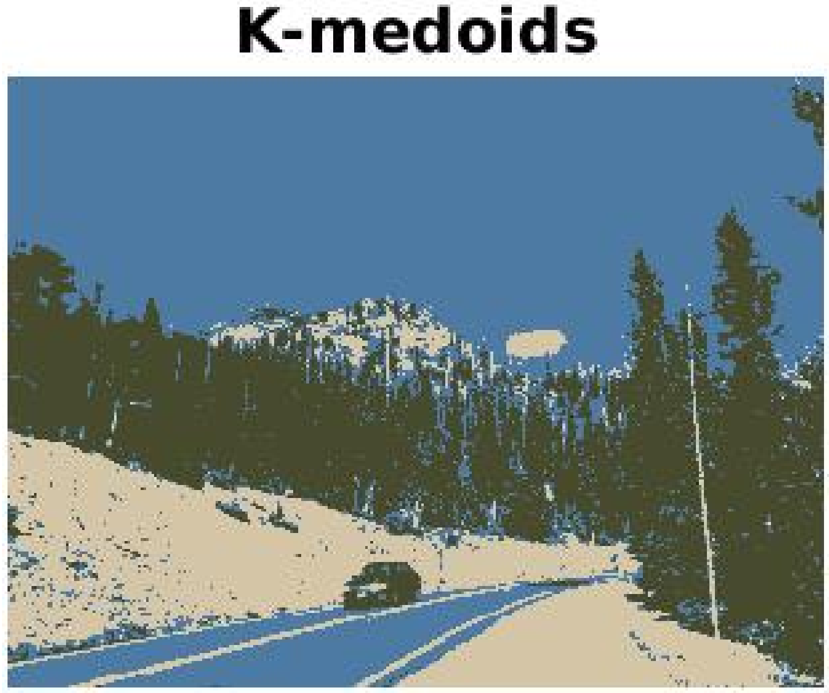
I ran K with 2, 3, 16 and 32. Every K I ran three times in order to get average runtime. Results (from one run for every K) from K-medoids are below:

========================= K = 2 ==============================================



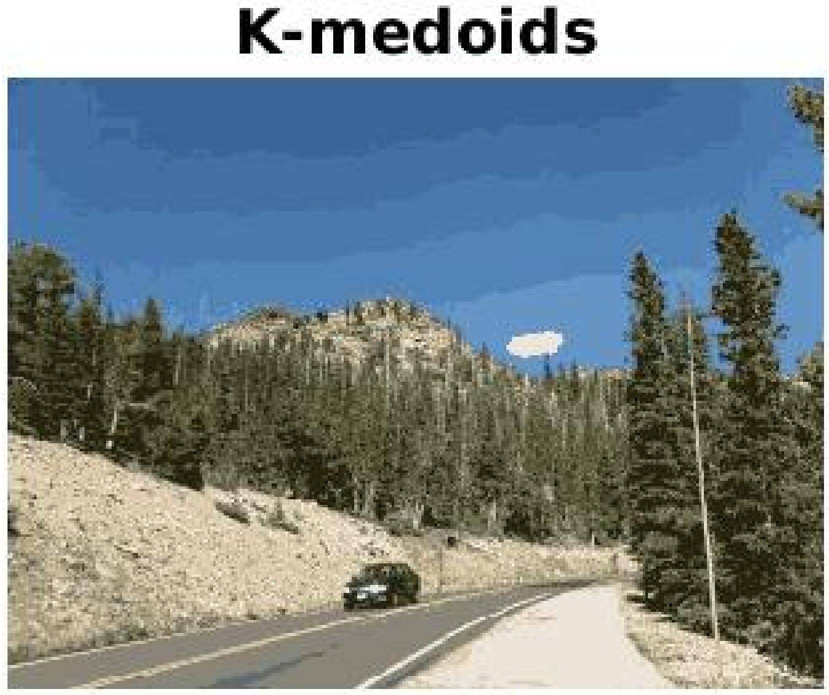
k_medoids_2.jpg

```
================================================================================
========================= K = 3 ================================================
```



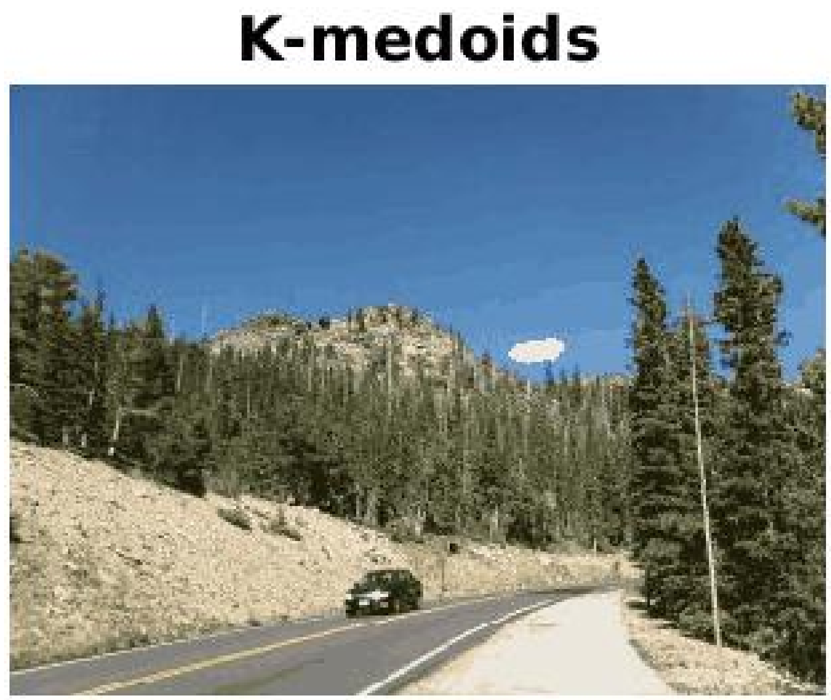
k_medoids_3.jpg

```
================================================================================
========================= K = 16 ================================================
```



k_medoids_16.jpg

```
================================================================================
========================= K = 32 ================================================
```



k_medoids_32.jpg
============================================================================

**Observationn**

Clearly, as K becomes larger, more colors are used, and thus, the result becomes more like the original picture. For small Ks, a slight change would make obvious difference while for larger Ks, the quality of picture would improve but not much for higher Ks.

**Time**

I calculated the average time it take to converge:

$$K = 2, \qquad time = 0.3611s$$
$$K = 3, \qquad time = 1.3914s$$
$$K = 16, \qquad time = 6.6026s$$
$$K = 32, \qquad time = 14.6547s$$

Higher Ks normly come with a longer time for a convergence. But for each certain K, the fluctuation is there because the process highly depends on the initial assignment.

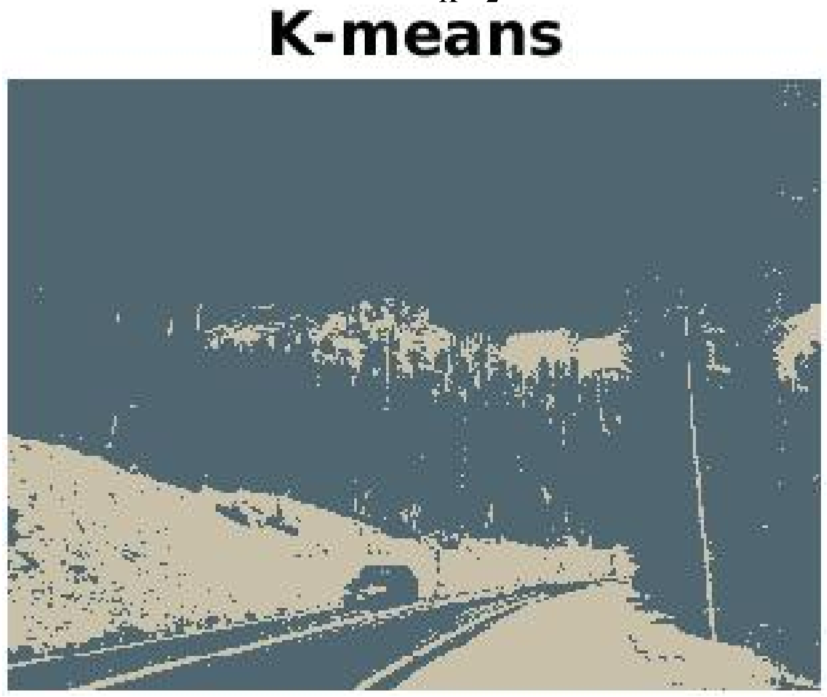
## 4

Using different initial centroids **does** affect final results. Especially for small Ks. Sometimes I will see same results and sometimes I will see different results.

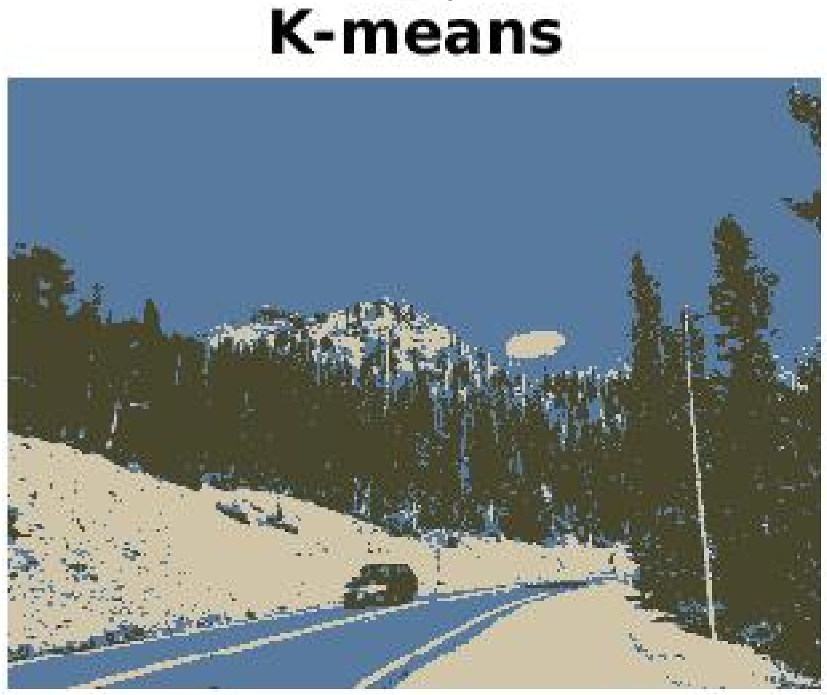
## 5

Results (from one run for every K) from K-means are below:

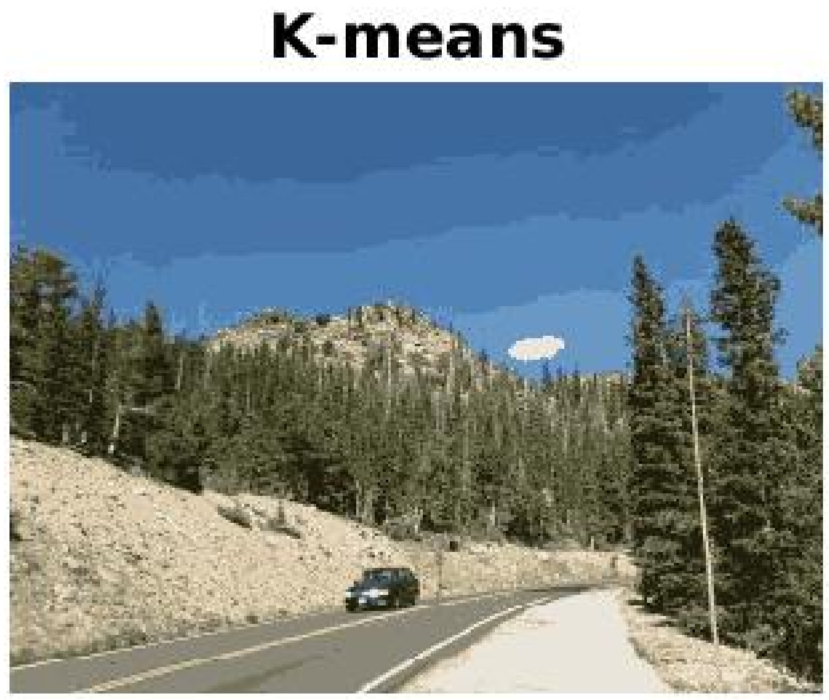
========================= K = 2 =============================================

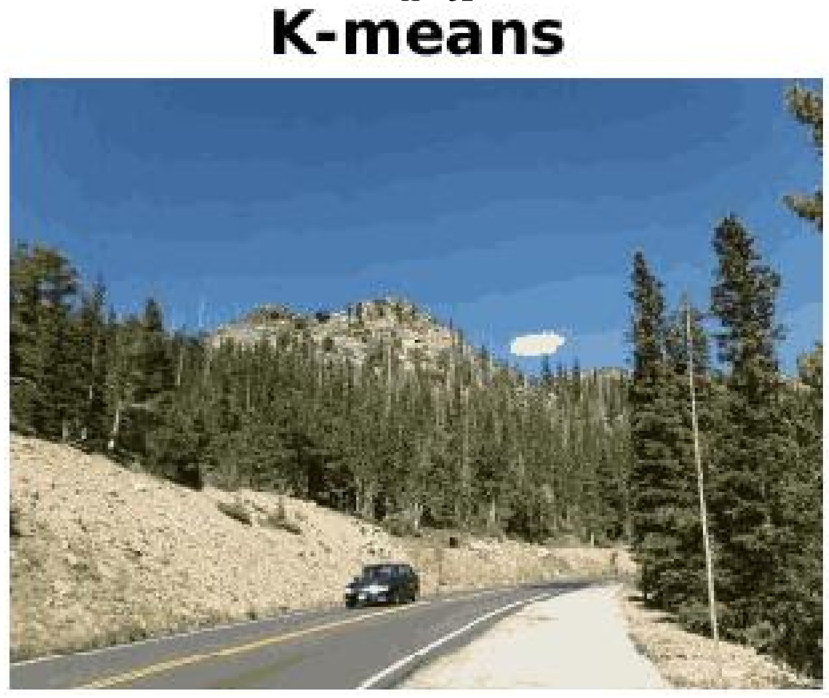


k_means_2.jpg
===========================================================================
========================= K = 3 =============================================



k_means_3.jpg
===========================================================================
========================= K = 16 ============================================



k_means_16.jpg
==============================================================================
========================= K = 32 ==============================================


k_means_32.jpg
==============================================================================

**Observationn**

Similarly, as K becomes larger, more colors are used, and thus, the result becomes more like the original picture. For small Ks, a slight change would make obvious difference while for larger Ks, the quality of picture would improve but not much for higher Ks.

## Time

I calculated the average time it take to converge:

$$\begin{aligned} K &= 2, \qquad time = 0.3330s \\ K &= 3, \qquad time = 0.4552s \\ K &= 16, \qquad time = 15.5103s \\ K &= 32, \qquad time = 38.4644s \end{aligned}$$

Higher Ks normly come with a longer time for a convergence. But for each certain K, the fluctuation is there because the process highly depends on the initial assignment.

## K-means vs. K-medoids

**output quality:** They produce pretty much the same figure quality. At least I cannot tell the difference in quality by eyes.
**robustness:** K-medoids is more robust since in my implementation the centers are always some data points. It is more robust to outliers than K-means.
**running time:** For my implementations, K-medoids is faster than K-means.