Optional Coding Assignment

CS 205A: Mathematical Methods for Robotics, Vision, and Graphics (Spring 2015)
Stanford University
Due Wednesday, June 3, 2:15pm (**no late days permitted**)

This coding assignment is completely optional. If you choose to complete it, the grade you receive on this assignment will replace your lowest homework grade (assuming it is lower). Please email your code and a writeup PDF to **cs205a@gmail.com** by the deadline above.

Problem 1 (50 points)**.** A *graph* is a data structure $G = (V, E)$ consisting of n vertices in a set $V \cong \{1, \ldots, n\}$ and a set of (undirected) edges $E \subseteq V \times V$. A common problem is *graph layout*, where we choose positions of the vertices in V on the plane $\mathbb{R}^2$ respecting the connectivity of G. For this problem we will assume $(i, i) \notin E$ for all $i \in V$.

(a) Take $\vec{v}_1, \ldots, \vec{v}_n \in \mathbb{R}^2$ to be the positions of the vertices in V; these are the unknowns in graph layout. The Dirichlet energy of a layout is

$$E(\vec{v}_1, \ldots, \vec{v}_n) = \sum_{(i,j) \in E} \|\vec{v}_i - \vec{v}_j\|_2^2.$$

Suppose an artist specifies positions of vertices in a nonempty subset $V_0 \subseteq V$. We will label these positions as $\vec{v}_k^0$ for $k \in V_0$. Derive two $(n - |V_0|) \times (n - |V_0|)$ linear systems satisfied by the x and y components of the unknown $\vec{v}_i$'s solving the following minimization problem:

$$\begin{aligned} &\text{minimize } E(\vec{v}_1, \ldots, \vec{v}_n) \\ &\text{subject to } \vec{v}_k = \vec{v}_k^0 \ \ \forall k \in V_0 \end{aligned}$$

HINT: Your answer should yield two linear systems $A\vec{x} = \vec{b}_x$ and $A\vec{y} = \vec{b}_y$; that is, the two systems should use the same matrix. A should be sparse, symmetric, and positive definite.

(b) Complete `problem1.m` to solve this system in Matlab:

(i) Compute the matrix A as a variable `A` and put both right hand sides $\vec{b}_x$ and $\vec{b}_y$ into the columns of the $(n - |V_0|) \times 2$ matrix `rhs`. Make sure that you use a sparse matrix for `A`.

(ii) Implement gradient descent (§11.1) to compute the unknown positions iteratively; you should update the x and y components simultaneously. The starter code will help you visualize the output during each iteration. For this and the next part, you will need to formulate and implement reasonable convergence conditions.

(iii) Implement conjugate gradients (§11.2.4) for the same problem.

(iv) In your writeup, compare the number of iterations needed to reach a reasonable graph layout using both strategies. In particular, plot the norm of the residual as a function of iteration for the two methods applied to the test data in `unitCircleFEM.mat`.

EC. Implement preconditioned conjugate gradients using a preconditioner of your choice. How much does convergence improve? (5 points)

NOTE: Two test cases are provided. You may wish to use the smaller example to debug, but make sure your code gives proper output on the larger example.

Problem 2 (50 points)**.** In this problem, you will implement a method for approximating the "earth mover's distance" (EMD) between histograms. For this problem, we define a *histogram* as a vector $\vec{p} \in [0,1]^n$ satisfying $\vec{1}^\top \vec{p} = 1$ (that is, $\vec{p}$ is a set of non-negative values summing to 1).

(a) Suppose we are given $D \in \mathbb{R}^{n \times n}_+$ satisfying $D^\top = D$. D_{ij} measures the distance between bins i and j of a histogram with n bins; for instance, if the histograms are in one dimension, we might define $D_{ij} = |i - j|$. The EMD between histograms $\vec{p}$ and $\vec{q}$ is

$$\mathcal{W}(\vec{p}, \vec{q}) \equiv \left\{ \begin{array}{rl} \min_{T \in \mathbb{R}^{n \times n}} & \sum_{i=1}^n \sum_{j=1}^n T_{ij} D_{ij} \\ \text{subject to} & T_{ij} \geq 0 \ \ \forall i, j \in \{1, \ldots, n\} \\ & \sum_j T_{ij} = p_i \ \ \forall i \in \{1, \ldots, n\} \\ & \sum_i T_{ij} = q_j \ \ \forall j \in \{1, \ldots, n\}. \end{array} \right.$$

In your writeup, explain what $\mathcal{W}(\vec{p}, \vec{q})$ measures about the difference between $\vec{p}$ and $\vec{q}$, and briefly describe an application for its computation. For this part, you are welcome to search online for discussion of this metric.

(b) EMD is difficult to compute when n is large. For $\vec{p}, \vec{q} \in (0,1]^n$ and $\alpha > 0$, an alternative is the *entropy-regularized EMD*, given by

$$\mathcal{W}_\alpha(\vec{p}, \vec{q}) \equiv \left\{ \begin{array}{rl} \min_{T \in \mathbb{R}^{n \times n}} & \sum_{i=1}^n \sum_{j=1}^n T_{ij} D_{ij} + \alpha \sum_{ij} T_{ij} \ln T_{ij} \\ \text{subject to} & \sum_j T_{ij} = p_i \ \ \forall i \in \{1, \ldots, n\} \\ & \sum_i T_{ij} = q_j \ \ \forall j \in \{1, \ldots, n\}. \end{array} \right.$$

Define a matrix K in terms of D and α so that the objective for computing $\mathcal{W}_\alpha(\vec{p}, \vec{q})$ can be written as $\alpha \cdot \mathrm{KL}(T|K)$, where the *KL divergence* between $A, B \in \mathbb{R}^{n \times n}_+$ is

$$\mathrm{KL}(A|B) \equiv \sum_{ij} A_{ij} \ln \frac{A_{ij}}{B_{ij}}.$$

(c) Show that the optimal matrix T in the minimization for $\mathcal{W}_\alpha$ can be written $T = \mathrm{diag}(\vec{v}) K \mathrm{diag}(\vec{w})$ for some $\vec{v}, \vec{w} \in \mathbb{R}^n$. Here, $\mathrm{diag}(\vec{z})$ is the matrix with diagonal elements $\vec{z}$.

(d) The *Sinkhorn algorithm* for computing $\mathcal{W}_\alpha$ (introduced by M. Cuturi) proceeds as follows:

1. Initialize $T^{(0)} \equiv K$.
2. For $i = 1, 2, 3, \ldots$
   I. If i is odd, compute

$$T^{(i)} \equiv \left\{ \begin{array}{rl} \arg\min_{T \in \mathbb{R}^{n \times n}} & \mathrm{KL}(T|T^{(i-1)}) \\ \text{subject to} & \sum_j T_{ij} = p_i \ \ \forall i \in \{1, \ldots, n\}. \end{array} \right.$$

   II. If i is even, compute

$$T^{(i)} \equiv \left\{ \begin{array}{rl} \arg\min_{T \in \mathbb{R}^{n \times n}} & \mathrm{KL}(T|T^{(i-1)}) \\ \text{subject to} & \sum_i T_{ij} = q_j \ \ \forall j \in \{1, \ldots, n\}. \end{array} \right.$$

Show that each $T^{(i)}$ can be written $T^{(i)} = \mathrm{diag}(\vec{v}^{(i)}) K \mathrm{diag}(\vec{w}^{(i)})$ for some vectors $\vec{v}^{(i)}, \vec{w}^{(i)} \in \mathbb{R}^n$. Write the steps of the Sinkhorn algorithm in terms of these vectors; your algorithm should involve only matrix-vector multiplies and per-element operations (multiplication/division).

(e) Implement the Sinkhorn algorithm in `EMD.m`, including reasonable stopping criteria.

(f) In your writeup, illustrate the behavior of your EMD implementation by devising several tests when $D_{ij} = |i - j|$.

Problem 3 (Up to 30 points extra credit; open-ended)**.** Describe and implement a nontrivial application of the SVD to another field in computer science, such as computer vision, graphics, or machine learning. Many possible ideas exist, and your work will be graded based on creativity and use of numerical methods (you can use Matlab's `svd` command if you wish). Some ideas:

- The "As-Rigid-As-Possible" method described in Example 10.5 (difficult!).
- The method described in "Analyzing Oriented Patterns" by Kass and Witkin for finding dominant feature directions in an image.
- Eigenfaces for face recognition described in §7.2.6.
- The "iterative closest point" (ICP) algorithm for aligning point clouds.
- The semidefinite embedding technique described in §10.4.3 (use the `cvx` Matlab package for easy but slow semidefinite programming).
- Optimization techniques that project Hessian matrices onto the cone of semidefinite matrices.
- "Latent semantic indexing" (LSI) for processing collections of text.
- Metric learning for comparing data points with different units in different dimensions.

Feel free to contact the course staff for ideas in your particular area of interest.