

CS6375 - Project 4

Khoi Nguyen

Part 1: K-means for image compression

Background

I implemented K-means to compress two sample images. Five values of K were tried, namely 2, 5, 10, 15, and 20. For each K, K-means was repeated five times with different initialization.

The compression ratio is defined as $\frac{size_{after}}{size_{before}}$. The smaller the ratio, the more space saved. A detail not described in the assignment is *how* to measure $size_{after}$. I took the trivial approach of saving the new image as a JPEG file and read the file size on the file system.

Results

The compression ratios for each K value are presented in Figure 1. Generally, when K increases, the compression ratio increases. However, for the Koala image, from K=5, the compression ratio surprisingly decreases! I believe this is the artifact of the JPEG compression algorithm that I don't have control over.

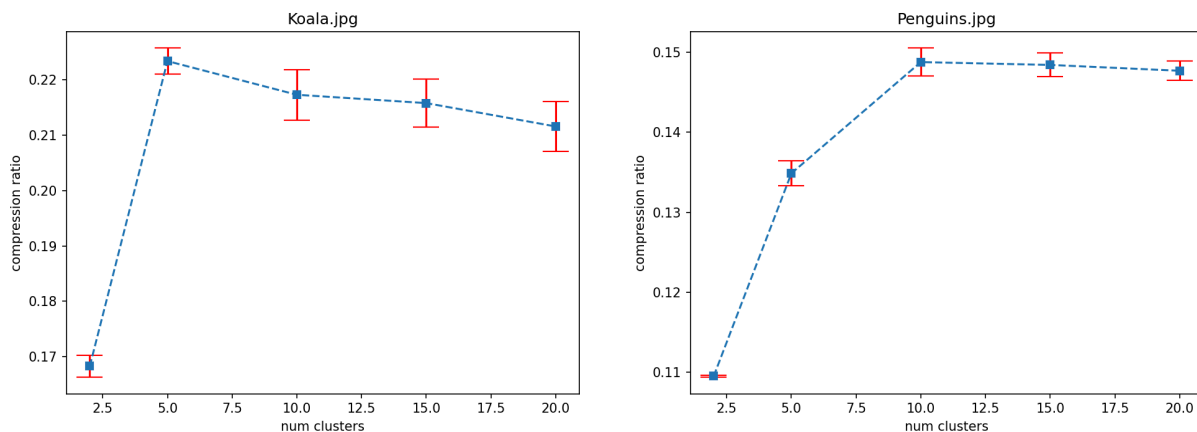




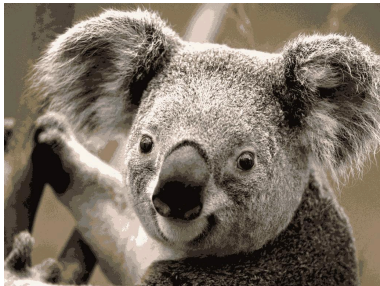





Figure 1. Compression ratios of K-means on two sample images

In Table 1 below, I provide some samples of the compressed images. We can see that, at K=2, we have nothing more than an “accented” black-and-white photo. This can already be useful for some applications. Going to K=5 and 20, the images become more lively (apparently, with 3 and 18 more colors), more details started to be captured. What is left until the original image is more colors to capture more fine-grained details.

As mentioned above, the choice of K really depends on the requirements of one's specific applications. But if I have to choose, I would pick $K=5$, because the images are much more lively than $K=2$, and not too far from $k=20$ or even the original image. However, the compression ratio for $k=5$ is still pretty low, which creates a good quality-efficiency balance.

	Koala	Penguins
k=2		
k=5		
k=20		
original		

Part 2: Bayes net learning

A single Chou-Liu Tree

I used the provided implementation by the instructor.

EM with Mixture of Chou-Liu Trees

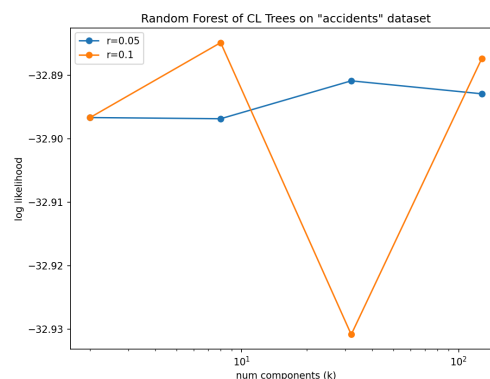
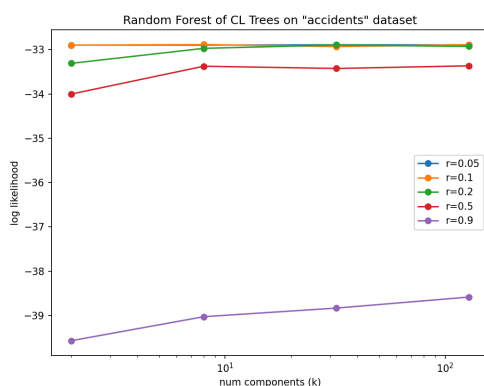
I implemented EM to learn a mixture of Chou-Liu Trees. For each dataset, I tried with different number of trees in the mixture -- $k \in \{2, 5, 10, 20\}$. However, a strange result is shown where the mixture will always converge to the equivalence of a single tree! Specifically, the log-likelihood on the training set converges only after 2 iterations¹.

Before trusting these results, I speculate possible issues with my implementation as follows. Firstly, I randomly initialized the trees in the mixture by letting each of them learn the structure and the parameters from a random 1% subset of the dataset. Can this produce too similar trees that are similar to the tree in the single case? More investigation is needed and is out of scope of this assignment. Secondly, there might simply be bugs in my mixture-of-trees implementation. However, so far I have not found anything.

Random Forest of Chou-Liu Tree

A Random Forest of Chou-Liu Trees is parametrized by k , the number of trees, and r , the fractions of randomly selected variable pairs whose mutual information are set to 0.

I use the “accidents” dataset to select hyperparameters. I tried with $k \in \{2, 8, 32, 128\}$ and $r \in \{0.05, 0.1, 0.2, 0.5, 0.9\}$. Results in Figure ? show that performance is generally better for smaller r 's and larger k 's. The best performance on the evaluation set was achieved with $(k, r) = (8, 0.1)$.



¹ The convergence condition is that the difference between the last two log likelihoods is less than 0.00001.

Figure ??: Hyperparameter tuning on the “accidents” dataset

Using those best values of k and r , I ran the Random Forest of CLTs on all datasets and measured the performance on the test sets.

Final results

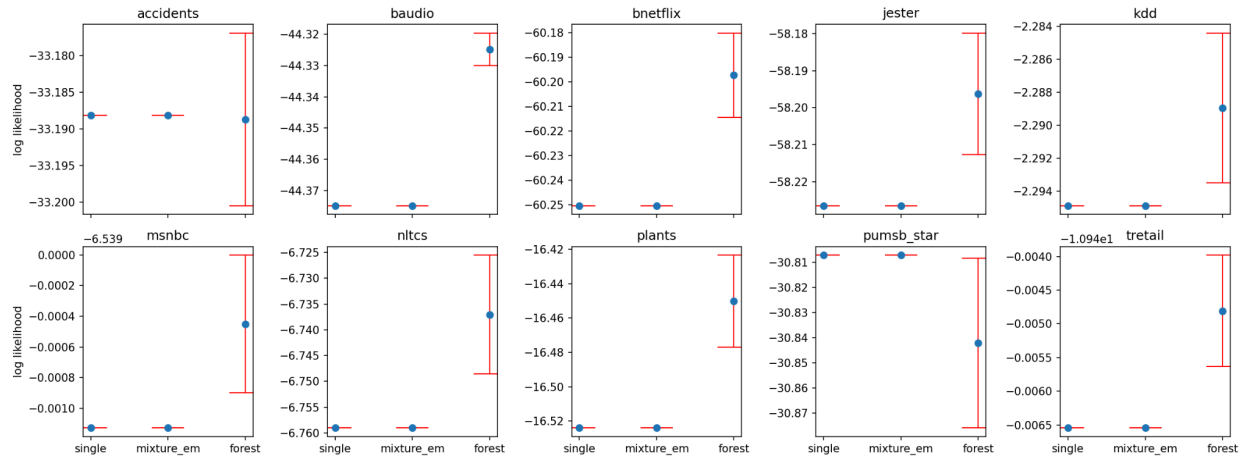


Figure ??: Final results of CLT experiments

The results are summarized in Figure ??. As we can see, mixtures of CLTs in my implementation does not differ from a single CLT. This is quite strange. I was expecting to see a lot more variation. However, I was not able to find any bugs in the code, so I would take the result as is.

Regarding random forests, there are 8 datasets where the random forest performs better. However, there is a dataset -- `pumsb_star` -- where it performs worse and one where the difference is not statistically significant.

If a ranking is needed, I would put Random Forest as the best algorithm, due to its superior performance across most of the datasets. Single tree and mixture of trees, would be equal.


Appendix A: Mutual Information and the Chow-Liu Algorithm

What is “mutual information” (MI)? a number showing how co-dependent two random variables are

- $I(A, B) = \dots$

Chow-Liu algorithm:

- Premises
 - Limit the Bayes net structure to trees
 - Set the objective to be minimizing the KL divergence between the joint probability distribution produced by the learned tree (after learning the parameter) vs the “actual” distribution.
 - What objective, in the context of trees, turns out to simply requires the tree to be the maximum spanning tree of graph G , where nodes are the variables, and the edges are their MI
- So the algorithm is:
 - Calculate $MI(A, B)$ for all A-B pairs
 - Get the maximum spanning tree (using Prim's), set as the Bayes net structure
 - Learn the parameters

Reference:  10-601 Machine Learning Spring 2015 - Lecture 14 . Some remarks about the lecture:

- Great lecturer. He is the author of our class's textbook.
- But he is tremendously humble, taking questions from students, seriously thinking of ways to answer the questions and really curious to know the answer, and openly admitting that he needs to work on paper later to find the answer.
- Students in that class (at CMU) asked very very good questions, looking at the algorithm/problem in various perspectives to get new insights, preparing for better generalization.