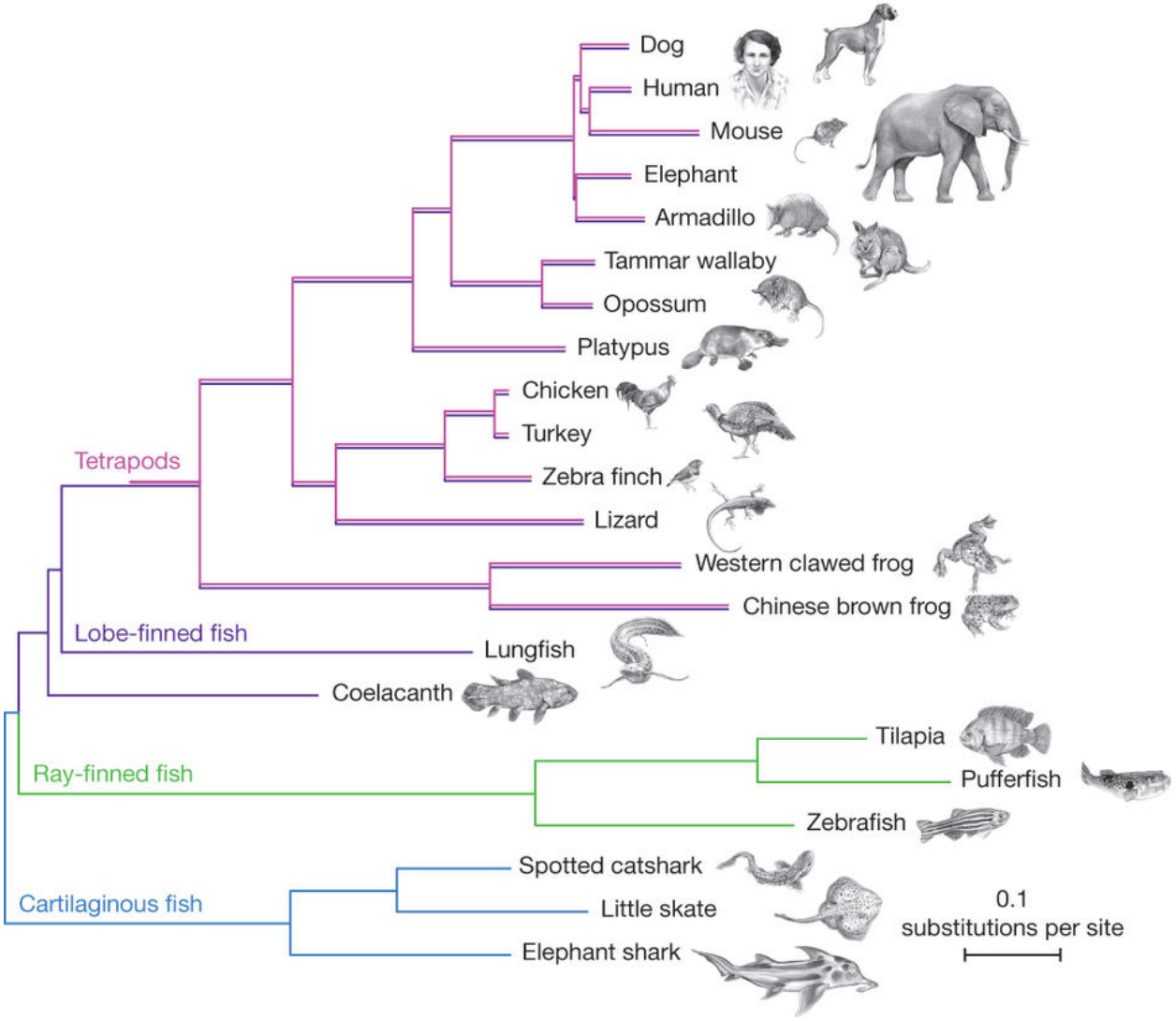
CS 189 Introduction to Machine Learning
Spring 2023 Jonathan Shewchuk

DIS12

1 Hierarchical Clustering for Phylogenetic Trees

A phylogenetic tree (or "evolutionary tree") is way of representing the branching nature of evolution. Early branches represent major divergences in evolution (for example, modern vertebrae diverging from modern invertebrate), while later branches represent smaller branches in evolution (for example, modern humans diverging from modern monkeys). An example is shown below.



Creating phylogenetic trees is a popular problem in computational biology. We are going to combine what we know about clustering, decision trees, and unsupervised learning.

We start with all the samples (in this case, animals) in a single cluster and gradually divide this up, until each animal is in its own cluster. This should remind you of decision trees! After k steps, we have at most 2^k clusters. Since we do not have labels, we need to find some way of deciding how to split the samples (other than using entropy).

We will use the same objective as in k-means clustering to determine how good our proposed clustering is (here, $|x|$ denotes length and $\|x\|$ denotes norm):

$$\min L = \sum_{i=1}^{k} \sum_{X_j \in S_i} \|X_j - \mu_i\|_2^2, \text{ where}$$
$$\mu_i = \frac{1}{|S_i|} \sum_{X_j \in S_i} X_j, \quad i = 1, 2, \ldots, k.$$

At each iteration, we will split each cluster with more than one element into two clusters, choosing the split that achieves the minimum resulting objective value L. The algorithm terminates when every sample point is in its own cluster, yielding an objective value $L = 0$.

(a) Consider the following six animals and their two features. Create the resulting decision tree.

Animal	Lifespan	Wings
Dog	12	0
Human	80	0
Mouse	2	0
Elephant	60	0
Chicken	8	2
Turkey	10	2

Solution: At each step, we are trying to minimize the squared distance from each animal to its cluster center. Given a feature, to find the best threshold value we can start by ordering the animals on that feature (like we normally would for a decision tree) and taking the midpoints between animals as possible thresholds. Since there is only one split available for wings, for each cluster we can easily verify which feature will yield a better split by simply checking if any better split exists on lifespan. At the root, lifespan is the better feature to split on. This yields five possible threshold values. By testing each value, we can verify that the midpoint of dog and elephant (36) yields the best objective value. Repeating this process, we obtain the following decision tree:

Root split: set a threshold of 36 on lifespan, yielding (Dog, Mouse, Chicken, Turkey) vs (Human, Elephant).

Depth 1: in the left branch (Dog, Mouse, Chicken, Turkey), set a threshold of 5 on lifespan to get (Mouse) vs (Dog, Chicken, Turkey); in the right branch (Human, Elephant), set a threshold of 70 on lifespan to get (Human) vs (Elephant).

Depth 2: from (Dog, Chicken, Turkey), set a threshold of 11 on lifespan to get (Dog) vs (Chicken, Turkey).

Depth 3: from (Chicken, Turkey) set a threshold of 9 on lifespan to get (Chicken) vs (Turkey).

(b) Prove that an optimal clustering on $k+1 < n$ clusters has an objective value that is at least as small as that of the optimal clustering on k clusters.

Solution: Let $\{S_1, \ldots, S_k\}$ denote a clustering with k clusters. If this clustering already has objective value 0, then, since the clustering objective is a sum of non-negative terms, each $\sum_{x_j \in S_i} \|x_j - \mu_i\|^2 = 0$, which implies that $x_j = \mu_i$ for all $x_j \in S_i$, for every S_i. Thus for any $|S_i| \geq 2$ we can clearly split S_i into two clusters, each with the same mean, whose objective value is also 0.

Suppose that the clustering $\{S_1, \ldots, S_k\}$ has objective value greater than 0. Choose any cluster S_i with non-zero cost, which implies both that $|S_i| \geq 2$ and that there exists an element $x_j \neq \mu_i \in S_i$. We can construct a new clustering by splitting S_i into two clusters $\{x_j\}$ and $S_i - \{x_j\}$. The former cluster clearly has cost 0, since we can take $\mu = x_j$. The latter cluster $S_i - \{x_j\}$ has new mean $\mu_i' = \frac{1}{|S_i - \{x_j\}|} \sum_{x_k \in S_i - \{x_j\}} x_k$. The cluster $S_i - \{x_j\}$ has lower cost than S_i since

$$\begin{aligned} \sum_{x_k \in S_i - \{x_j\}} \|x_k - \mu_i'\|^2 &\leq \sum_{x_k \in S_i - \{x_j\}} \|x_k - \mu_i\|^2 \\ &\leq \sum_{x_k \in S_i} \|x_k - \mu_i\|^2. \end{aligned}$$

The first inequality follows from the fact that the mean of a set of points minimizes the sum of the Euclidean distances to those points. To see this, take the derivative of the objective with respect to μ and solve for the minimum. The second inequality follows from adding one additional non-negative term.

2 Kernel k-means

Suppose we have a dataset $\{x_i\}_{i=1}^N, x_i \in \mathbb{R}^n$ that we want to split into K clusters. Furthermore, suppose we know a priori that this data is best clustered in a large feature space $\mathbb{R}^m$, and that we have a feature map $\phi : \mathbb{R}^n \to \mathbb{R}^m$. How should we perform clustering in this space?

(a) Write the objective for K-means clustering in the feature space (using the squared L_2 norm in the feature space). Do so by explicitly constructing cluster centers $\{\mu_k\}_{k=1}^K$ with all $\mu_k \in \mathbb{R}^m$.

Solution:

$$L = \sum_{k=1}^{K} \sum_{x_i \in S_k} \|\phi(x_i) - \mu_k\|^2$$

(b) Write an algorithm that minimizes the objective in (a). **Solution:**

1. Compute $\phi(x_i)$ for every point x_i.

2. Do the standard k-means on $\{\phi(x_i)\}$.

(c) Write an algorithm that minimizes the objective in (a) without explicitly constructing the cluster centers $\{\mu_k\}$. Assume you are given a kernel function $\kappa(x, y) = \phi(x) \cdot \phi(y)$.

Solution:

We proceed by coordinate descent on the objective in (a). First, given a clustering, the setting of μ_i that minimizes L is

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} \phi(x)$$

Second, given a setting of the μ's, the optimal clustering is given by assigning x_i to the cluster $\arg\min_{1 \le k \le K} f(i, k)$, where

$$f(i, k) = \|\phi(x_i) - \mu_k\|^2$$

To kernelize this, we write

$$f(i, k) = \phi(x_i) \cdot \phi(x_i) - 2\phi(x_i) \cdot \mu_k + \mu_k \cdot \mu_k$$

Substituting the setting of μ_k,

$$= \phi(x_i) \cdot \phi(x_i) - \frac{2}{|S_k|} \sum_{x_j \in S_k} \phi(x_i) \cdot \phi(x_j) + \frac{1}{|S_k|^2} \sum_{x_j, x_l \in S_k \times S_k} \phi(x_j) \cdot \phi(x_l)$$

Now we can replace the inner products with kernel evaluations

$$= \kappa(x_i, x_i) - \frac{2}{|S_k|} \sum_{x_j \in S_k} \kappa(x_i, x_j) + \frac{1}{|S_k|^2} \sum_{x_j, x_l \in S_k} \kappa(x_j, x_l)$$

This yields the following algorithm:

1. Compute the kernel matrix $G_{ij} = \kappa(x_i, x_j)$.

2. Start with an initial clustering $\{S_k\}$.

3. Compute the new cluster index for each x_i as $\arg\min_{1 \le k \le K} f(i, k)$.

4. Update $\{S_k\}$

5. Repeat steps (3) and (4) until convergence.

3 Regularization and Kernel k-Means

Recall that in k-means clustering we attempt to minimize the objective

$$\min_{C_1, C_2, \ldots, C_k} L = \sum_{i=1}^{k} \sum_{X_j \in C_i} \|X_j - \mu_i\|_2^2, \text{ where}$$

$$\mu_i = \text{argmin}_{\mu_i \in \mathbb{R}^d} \sum_{X_j \in C_i} \|X_j - \mu_i\|_2^2 = \frac{1}{|C_i|} \sum_{X_j \in C_i} X_j, \quad i = 1, 2, \ldots, k.$$

The sample points are $\{X_1, \ldots, X_n\}$, where $X_j \in \mathbb{R}^d$. C_i is the set of sample points assigned to cluster i and $|C_i|$ is the cluster's cardinality. Each sample point is assigned to exactly one cluster.

(a) What is the minimum value of the objective when $k = n$ (the number of clusters equals the number of sample points)?

Solution: The value is 0, as every point can have its own cluster.

(b) (Regularized k-means) Suppose we add a regularization term to the above objective. The regularized k-means objective is now

$$\min_{\mu_i \in \mathbb{R}^d} \sum_{i=1}^{k} \left(\lambda \|\mu_i\|_2^2 + \sum_{X_j \in C_i} \|X_j - \mu_i\|_2^2 \right).$$

Show that the optimum of

$$\min_{\mu_i \in \mathbb{R}^d} \lambda \|\mu_i\|_2^2 + \sum_{X_j \in C_i} \|X_j - \mu_i\|_2^2$$

is obtained at

$$\mu_i = \frac{1}{|C_i| + \lambda} \sum_{X_j \in C_i} X_j.$$

Solution:

For a cluster i, consider the function

$$f(\mu_i) = \left(\sum_{X_j \in C_i} \|X_j - \mu_i\|_2^2 \right) + \lambda \|\mu_i\|_2^2.$$

Its gradient with respect to μ_i is

$$\begin{aligned} \nabla_{\mu_i} f(\mu_i) &= \left(2 \sum_{X_j \in C_i} (\mu_i - X_j) \right) + 2\lambda \mu_i \\ &= 2 \left((|C_i| + \lambda)\mu_i - \sum_{X_j \in C_i} X_j \right). \end{aligned}$$

Setting it to zero, we have $\mu_i = \frac{1}{|C_i|+\lambda} \sum_{X_j \in C_i} X_j$. As the function f is convex, the minimum is obtained at the critical point, $\mu_i = \frac{1}{|C_i|+\lambda} \sum_{X_j \in C_i} X_j$.

(c) (Kernel k-means) Suppose we have a dataset $\{X_i\}_{i=1}^n$, $X_i \in \mathbb{R}^\ell$ that we want to split into k clusters, i.e., finding the best k-means clustering (without regularization). Furthermore, suppose we know *a priori* that this data is best clustered in an impractically high-dimensional feature space

$\mathbb{R}^m$ with an appropriate metric. Fortunately, instead of having to deal with the (implicit) feature map $\Phi : \mathbb{R}^\ell \to \mathbb{R}^m$ and (implicit) distance metric[1], we have a kernel function $\kappa(X_1, X_2) = \Phi(X_1) \cdot \Phi(X_2)$ that we can compute quickly on the raw samples. How should we perform the kernelized counterpart of k-means clustering?

Derive the missing portion of this algorithm, and show your work in deriving it. The primary issue to consider is that although we define the means μ_i in the usual way, we can't ever compute Φ explicitly because it's way too big. Therefore, in the step where we determine which cluster each sample point is assigned to, we must use the kernel function κ to obtain the right result. Review the lecture on kernels if you don't remember how that's done.

Algorithm 1: Kernel k-means

Require: Data matrix $X \in \mathbb{R}^{n \times d}$; number of clusters K; kernel function $\kappa(X_1, X_2)$
Ensure: Cluster class(j) assigned for each sample point X_j.
function KERNEL-K-MEANS(X, K)
  Randomly initialize class(j) to be an integer in $1, 2, \ldots, K$ for each X_j.
  while *not converged* **do**
    for $i \leftarrow 1$ ***to*** K **do**
      Set $S_i \leftarrow \{j \in \{1, 2, \ldots, n\} : \text{class}(j) = i\}$.
    for $j \leftarrow 1$ ***to*** n **do**
      Set class(j) $\leftarrow \arg\min_k$(Action item: derive the missing portion here)
  Return S_i for $i \leftarrow 1, 2, \ldots, K$.
end function

Solution: Given a clustering S_i, we define

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} \Phi(x)$$

to minimize $\sum_{x \in S_i} \|\Phi(x) - \mu_i\|_2^2$. (But we don't explicitly compute μ_i!)

Given this choice of the μ_k's, the K-means clustering is found by assigning each X_i to the cluster $\arg\min_k f(i, k)$, where

$$\begin{aligned} f(i,k) &= \|\Phi(X_i) - \mu_k\|^2 \\ &= \Phi(X_i) \cdot \Phi(X_i) - 2\Phi(X_i) \cdot \mu_k + \mu_k \cdot \mu_k \\ &= \Phi(X_i) \cdot \Phi(X_i) - \frac{2}{|S_k|} \sum_{X_j \in S_k} \Phi(X_i) \cdot \Phi(X_j) + \frac{1}{|S_k|^2} \sum_{X_j \in S_k} \sum_{X_l \in S_k} \Phi(X_j) \cdot \Phi(X_l) \\ &= \kappa(X_i, X_i) - \frac{2}{|S_k|} \sum_{X_j \in S_k} \kappa(X_i, X_j) + \frac{1}{|S_k|^2} \sum_{X_j \in S_k} \sum_{X_l \in S_k} \kappa(X_j, X_l). \end{aligned}$$

[1] Just as how the interpretation of kernels in kernelized ridge regression involves an implicit prior/regularizer as well as an implicit feature space, we can think of kernels as generally inducing an implicit distance metric as well. Think of how you would represent the squared distance between two points in terms of pairwise inner products and operations on them.

The first term does not vary with k, so we can drop it. Therefore, we write

$$\text{class}(i) = \arg\min_k \left(\frac{1}{|S_k|^2} \sum_{X_j, X_l \in S_k} \kappa(X_j, X_l) - \frac{2}{|S_k|} \sum_{X_j \in S_k} \kappa(X_i, X_j) \right). \tag{1}$$

(d) The expression you derived may have unnecessary terms or redundant kernel computations, especially when you compute class(j) for every $j \in [1, n]$. Explain how to eliminate them; that is, how to perform the computation quickly without doing irrelevant computations or redoing computations already done.

Solution: First, we dropped the term $\kappa(X_i, X_i)$. Second, observe that the first summation above is the same for every sample point X_i, so we can compute it once for each cluster per iteration, and use it with every sample point. Third, observe that we need to compute a kernel computation for every pair of sample points (not necessarily distinct), but we only need to do it once at the beginning of the algorithm. This amounts to computing every entry of the kernel matrix, keeping in mind that the kernel function (and the kernel matrix) is symmetric to avoid redundant computations. Then the computed kernel values can be used multiple times per iteration, in both the first and second summations.