CMSC 25025 / STAT 37601

Machine Learning and Large Scale Data Analysis

Assignment 3

Due: Thursday, April 25, 2013

This assignment consists of three problems. Al of them are "pencil and paper" problems. They are intended to be short, and to help you with the programming project that you are currently working on.

1. *Feeling the Heat* (30 points)

   The Laplacian is the differential operator given by the the sum of the second derivatives

   $$\Delta f(x) = \frac{\partial^2 f(x)}{\partial x_1^2} + \frac{\partial^2 f(x)}{\partial x_2^2} + \cdots \frac{\partial^2 f(x)}{\partial x_d^2}$$

   for a function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$. One of the many uses of the Laplacian is to model the diffusion of heat through a substance.

   Recall that for a graph on n nodes, we suppose we have a weight w_{ij} assigned to edge (i, j). Assume the weights are symmetric, so that $w_{ij} = w_{ji}$, and let $W = [w_{ij}]$ be the $n \times n$ weight matrix. The *graph Laplacian* is the $n \times n$ matrix defined by

   $$\Delta = D - W. \tag{1}$$

   Show that

   $$\sum_{ij} w_{ij}(f_i - f_j)^2 = f^T \Delta f$$

   for any vector $f = (f_1, f_2, \ldots, f_d)$.

2. *Working in Harmony* (30 points)

   Now recall our harmonic function semi-supervised learning algorithm. We have some points x_i labeled $y_i = 1$, some points labeled $y_i = 0$, and many points without any label. We form a weighted graph with graph Laplacian $\Delta = D - W$. We then estimate a value f_j for each node j by minimizing

   $$\sum_{ij} w_{ij}(f_i - f_j)^2 = f^T \Delta f$$

   subject to $f_i = y_i$ if x_i is labeled y_i. Show that the solution is *harmonic*; namely,

   $$f_i = \frac{\sum_j w_{ij} f_j}{\sum_j w_{ij}}$$

   for each node i.

3. *Loving Your Neighbors* (40 points)

   Suppose we have n points $\{x_i\}_{i=1}^n$ and a distance $d_{ij} = d_{ji} = \text{dist}(x_i, x_j)$ between each pair.

   The k nearest neighbors of a point x_i are the k points closest to x_i, in terms of the distance d_{ij}. The k-nearest neighbor graph G_k has n nodes, one for each point x_i, and an edge between i and j if either x_j is one of the k nearest neighbors of x_i, or if x_i is one of the k nearest neighbors of x_j.

   Derive an algorithm that computes the smallest k such that G_k is a connected graph. Sketch the algorithm in pseudo-code. What is the complexity of the algorithm?