6.047 Problem Set 2 Writeup

Matthew Feng

October 8, 2018

1 Naive Bayes

(a) Naive Bayes Assumption

No, the naive Baye's assumption that the features are independent conditioned on the class does not hold in this case. This is because the complexity of the sequence is likely dependent on the length of the sequence, as well as the GC content. If the GC content of a sequence is high, then the sequence is more likely to be a CpG island, which would mean that the complexity is lower because of the repeated base ordering.

(b) Computing MLEs

Maximum Likelihood Estimates

	X_1	X_2	X_3
$P(\cdot \mid repeat)$	0, short 1, long	0, low content 0, medium content 1, high content	2/3, low complexity 1/3, high complexity
$P(\cdot \mid gene)$	0, short 1, long	2/3, low content 1/3, medium content 0, high content	1/3, low complexity 2/3, high complexity
$P(\cdot \mid motif)$	1, short 0, long	0, low content 1/2, medium content 1/2, high content	1/4, low complexity 3/4, high complexity

Prior probability distribution

$$P(Y) = \begin{cases} 3/10, & Y = repeat \\ 3/10, & Y = gene \\ 4/10, & Y = motif \end{cases}$$

(c) Predicted MAP Class

The maximum a posteriori estimate of class of $(X_1, X_2, X_3) = (long, medium, low)$ is $Y = gene$; we don't need to compute the denominator of Baye's theorem for two reasons: the other classes have probability of 0 in the numerator, and that the MAP estimate of class is proportional only to the numerator; the denominator is just a normalization factor that is the same for all classes. Therefore, if we are only doing classification, we can just compare proportions rather than exact values.

$$P(Y|X_1, X_2, X_3) = \frac{P(X_1, X_2, X_3|Y)P(Y)}{P(X_1, X_2, X_3)} \qquad \text{(Baye's theorem)}$$
$$= \frac{P(X_1|Y)P(X_2|Y)P(X_3|Y)P(Y)}{P(X_1, X_2, X_3)} \qquad \text{(Naive Baye's assumption)}$$

2 Classification of Conserved Regions

(a) Conditional probabilities

Alignment 1 $\log \mathbb{P}(S|N) = -17.098$, $\log \mathbb{P}(S|C) = -24.177$

Alignment 2 $\log \mathbb{P}(S|N) = -17.504$, $\log \mathbb{P}(S|C) = -13.256$

(b) Classification error I

The classification error, or amount of time that $P(S|C) > P(S|N)$ even though S is sampled from N, is 0.128 (12.8%).

(c) Classification error II

The classification error, or amount of time that $P(S|N) > P(S|C)$ even though S is sampled from C, is 0.1412 (14.1%).

(d) Reduce classification error

(i) Good discriminators

Score values 1 and 6 are good discriminators between the two models, because of the difference between the relative frequencies is high.

(ii) Bad discriminators

Score values 2 and 3 are poor discriminators between the two models, because of the difference between the relative frequencies is low.

The rate of classification errors would not decrease if we dismissed alignment scores of 0, because 0 is a good discriminator; alignments of 0 are twice as likely to occur in model N than in model C.

(e)

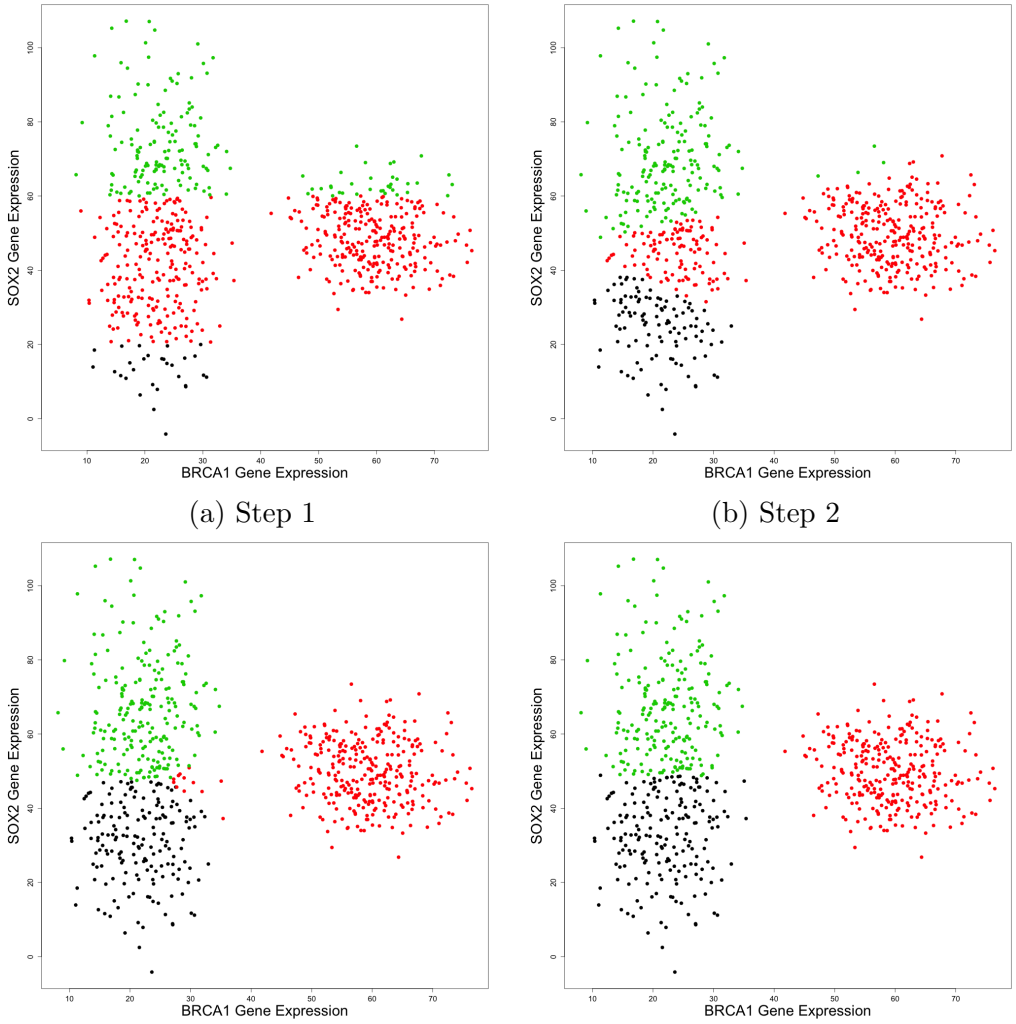
3 K-means Clustering

(a) k-means implementation

See `kmeans.py`.

(b) `tissue1` results

See Figure 1.



(a) Step 1

(b) Step 2

(c) Step 3

(d) Step 4

Figure 1: Four steps of convergence for the k-means algorithm on `tissue1_data.txt`.

(c) Improving k-means

See Figures 2 and 3. One of the flaws of k-means clustering is that it doesn't always find the most obvious clusters if the initial guesses for the centroid are not good. If we look at Figure 2, we can see that the clusters that were found were biased by the initial guesses for the centroids, which led the algorithm to find oddly-shaped clusters instead of the "obvious" ones that we see with our own eyes. In order to remedy this problem, it is beneficial to run k-means with random initialization, meaning that the initial guesses for the centroids are randomly placed in the feature space. In particular, k-means can be improved by running many times with random initialization, and then using the clustering that has the lowest average distance to the centroids.

(d) Fuzzy k-means

We can implement fuzzy k-means by assigning to each point in our dataset not a class, but a probability that it belongs to a certain class. We can do so by computing the probability that a certain point belongs to a cluster via a normal distribution (farther away is less probable, with a Gaussian probability dropoff). When we update the location of the centroids, we compute the weighted average of all the data points, with the weight of contribution for each data point weighted by the probability it belongs to the cluster. See `fuzzykmeans.py` for the implementation.

We could graphically represent the fuzzy clusters by mixing RGB colors; class 1 could be encoded as red, class 2 as green, and class 3 as blue. Depending on the probability that the point belongs do class i, we could use function (the CSS function) `rgb(r, g, b)` to find a representative color for the probabilities.

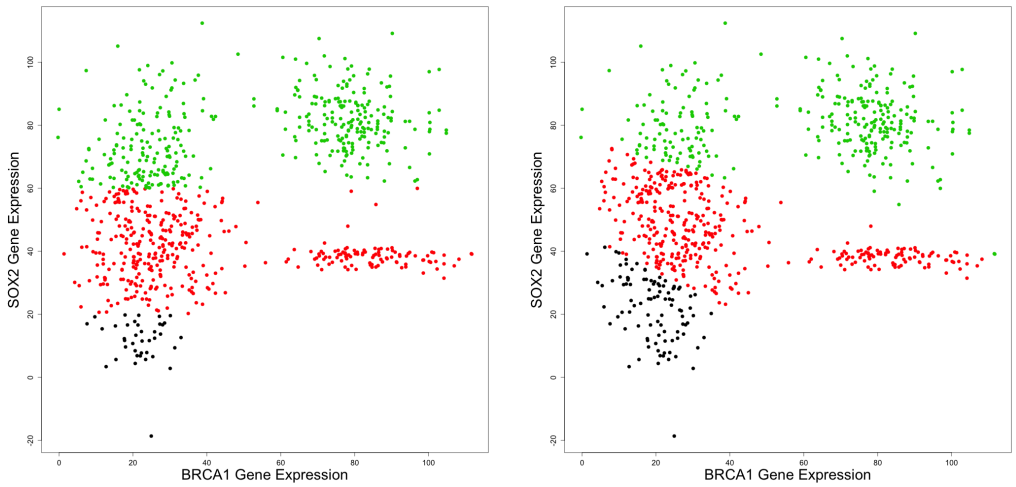
4 Final project preparation

(b) Evaluate proposals

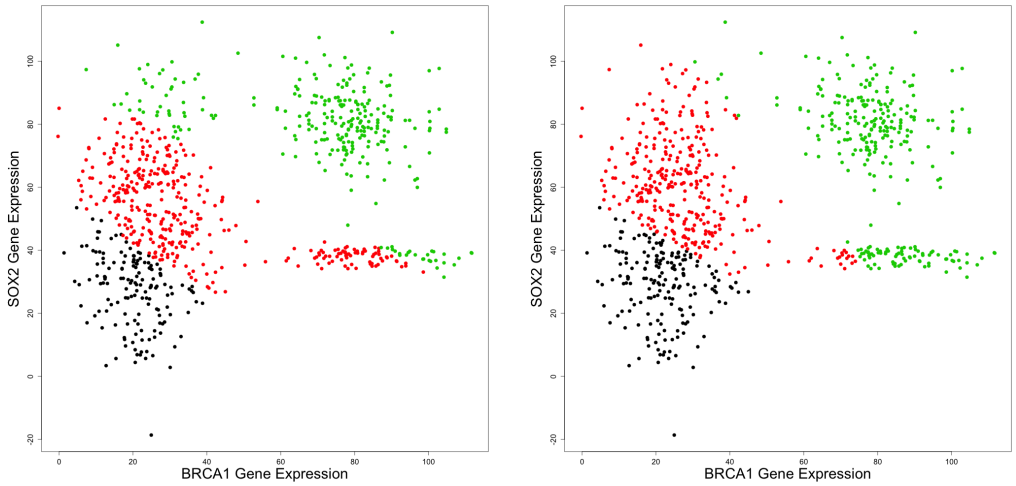
(c) Evaluate scientific papers

(d) Project idea

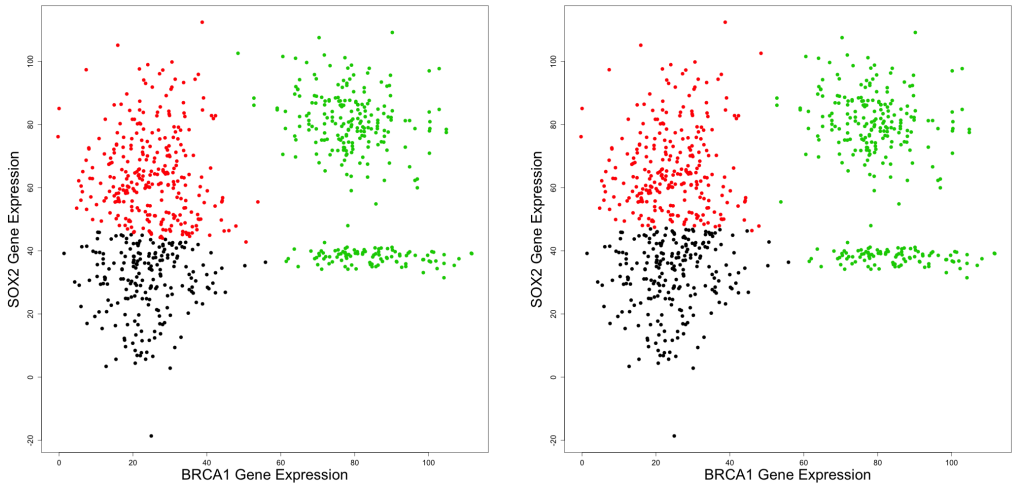
(e) Potential project partners



(a) Step 1 (b) Step 2

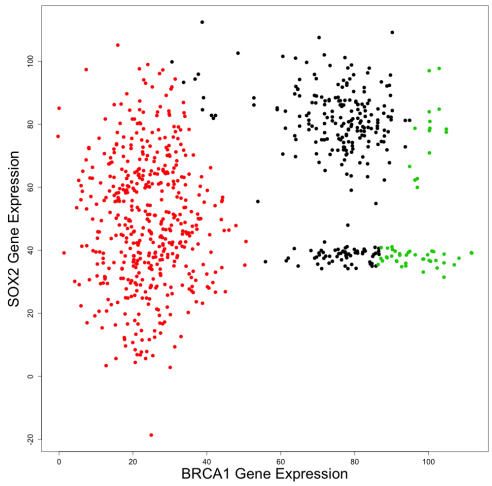


(c) Step 3 (d) Step 4

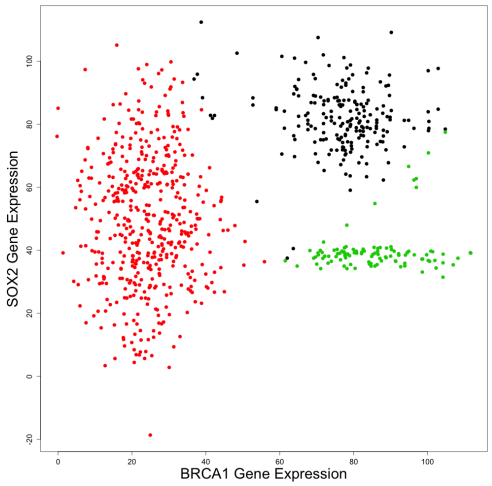


(c) Step 5 (d) Step 6

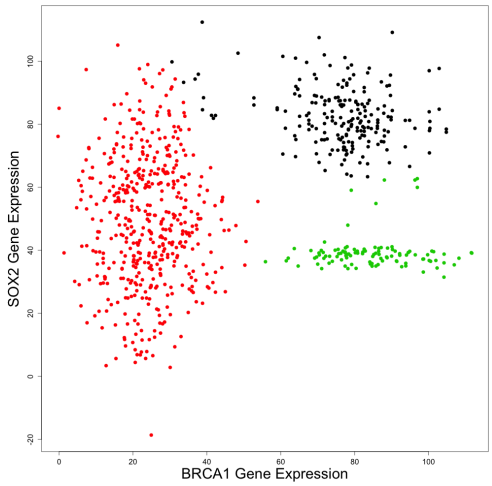
Figure 2: Six steps of convergence for the k-means algorithm on `tissue2_data.txt`.



(a) Step 1



(b) Step 2



(c) Step 3

Figure 3: Three steps of convergence for the k-means algorithm with random initialization on `tissue2_data.txt`.